Pitney Bowes MapInfo

From Mobile to SOA:

A Guide for Optimized Application Deployment

WHITE PAPER

Tom Myers • Senior Product Manager, Pitney Bowes MapInfo

From Mobile to SOA:

A Guide for Optimized Application Deployment

ABSTRACT

CUSTOMER NEED HAS BEEN AN IMPORTANT FACTOR IN THE EMERGENCE OF MULTIPLE COMPUTING TIERS. TODAY'S APPLICATION DEVELOPERS AND IT ADMINISTRATORS HAVE THE ABILITY TO DEVELOP AND DEPLOY THEIR SOLUTIONS TO MOBILE, DESKTOP, MIDDLE, SERVICES ORIENTED ARCHITECTURE (SOA), AND DATABASE TIERS. THESE OPTIONS PRESENT MANY OPPORTUNITIES: COMPUTING UBIQUITY; SHORTER DEVELOPMENT CYCLES; HARDWARE AND SOFTWARE MAXIMIZATION; AND, OVERALL, THE ABILITY FOR ORGANIZATIONS TO BETTER LEVERAGE EXISTING PEOPLE AND INFRASTRUCTURE INVESTMENTS. BUT ALONG WITH THE OPPORTUNITIES HAVE COME CHALLENGES, THE MOST SIGNIFICANT OF WHICH IS THE QUESTION OF "WHERE?" IN OTHER WORDS, GIVEN A CUSTOMER'S REQUIREMENTS, WHERE SHOULD THE PIECES OF A SOLUTION BE DEVELOPED AND DEPLOYED?

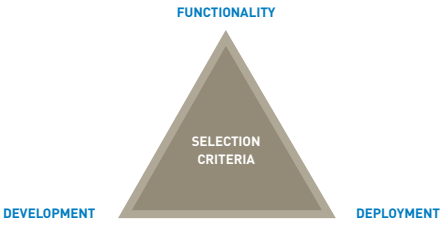
MANY CONSIDERATIONS ARE INVOLVED IN SELECTING APPLICATION SOFTWARE OR APPLICATION DEVELOPMENT TOOLS.

These same opportunities and challenges exist when it comes to geographic information systems (GIS)—applications that bring location intelligence to an organization. Because these applications often rely on extensive data stores, have complex computing "ecosystems," and require intelligent design in meeting user performance and usability expectations, the challenges can be even more significant. This paper is a guide to help you answer the question of "where?"

Customer Requirements

Many considerations are involved in selecting application software or application development tools. Three significant considerations are how the product can be developed (e.g., access to capabilities or the object model); how the end product, once built, can be deployed and maintained; and what functionality is present and can be utilized.

Diagram 1



Because they are critical to the success of any project, these factors are sometimes called "selection criteria." For example, when selecting a toolkit, the application developer will almost always ask three types of questions.

1. Is this development tool going to have the **functionality** I need to solve my business problem? In the case of location intelligence, what are the geospatial capabilities?

2. How am I going to access this product, and how am I going to **develop** against it? For example, in what language or environment do I need to program?

3. How am I going to **deploy** and maintain this solution once it is built? Can I leverage my existing investment in people skill and IT infrastructure?

To meet these requirements, the computing industry has evolved several tiers for computing, each with its own characteristics, strengths, and weaknesses.

Tier Strengths

Before we examine the benefits of these several tiers, let's back up and pose a question: why do so many options exist in the first place? Outside of market forces, market positioning, and competition, the easy answer is that no single approach or tier is able to solve the challenges for all business problems. If your mobile phone could execute 1,000 Web requests per minute, why would you need a mobile solution and a Web Services or SOA solution? If your data dissemination needs could be fulfilled without centralized processing and storage, why would you need both a database solution and a desktop solution? The fact is that, along with the strengths, these tiers also have limitations; hence, the evolution of options.

