

BIOGRAPHICAL INFORMATION

Steve Grisé
Product Manager
ESRI

Specific Responsibilities

Mr. Grisé joined ESRI in 1999. In the past 5 years his focus has been on developing a set of industry-driven data models in 30 different industries and scientific disciplines. He has also worked extensively with ESRI utility and telecommunications business partners that are developing solutions using ESRI technology. Currently his focus is on working with Enterprise project teams to develop and implement their GIS technology strategy.

Past Experience

15 years experience developing large GIS and systems integration projects for utilities and local government. Experience in project implementation and software development.

Educational Information

BA – Geography. University of Winnipeg

WEB SERVICES TECHNOLOGY - A REPLACEMENT FOR DESKTOP GIS?

Steve Grisé
ArcGIS Product Manager, ESRI

380 New York St.
Redlands, CA 92373
Telephone (909) 793-2853
Fax: (909) 793-5953
E-mail: sgrise@esri.com

Learning Objectives

1. Learn how web services will impact enterprise GIS
2. Learn recent trends in commercial software to support web services
3. Discover the advantages for use of web services over traditional techniques

ABSTRACT

Traditional departmental and enterprise implementations of geographic information systems involved spatial data being stored on a server and applications running on client desktop computers. With the evolution of web services technology such as XML, SOAP and Service Oriented Architectures in mainstream GIS software, the traditional approach to GIS software deployment is changing. This paper will present how web services will significantly impact traditional client-server architectures with more flexible and open solutions.

INTRODUCTION

Web Services are the next wave of software technology that will impact GIS users, developers, project managers, and system integrators. Web Services can be used to augment existing computer systems, and in the context of Enterprise systems web services promise to:

1. Enable a shift towards more flexible and open architectures,
2. Simplify integration,
3. Provide new opportunities for application developers.

For project leaders in local government and utility organizations, the amount of raw information available on web services is overwhelming. With all of this information, however, the tendency is for that information to be oriented towards programmers. It is difficult to understand how this technology will impact existing GIS/IT application architectures. This is creating confusion about web services, service oriented architectures, server-based web applications, and client based web applications. A key goal of this paper is to clarify some of the terms and definitions so that we can start to talk more clearly about how project teams can use web services.

Specifically, many people have made a superficial observation that we are heading “back to the mainframe days” in terms of computing infrastructure and applications. In this new environment the web browser replaces the “dumb terminal”, and a more sophisticated, loosely coupled set of web services is the new “mainframe”. It is a logical conclusion that web-based applications may replace desktop GIS applications, and that web services technology will drive this new type of architecture into existing GIS environments. From a pure technology standpoint, however, the analogy doesn’t work well.

This paper provides definitions for Web Services and Web Applications, and then explores current GIS/IT architectures and the potential uses of web services technology. The final conclusion is that Desktop GIS will not be significantly impacted by this new technology, and that the real benefits of web services will be realized in areas such as embedded applications and system integration.

WEB SERVICES DEFINITION

Before getting into the details of web services for GIS applications, it is important to have specific definitions. In this paper so far, “Web Services” has been used as a phrase to describe a number of related technologies, but at a detailed level Web Services are a focused technology for developers. At a simplistic level, one way to think of web services is that they are web-enabled shared programming library (like a .dll). Earlier web services definitions had an object-oriented emphasis: people imagined web interfaces as a set of objects and methods that could be accessed across network and computer platform boundaries in a relatively *tightly-coupled* architecture.

This approach was roughly equivalent to previous solutions such as remote procedure call (RPC). In this context *tightly-coupled* means that programs participating in the architecture had to understand a significant amount of detail of the implementation on the remote computer system for communication to occur. For example, to get a map from a remote system, one would have to know the details of the objects being called and some detail about the response I would expect to receive. The pseudocode could look something like:

```
MyMap = execute remotesystem.object.drawmethod lowerx, lowery, upperx, uppery,
```

In this example the calling program has to know that “drawmethod” exists and the parameters required by drawmethod. This approach is tightly coupled because it requires a level of knowledge in the calling program that is tightly tied to the implementation on the remote system. While interface description languages were developed to abstract some of this knowledge from the calling program, typically this required a high level of communication that programmers could bypass once they understood the semantics of the published API. If the API definition changed, a calling program could stop working. From a maintainability standpoint, the result was little better than RPC or previous attempts at interoperability because the interfaces were too fine grained, and developers knew too much about the system they were integrating with.

After years of work on CORBA, DCOM, COM+, and finally World Wide Web Consortium (W3C) specifications, the major industry players have agreed to a simpler

approach. This is being accompanied by significant investments in platform software development. The basis of these Service-Oriented Architectures is [Box, He]:

1. XML (with schema languages such as XML Schema and RelaxNG)
2. HTTP (get, put, etc.)
3. SOAP (Service descriptions in Web Services Description Language (WSDL))

The important lessons here are that you must only implement these capabilities to participate in *Service-Oriented* architectures. The intent is to provide a simpler, higher level of integration.

Techniques for more interoperable systems have been changing rapidly. There is movement to rename the acronym SOAP from the words “Simple Object Access Protocol” to “Service Oriented Architecture and Programming”. This is a rapidly changing definition that at the time of writing the abstract for this paper SOAP had the first meaning, at time of writing the paper SOAP is not an acronym at all, and by the time this paper is presented SOAP will most likely refer to the service-oriented definition. At one level this may just appear to be semantics, but there is a significant shift occurring from object-based architectures to a services-based architectures. The approach being taken is for systems participating in this architecture to accept formatted XML messages and respond using formatted XML messages. This may seem like a subtle shift, but it is important from an interoperability standpoint. The pseudocode for the same drawmethod example would look something like:

```
<SOMEXML version="1.1">
  <REQUEST>
    <GET_IMAGE>
      <PROPERTIES>
        <ENVELOPE minx="-13" miny="37" maxx="40" maxy="65" />
      </PROPERTIES>
    </GET_IMAGE>
  </REQUEST>
</SOMEXML>
```

From one perspective, these two examples produce the same result, but there is an important distinction because in the second example we have no idea what the remote system will do with the request, we only needed to understand the semantics/tags for XML requests and responses. This small difference in programming technique makes a huge difference for interoperability of systems in Service Oriented Architectures. In the new SOAP world systems are more loosely coupled and more interoperable. There are many good articles on SOAP available on the Internet, please use the references provided at the end of this paper for further information.

WEB APPLICATIONS DEFINITION

With a basic understanding that Web Services allow programmers to integrate systems using XML and SOAP, we can move on to define Web Applications. Simply put, web services are web-enabled shared libraries that programmers can access. Web applications, on the other hand, are web-enabled executables or programs. Most of the time users will think of these applications as a web page or a URL because that’s what they will interact with, but at an architectural level there are significant differences.

For example, an application could be built that accesses one or more web services, but a user cannot interact directly with a web service. Users can only interact with web applications, while programmers/programs can interact with web services.

Building on the previous example for drawing a simple map, only an application would directly make a request of a map image service. For instance, a user could click on a map and the application could prepare a request for a new map image that zooms in/centers on the new location where the user clicked. That request would be sent to a map service, and the response from the service would be interpreted by the application. Next the application would present the map to the user.

Beyond that simple example, however, there are many applications that can access web services. A significant part of the confusion surrounding web services is that we can only really talk about how to use the technology in the context of an application; many times people mix the ideas of web services and web applications into the same discussion. It is important to separate the two, however, because they are as different as executables and programming libraries.

TYPICAL GIS APPLICATIONS AND ARCHITECTURES

In the last 20 years a number of different approaches have been taken to leverage geospatial information. Many organizations continue to focus on mapping and related record keeping, but many organizations have made the use of geographic information more central to their daily business operations. Figure 1. presents a typical mapping-oriented IT architecture.