The chart on the next page highlights some strengths associated with each tier. As some of these are readily apparent, we will not spend time on all of them here. However, we will dive a bit deeper into some others. It is important to note that the strengths found in each tier are just that. You could likely accomplish all of the below tasks in any given tier, but some tasks are clearly more suited to some tiers rather than others. In some cases, it may not even make sense to accomplish a given task in a certain tier due to it being cost-prohibitive, unreliable, etc.

DESKTOP/MOBILE TIER	MIDDLE/WEB TIER	SOA TIER	DATABASE TIER
Standard Strengths	**Standard Strengths**	**Standard Strengths**	**Standard Strengths**
Object Oriented	Object Oriented	Request/Response Oriented	SQL
Applications	Applications	Web Services	Enterprise Data Depot
Stateful	Stateful	Stateless	Stateless
Supports needs of one user	Supports needs of many application users	Supports needs of many software agents	Supports needs of many software agents
Orchestration tier	Orchestration tier	Platform tier	Platform tier
Portable/Embeddable	Portable/Embeddable	Centralization	Centralization
Display device output	Browser output	Browser output	Limited output
Data Read/Write	Data Read	Data Read	Data Write
Miscellaneous Strengths	**Miscellaneous Strengths**	**Miscellaneous Strengths**	**Miscellaneous Strengths**
Application/business logic tier	Application/business logic tier	Simplified APIs accessible to any application or business process	Enterprise Data Dissemination
May or may not be portable device	Business logic can access data locally	Proliferated XML	Data management

OO vs. RR vs. SQL

Objected Oriented (OO) programming encompasses many popular programming languages such as Java, C# and VB.NET in which a developer can solve a business problem by building a data model consisting of objects, classes, methods, and variables. These third-generation languages are powerful in that, through proper design, they can solve large problems with less code, particularly when compared with procedural languages of the past. For this reason, they are the language of choice for the building of business logic—the intelligence behind all application software and solutions.

A Request/Response (RR) environment such as Web Services is very different. A Web service is an interface that provides programmatic access to capabilities via well-defined and open interfaces. Web services include standards such as XML to assist with the exchange of commands and data. Although it is infinitely more procedural than OO, the openness and proliferation of XML means that completely foreign agents (e.g., any programming language, application software, or device) can leverage a Web Service or be built for this task. This idea is relatively new in computing circles and is referred to as services oriented architecture (SOA). At its heart, SOA promises to connect disparate systems, software, data, and functions.

Regardless of your language of choice, at some stage your enterprise solution will access a database and hence require the writing of SQL. This has essentially made SQL a ubiquitous language.

AT ITS HEART, SOA PROMISES TO CONNECT DISPARATE SYSTEMS, SOFTWARE, DATA, AND FUNCTIONS.

Stateful vs. Stateless

Stateful servers keep track of events and data for individual users within a multi-user environment. These servers are often more efficient than stateless servers because duplicate events and data within a user session do not need to be recomputed with each request. To help us illustrate this, let's consider a common spatial operation such as the building of a thematic map, for example, a color coded map of demographics by ZIP code (postal code) for the entire United States. User A asks for a ZIP code map based on median household income. User B asks for the same map but based on population. Both users want to zoom in on that map to get a closer look.

- We need to make sure that User A gets the income map and not the population map.
- We don't want to recalculate the theme for all 43,000 ZIP codes; instead, we want to re-use that work and simply zoom in on the map.

By maintaining the original theme computation in memory and keeping track of both User A and User B, a single server side application has to do less work and can therefore complete the task more quickly.

Stateless servers, on the other hand, dispose of events and data associated with particular users in a multi-user or multi-application environment. Stateless servers are able to treat each request generically and are often more scalable than stateful servers because there is no overhead associated with user session management. Stateless servers throw away all the hard work between requests, but when properly utilized that work is either trivial or often not needed for re-use. A common spatial operation such as a geocode (determining a latitude and longitude for a street address) is an ideal stateless Web service because the processing required is minimal and there is little value in holding on to geocodes between requests.

The Support of Agents vs. Users

Much of the excitement centered on SOA comes from this new, service-focused paradigm. And anything that can read and write XML—any programming language, application software or device—can be serviced using SOA. Web services provide the application developer with tools to build an infrastructure to serve multiple applications or agents. While this may sound expensive, it doesn't need to be. Web services enable an organization to consolidate data and functionality in a single place and provide access to many different departments, partners, and customers. No wonder then that Web services are viewed as an answer to closed, under-utilized legacy systems.

The benefits of this are even more apparent on the receiving end. The software applications/agents that access these services do not require a deep understanding of the technology or, for that matter, an investment in the maintenance of that technology. They just need to access it. One good example in GIS can be found through data Web services such as satellite imagery. An application developer wishing to add a satellite image to a map could go out and acquire the data, store it, maintain and integrate it into their application. Or they could leave much of that up to a service provider and simply access the data through a series of Web requests.

In the SOA model, the support of agents and applications is the primary focus. Again, it is services oriented. At the end of the day, however a user needs a graphical interface, local data to access and interrogate, and business logic to aid in their decision-making. Although SOAs can help support the overall solution, they are not built to meet these requirements. This is largely the job of the OO application developer, whose solution will be deployed in the middle, desktop, or mobile tier. In this tier, we find powerful tools for the development of business logic and user interfaces,

support for the access of local data sources, and the management of state for the efficient execution of complex operations. In a nutshell, the goal of SOA is to serve agents and applications, which are often written in OO or a similar language, and ultimately serve end users.

Orchestration Tier vs. Platform Tier

Inherent in much of what we've discussed above is the fact that SOA and database tiers are the platform for many disparate systems, while applications in the middle and desktop/mobile tiers need to be portable, often serving as building blocks embedded in a larger solution. The platform tiers do not need to be mobile and do not need to be subsumed by larger solutions—but they can be. Most IT groups will play to the strengths of these "platform" tiers: their ability to be centralized and their ability to serve many agents, largely foreign, throughout an organization.

The business logic that gets built as the application is that much closer to the user. It is that last mile that some have labeled the "orchestration" tiers. Because users are unique and never stay in one place, development in these tiers require a lot of choreography. Applications are user-centric and hence they rely on a variety of capabilities and data that can be brought together in these tiers. Because there are so many users with so many distinct needs in so many different places, solutions built in these tiers must be portable.

Another example from the GIS world helps bring this point home. Asset tracking is an area that has received a lot of recent attention. Such systems are often comprised of three major components: a device that can be attached to a person, car, truck, train, or object; a network for transmitting the location of the device; and the software for visualizing and then acting on this information. Users of this technology will vary, but they are often decision-makers tasked with making sure people and goods reach their destination on time and within cost. A decision support system needs to be available in many places (e.g., desktop, mobile, via a Web browser), needs to present real-time data, and absolutely needs to be able to compare this data to a backdrop of customer, infrastructure, and geographic data. Finally, the intelligence behind the system involves software for scheduling, reporting, charting, geographic analysis, and display. All of this can best be composed in the "orchestration" tiers, perhaps relying on services but certainly relying on a database for management of locations.

Reading and Writing

As these tiers have evolved, all have proven to excel at the execution of commands. The same cannot be said when it comes to the reading and writing of data. Within the database tier there exists a combined centuries of knowledge regarding the writing of data. Database systems have sophisticated operations for transaction control, user access control, and versioning to ensure the integrity of a data store. These systems can handle long transactions and are particularly good at helping users manage volumes of data. Outside of this tier we find a similar if less sophisticated capability. In the desktop tier, these same operations are available on "less intelligent" data sources such as system files. In the middle and SOA tiers, we have similar logic but the additional challenge of multi-user access. Suffice to say, reading and writing is available in all four tiers; however, the support of these operations varies greatly. In the GIS world, the support of multi-user writes often requires a spatial database.

IN THE GIS WORLD, THE SUPPORT OF MULTI-USER WRITES OFTEN REQUIRES A SPATIAL DATABASE.