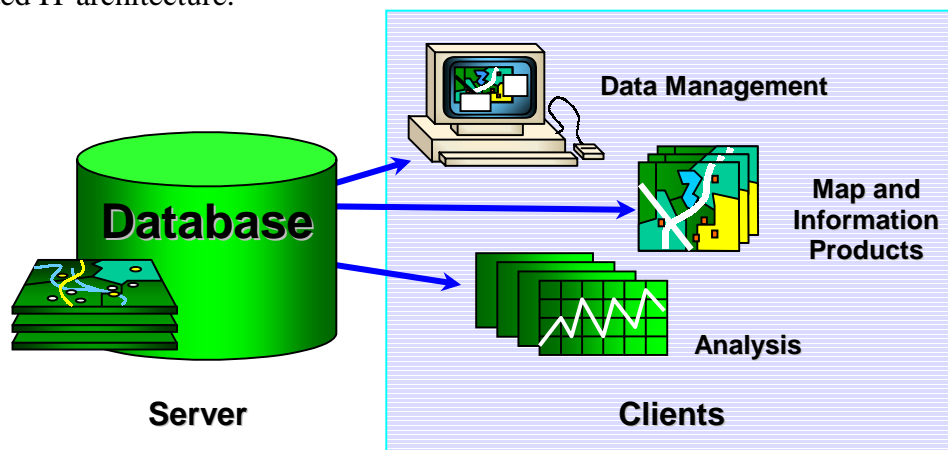


Figure 1. Traditional Client-Server system with Desktop Client Applications and Data Server

In this type of environment, thick application clients are used to interact with the database server for data management and mapping. Typically, applications for analysis also exist for functions such as utility capacity planning, transmission corridor routing, and environmental planning.

HIGH-END DESKTOP APPLICATIONS VS. WEB APPLICATIONS

In this traditional architecture, two important technology factors make it unlikely that desktop GIS will be replaced with Web Applications. First, many large sites have already utilized Citrix and Windows Terminal Server emulation for the desktop application as a way to centralize systems management and reduce network traffic. In this context, replacement web applications that interact with web services would only serve to increase network traffic over a Citrix-style solution. In a Citrix/WTS environment, end-user mouse events are transmitted to an application server, and only a small amount of screen update data is passed back to the client. This environment reduces network traffic to the point that a 56k connection can be used to run a sophisticated network editing application. Some Local Area Network implementations will be able to choose this option, but it will not provide significant performance or administration advantages.

At the server/services tier, the shift to web-based applications would likely increase the number of application servers for graphic-intensive applications such as as-built posting and engineering design. For instance, in the Citrix example, a typical environment would consist of a thin client, an application server, and a database server. In a web application, we would add an additional software layer on the application server to receive and route requests to appropriate systems. It is likely that this would increase hardware requirements compared to a Citrix/WTS environment.

The second reason that Web applications are not a complete replacement for desktop GIS is functionality. Web-based graphical applications have practical limits for application developers. For instance, while it is possible to write server-side editing applications, it is not straightforward to incorporate graphical feedback such as rubber-banding while drawing a line. Specifically the performance for this type of application will not be acceptable without significant bandwidth or developer effort. It is difficult to manage the graphical window interactions between client and server due to the fact that web services typically have stateless connections – basically a request-response communication method rather than a continuous, stateful connection. While these capabilities can be developed, the extra cost and complexity of this type of editing typically outweighs the benefits. The conclusion is that applications will be feasible for simple editing such as adding address or service points, but will not be appropriate for complex as-built posting and engineering design applications.

WEB APPLICATIONS

There are, however, other interesting uses for web applications that are more likely to be adopted for enterprise implementations. Figure 2 shows a typical web-based mapping architecture.