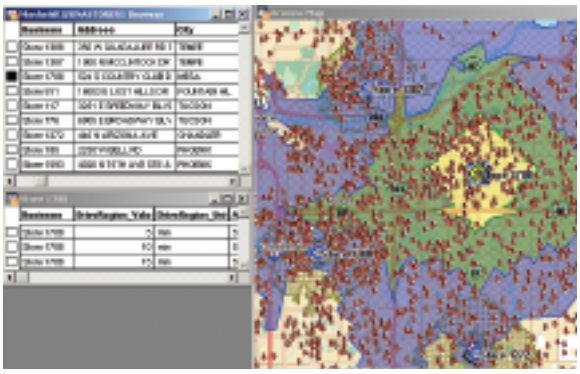
The Emergence of the Browser Tier

With sophisticated Web browser-based technology and approaches, it is now quite common to have processing in what was once referred to as thin clients (our Web browsers). Technologies such as Flash, Silverlight (formerly Microsoft Windows Presentation Foundation Everywhere), and JavaScript—combined with innovative approaches such as AJAX and mash-ups—have created a new tier of computing. With the advent of the so-called Web 2.0, users now expect a full desktop experience from their Web browser. But getting there requires processing on the client. As an extension to the above discussion, the browser tier has many strengths including easy to use and scripting languages, simple user interfaces, statefullness, and browser-based output. When considering the diagram above, one might visualize a third dimension where the browser tier resides above all others and has direct access to them. Mash-ups in particular take this approach. Whether the data or service resides on the local desktop, in the middle tier, or a Web service, mash-ups strive to unite all of these elements. Suffice to say, this is a tier under recent and rapid development.

Use Scenarios

When it comes to GIS and location intelligence, discussions of tier choices will always return to Diagram 1 above. Will the choice have the functionality required and leverage existing investments in people and infrastructure? As we also outlined above, certain operations are well-suited to a particular development or deployment type (e.g., you might be able to squeeze your servlet container onto your PDA, but is it really suited for that?).

One recent success story helps illustrate this point. The deployment included the need for both the centralization of core capabilities (SOA) and the distribution of applications that consume these capabilities (middle tier). The customer built a Web application that compared, analyzed, and displayed sales territories relative to customer locations. The application called out to a central server for geocodes and a street network backdrop. By using the central server, they made data and services maintenance much easier.



By exposing the services through open and easy-to-use Web services, these core capabilities could be utilized by any agent or application in an organization and not just the first application that was deployed. For example, that central data was useful to other departments such as marketing and production. Building the business logic in the middle tier also made sense because there was clearly a need for sophisticated business logic to compose both disparate and local data and capabilities. Other strong suits in this tier include session-state management and portability.

Here are a handful of other examples based on common customer needs.

From Mobile to SOA:

A Guide for Optimized Application Deployment

EMERGENCY RESPONSE SYSTEM (MOBILE TIER)

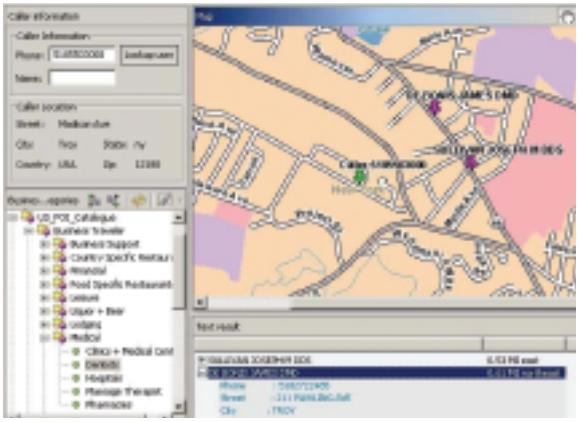
A customer wanted to provide police officers with a mapping and spatial analysis application that would help them manage and respond to incidents. The application needed to be deployed in squad cars on disconnected mobile devices and laptop computers running both Linux and Windows.