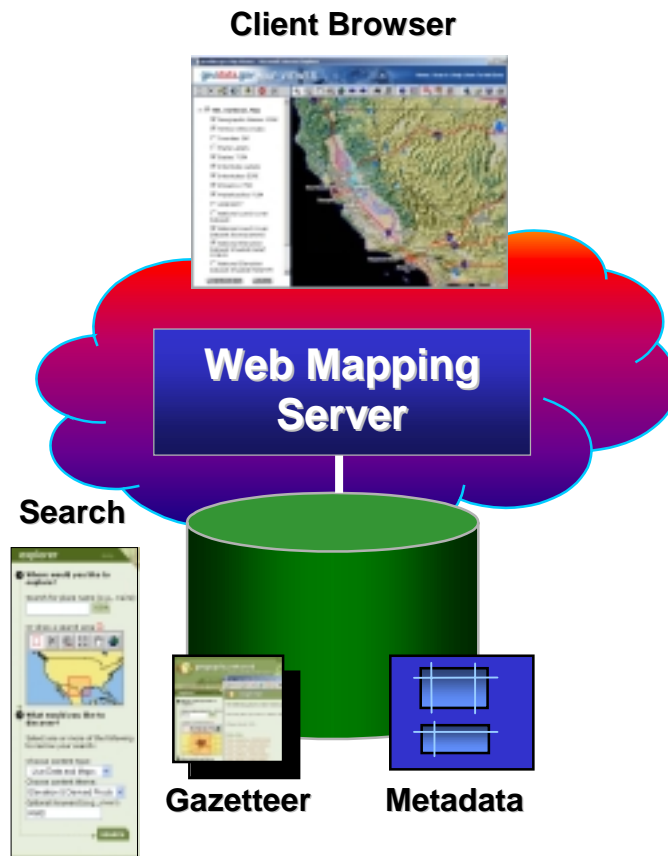


Figure 2. Typical Intranet/Internet Mapping Application and Services.

To make the data available to internal and external users, GIS-based web services have been used extensively in the last several years. In fact, these internet mapping solutions are typical of web applications that use web services. These applications send a request for a map, a service receives the request and dispatches it to an appropriate map server, and ultimately a response is sent back to a web client. One example of this is a utility Customer Service representative could type in an address or account number and a web application would present a map from a web service. Figure 3 presents the basic service architecture for modern web mapping applications.

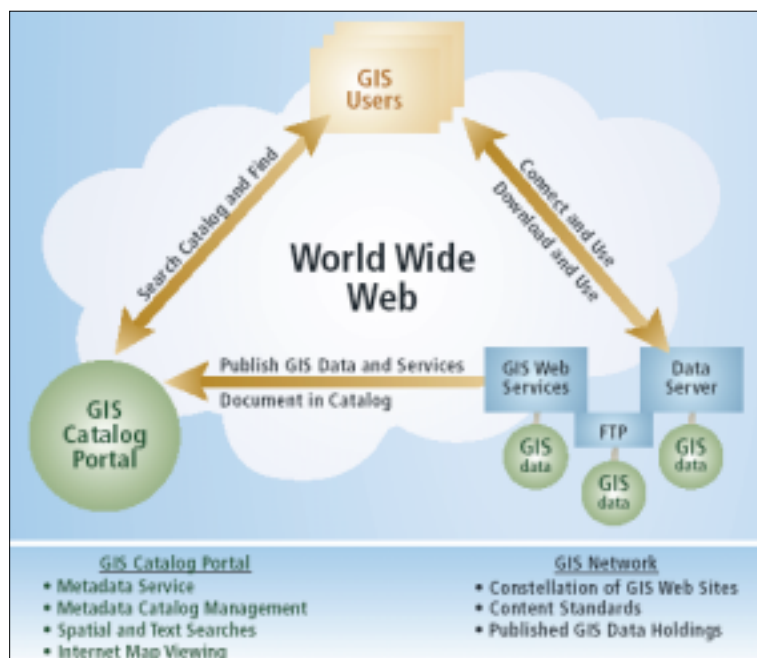


Figure 3. Basic Publish, Search, and Connect functions of Web Mapping Services.