Key requirements:

- The application was intended for a single purpose and single user
- The customer was skilled in Java programming
- Target machines would be disconnected
- Target machines would be multi-platform and have minimal system resources
- The application needed to bring together a variety of local data sets and technology
- The application had to both read and write data

Given all of these requirements, the natural choice was to build the solution based on a Java SDK and desktop deployment. This application is successfully running and has helped the police department respond quicker to incidents and better plan where to place assets based on historical data.

FIELD FORCE MANAGEMENT (MIDDLE/WEB TIER)



In this case, a customer needed a tool to help manage a large team of field technicians. At corporate headquarters, a team of customer service representatives needed a view on where technicians were, where they planned on going, and which of those technicians were available when unplanned calls came in.

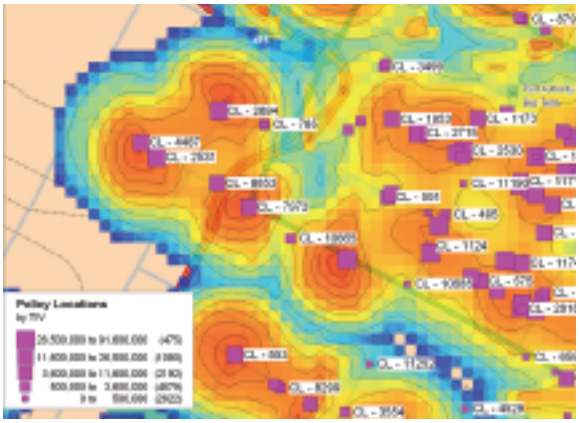
Key requirements:

- The application needed to service multiple users over an intranet (Web browser required)
- The application was intended for a single department
- The application needed to bring together a variety of local data sets
- For the sake of efficiency, complex spatial operations (e.g., searches within regions and shortest path street network analysis) required that state be maintained between user requests

The customer had a significant investment in .NET, both as it relates to IT infrastructure and people skillset, so a middle tier .NET Web deployment was the right choice for their solution. Now, customer service representatives have the information they need at their fingertips and their customers are receiving significantly better service.

CUSTOMER SERVICE REPRESENTATIVES HAVE THE INFORMATION THEY NEED AT THEIR FINGERTIPS.

RISK ANALYSIS (SOA TIER)



An insurance company needed to provide their underwriters with information on natural and man-made hazards, along with information on how much insured properties in a given location might be at risk from such hazards. The customer needed to distribute the information to a number of applications throughout their enterprise, some as simple as word-processing applications and others that were more complex decision support systems.

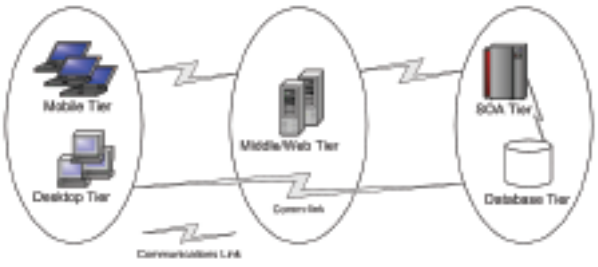
Key requirements:

- The system needed to perform a number of operations such as mapping, spatial analysis, geocoding, and routing
- These operations would be accessed by multiple departments and a variety of applications
- Application developers would need open programming interfaces as their solutions would be written in many different languages on many different systems
- Core capabilities needed to be centralized so as not to duplicate efforts in multiple departments
- The system needed to be highly scalable due to the sheer number of operations

All of this—combined with an existing investment in Web services—made an SOA deployment the right choice for this customer. As a result, they are now quoting more accurate policies based on risk assessment, thus lowering their exposure to a catastrophic event.

Bringing it All Together

When enterprise developers do their work, they rely on multiple tiers both to simplify design and to facilitate division of labor. The designs that work best are the ones that encapsulate the details of the underlying tier's implementation and simply answer the needs of the calling tier. One simple example lies in the details of a Web service that returns driving directions. Such a service can expose the essentials (e.g., a GetDirections Web request that requires start and destination addresses) but it will hide the network analysis algorithm run on the street network data that produces the driving directions. Although simple applications may not require computing in multiple tiers, it is hard to find an application these days that works in isolation within a single tier. The fact is, no one tier can solve the need of developing truly scalable enterprise-wide applications.