To date, however, these solutions have been relatively difficult to customize on the server side, and the granularity of the programming interfaces provided is very coarse – basically they provide simple interfaces such as GET_IMAGE described previously in this paper. These basic mapping services are likely to remain the mainstay of web service utilization for geographic information because the typical web application has very simple requirements for user interaction. Although a move to Service Oriented Architectures will make these systems more open and interoperable, the functionality of these basic Internet mapping solutions will remain the same.



Figure 4. Typical output from a Web Mapping Service.

There are also new opportunities to build custom applications that integrate data from geographic and non-geographic data in a web browser. For instance, a gas utility may want an end user application that provides a map of streets and network data along with today's service requests. The data for the streets and network is likely to be managed in a GIS, and the data for the customer service requests is likely to be managed in an ERP or service request database. Web services make it much easier to integrate data from multiple sources and present a straightforward user interface in a web application.



Leak reported at: 1181 6TH ST E, 55106
 Time reported: 11/17/2003 5:32:54 PM
 Valve(s) to close: 1000421
 Response: Driving time: 3 minute(s)
 Crew: Davis 294
 Status: Open

Customers affected:

MR/MRS_215233390 1221 4TH ST E ST PAUL, MN 55106-5309 (651) 555-3833 AVG GAS USAGE (BTU): 85	MR/MRS_35513333 1214 4TH ST E ST PAUL, MN 55106-5355 (651) 555-3833 AVG GAS USAGE (BTU): 119
MR/MRS_97060387 1211 4TH ST E ST PAUL, MN 55106-5309 (651) 555-3833 AVG GAS USAGE (BTU): 69	MR/MRS_67914987 1273 4TH ST E ST PAUL, MN 55106-5311 (651) 555-3833 AVG GAS USAGE (BTU): 79

Figure 5. An example of an Embeddable map produced by a Web Service. Tabular information such driving directions and a list of affected customers can be retrieved from other web services and integrated in a browser application.

Modern Web Applications go beyond traditional mapping and enter the editing domain. For instance, the ability to edit feature attributes using web mapping tools has not been supported in most Internet mapping toolkits. The opportunity with web applications is to have a similar web-mapping interface, along with the ability to provide tools for attribute editing. The key part of these editing tools is that they share common business logic, custom components, and versioning models with other applications, including the desktop editing tools. Similarly, more sophisticated end user applications such as field inspections can use the same service architecture and underlying logic. As mentioned previously in this paper, these types of applications are likely to be limited to simpler user interaction paradigms. In this context, web applications will use web services to add geographic interaction to other business applications.

SYSTEM INTEGRATION USING WEB SERVICES

More sophisticated IT architectures have evolved, particularly for energy utilities, and geospatial technology is now commonly integrated with Work Management Systems (WMS), Customer Information Systems (CIS), Enterprise Resource Planning (ERP), and Outage Management Systems (OMS). In the early days, a number of point-to-point interfaces were developed between proprietary solutions, but as systems have evolved the direction has been towards Enterprise Application Integration and messaging-oriented architectures. Figure 6 shows the typical integration points between these systems.

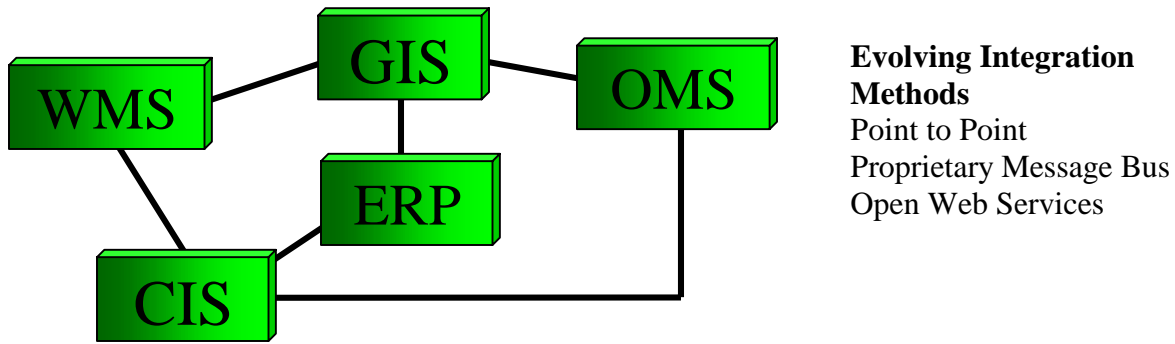


Fig 6. Typical Integrated Utility Enterprise Systems.

A number of custom-built solutions have been developed for this style of architecture, and some System Integrators have built sophisticated, configurable solutions with accompanying delivery methodologies and business models. Web services technology will simplify the integration problem by reducing proprietary interfaces for the messaging technology itself (i.e. no need to develop proprietary wrappers for each system to communicate with a message bus), and the problem will be further simplified as commercial software vendors provide XML/SOAP interfaces to their systems (i.e., no need to develop a custom SOAP interface for each system).

Overall these advancements will reduce the complexity of integration, but EAI/backbone solutions will still likely play an important role, partly because of the need to integrate with legacy solutions that do not support the new standards, but partly because they will continue to add value for managing security, performing semantic translations, and managing message routing/error handling. Figures 7 and 8 show a typical before and after diagram for how web services will change messaging-based integration solutions.

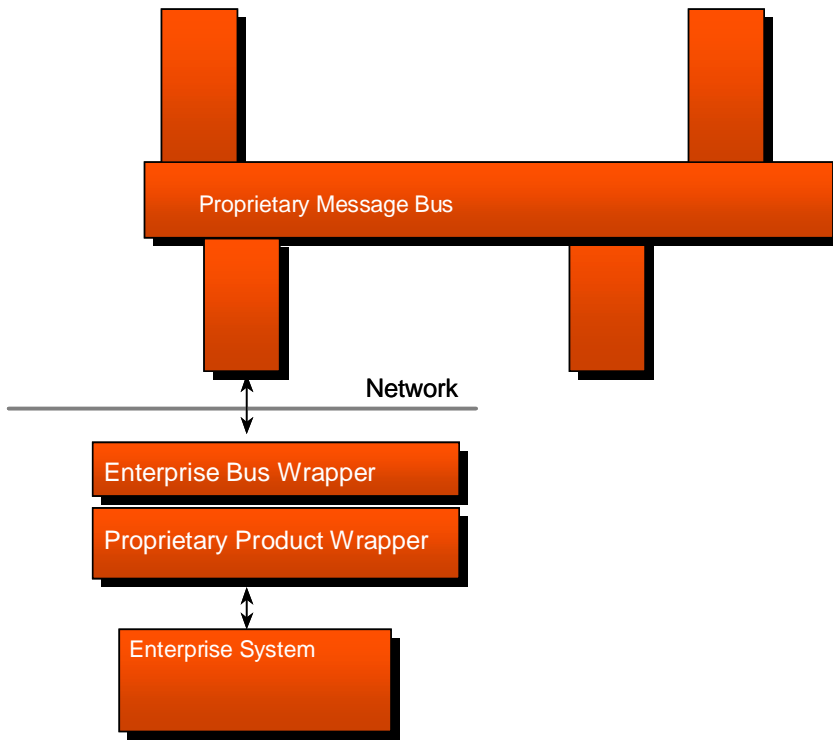


Figure 7. – Message-based Enterprise Backbone/bus Pre-Web Services.