Pitney Bowes MapInfo Solutions

For many organizations, desktop, middle tier, and SOA tier deployments are an absolute necessity, as are .NET- and Java-compliant development kits. For these reasons, vendors will continue to offer products with multiple development and deployment options.

Pitney Bowes MapInfo has what it takes to meet these requirements. In addition to what is listed below, we offer support for both the mobile and spatial database tiers via MapX Mobile and SpatialWare, respectively.

We have MapXtreme 2005, a single .NET SDK for both desktop and Web-based development and deployment. The MapXtreme 2005 object model is 100% .NET, and as a result we are able to provide a very familiar and highly usable product through tight integration with Visual Studio .NET.

We also have MapXtreme Java, a 100% Java engine that can be used for both desktop and Web deployments. Being Java, it is highly embeddable and portable to virtually any device.

Finally, we have our Envinsa platform, built exclusively for those looking to leverage a Services Oriented Architecture and take the latest approach to enterprise computing.

A decade after the launch of MapInfo's first software development kit, we continue to meet the demands of application developers with location-enabled toolkits and applications.

ABOUT THE AUTHOR: MR. MYERS IS A SENIOR PRODUCT MANAGER AT PITNEY BOWES MAPINFO. HE HAS RESPONSIBILITY FOR PITNEY BOWES MAPINFO'S DEVELOPMENT TOOLS INCLUDING MAPXTREME AND ENVINSA.
MR. MYERS HAS BEEN WITH PITNEY BOWES MAPINFO FOR OVER NINE YEARS. IN PREVIOUS ROLES HE HAS SERVED AS BOTH TECHNICAL MANAGER AND LEAD DEVELOPER OF INTERNET SERVICES BASED ON THE PRODUCTS HE NOW MANAGES. PRIOR TO JOINING PITNEY BOWES MAPINFO, HE RAN A SUCCESSFUL IT SOLUTIONS FIRM FOR SEVERAL YEARS AND HELD POSITIONS AT THE UNIVERSITY OF WASHINGTON, UNIVERSITY AT ALBANY AND ROCKEFELLER COLLEGE.
MR. MYERS STARTED HIS CAREER AS A NASA FUNDED ANALYST WITH THE CONSORTIUM FOR INTERNATIONAL EARTH SCIENCE INFORMATION NETWORK (CIESIN) WHERE HE HELPED PIONEER THE INTEGRATION OF GEOGRAPHIC INFORMATION SYSTEMS WITH WEB BASED TECHNOLOGY. HE HAS GRADUATE DEGREES IN LAND USE PLANNING (MRP) AND GEOGRAPHY (MA) AND HAS OVER 17 YEARS OF INDUSTRY EXPERIENCE.

Pitney Bowes MapInfo

UNITED STATES

One Global View
Troy, NY 12180-8399

main: 518.285.6000
1.800.327.8627
fax: 518.285.6070

sales@mapinfo.com
www.mapinfo.com

CANADA

26 Wellington Street East
Suite 500
Toronto, Ontario
M5E 1S2

main: 416.594.5200
fax: 416.594.5201

canada.sales@mapinfo.com
www.mapinfo.ca

EUROPE/UNITED KINGDOM

Minton Place
Victoria Street, Windsor
Berkshire SL4 1EG UK

main: +44 (0)1753 848200
fax: +44 (0)1753 621140

europe@mapinfo.com
www.mapinfo.co.uk

ASIA-PACIFIC/AUSTRALIA

Level 7
Elizabeth Plaza
North Sydney
NSW 2060

main: 61.2.9437.6255
fax: 61.2.9439.1773

australia@mapinfo.com
www.mapinfo.com.au

92033-802