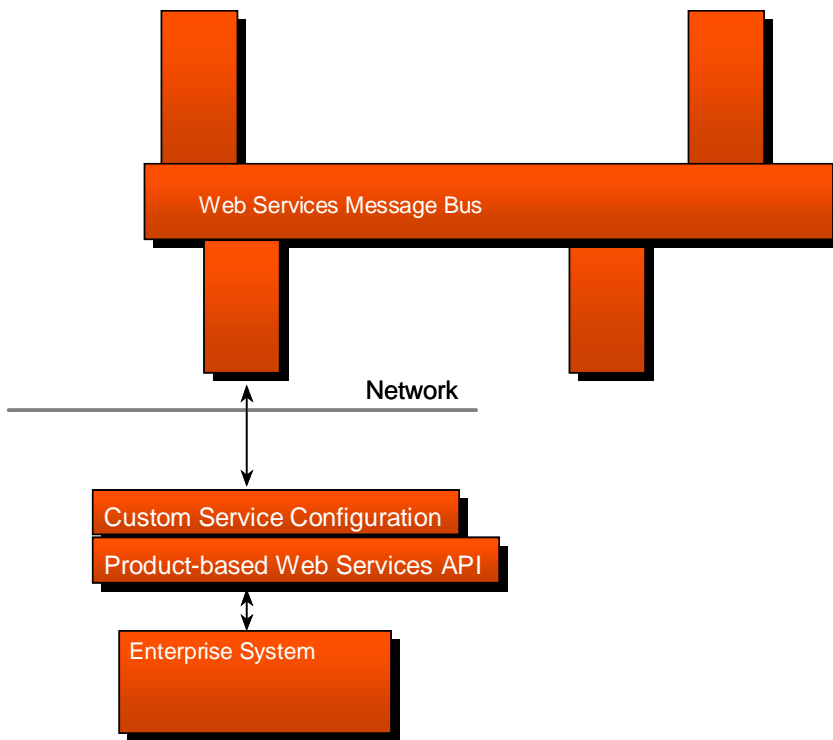


Figure 8. – Message-based Enterprise Backbone/bus With Web Services.

The key difference between Figures 7&8 is the amount of configuration and programming required to connect an enterprise system to the enterprise bus. With web services both product and message bus support for XML/SOAP should simplify systems integration. This is partly due to the basic standards that every product should support, along with the ease of common configuration/schema mapping tools for all systems. It is also due to the fact that XML, SOAP, and HTTP provide the basic infrastructure that was developed for previous proprietary backbone solutions. At this point in time, however, there are still additional needs such as security that are not completely addressed by the web services infrastructure. It is likely that enterprise backbone and messaging middleware will continue to play a major role in systems integration architectures.

SUMMARY

Web Services technology presents new opportunities for systems integration and business applications. At one level the technology is basically developer-oriented and provides a new type of loosely coupled programming interfaces based on XML, SOAP, and HTTP. At another level, Web Applications that access web services will provide many new opportunities for application development based on current GIS/IT architectures. Traditional Professional GIS editing, analysis, and mapping functions will not be significantly impacted, but many new types of applications will be possible. The implications of this new technology are that geographic information will become more embedded, more accessible, and more essential to daily business operations.

REFERENCES

Box, D 2003. *A Guide to Developing and Running Connected Systems with Indigo*
<http://msdn.microsoft.com/events/pdc/default.aspx?pull=/msdnmag/issues/04/01/indigo/default.aspx>

Box, D 2003. Service-Oriented Architecture and Programming, Parts 1 and 2
<http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/20030827SOAPDB/manifest.xml>
<http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/20030902SOAPDB/manifest.xml>

ESRI , 2002. *ArcXML Programmer's Reference Guide*
http://downloads.esri.com/support/documentation/ims/ArcXML_Guide/Support_files/arcxmlguide.htm

Gott, Karl 2003. *The next stage of evolution for e-business*, IBM Web Services architecture overview
<http://www-106.ibm.com/developerworks/webservices/library/w-ovr/?dwzone=webservices>

He, Hao 2003. *What is Service-Oriented Architecture?*
<http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>

MSDN SOAP Site
<http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28000523>

Programming Articles on webservices.xml.com
<http://webservices.xml.com/programming/>

RelaxNG Web Site
<http://www.relaxng.org>

World Wide Web Consortium
<http://www.w3c.org>