Whitepaper

Tom Myers
Senior Product Manager
MapInfo Corporation

The .NET Advantage –

Developing in an Open Environment

for Maximum Success

Abstract: The selection of an application development tool for the creation of spatially enabled applications usually involves many considerations. Price aside, three common and significant factors include: how the toolkit can be developed (e.g., access to the object model and development environment); how the end product, once built, can be deployed and maintained; and, in the creation of spatially enabled applications, what spatial and non spatial functionality can be utilized. With the release of the Microsoft .NET Framework, spatial application developers have significantly more options in these areas, and in particular in the development and deployment of their solutions. The following paper will highlight how MapInfo Corporation, over the course of a two year effort, utilized .NET technology in the creation of the next generation of Windows-based spatial application development tools – MapXtreme 2004. The latest release, MapXtreme 2005, is now available.
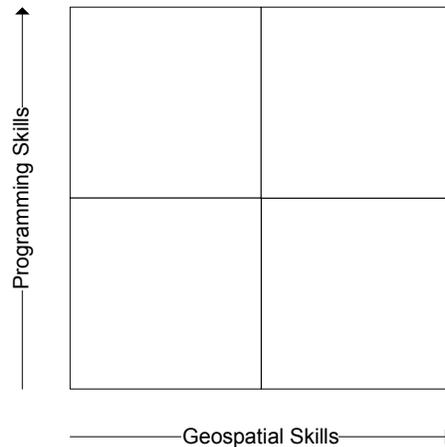
Whitepaper

INTRODUCTION

The geospatial community includes a wide range of application developers. One can plot these developers on a matrix where the X axis would range from the geospatial novice to geospatial expert while the Y axis would range from the programming novice to programming expert.

Figure 1

This diversity drives the need for the creation of development tools that are both easy to use and open. With the release of the .NET Framework by Microsoft Corporation, independent software vendors (ISVs) have the ability to cater to a broader set of application developers. This is largely due to the fact that .NET provides developers with many more options in the development and deployment of their applications.

A BRIEF INTRODUCTION TO THE .NET FRAMEWORK

Microsoft's .NET Framework is the programming model underlying .NET for developing and deploying applications. The framework provides a number of new techniques for the application developer. Three of significance are: the support of Web services; a common language runtime; and, the isolation of application assemblies which translates to improved portability.

Web services provide functionality to the application developer and to the solutions he or she builds. This functionality is accessible via standard protocols such as XML and SOAP and are in many respects vendor neutral. The services themselves and the functionality that is provided (e.g., a service that provides maps, demographics, or driving directions) may reside in many different places. The service may be fee based and available on the Internet or deployed locally within an enterprise LAN. The

key is that, regardless of where the developer plans to develop and deploy their application, Web Service should relieve them of certain responsibilities such as being an expert in the technology behind the service and administering and providing the technology to users.

The Microsoft common language runtime manages several important features within the .NET Framework including language integration. Language integration provides development teams with an extremely versatile environment. Developers utilizing .NET can program in many different languages including Visual Basic, C# or J#. As important is the fact that code written in any of these languages can be merged into a single project thus allowing developers with different backgrounds and expertise to work on the same project team.

With the introduction of .NET, the business of deploying and maintaining applications has become much easier. With a hard separation between the business logic of an application and the presentation of that logic, the deployment and utilization of both local and disparate systems is more seamless than ever before. Thin and thick clients, local and remote processing can all be built on the same underlying code thus removing the need to re-write an application for a particular deployment type. In addition, with .NET came the introduction of assemblies which, in essence, are the next generation of dynamic link libraries (DLLs). In a nutshell, assemblies make the development of portable applications possible.

PROJECT HISTORY AND ACCOMPLISHMENTS

At MapInfo Corporation we cater to a wide variety of developers with a diversity of skills and experience. Our portfolio of spatial development tools have met the needs of Web and desktop developers; the Java and Windows communities; enterprise development teams as well as 'one man shops'. The introduction of the .NET Framework not only signaled a shift in the IT industry, it provided an opportunity for many ISVs including MapInfo. Over a two year effort, MapInfo moved our Windows-based development tools into the .NET Framework. One of the reasons for doing this is so that we and our customers can leverage the openness inherent in the framework itself. It makes us all more productive and flexible.

Looking at the project from a customer's perspective, our goal was simple: provide a more compelling product. In the case of development tools this often boils down to three criteria portrayed in the following figure.

Whitepaper

Figure 2

Functionality

Selection
Criteria

Development   Deployment
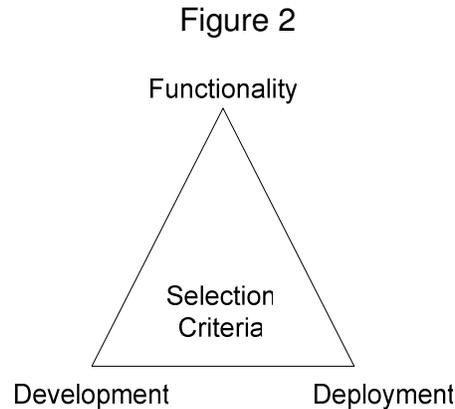
From this perspective, .NET provided us with the framework to improve our products in all of the above areas.

Development of the Technology

Over the last several years, the geospatial community has taken important steps towards more open and interoperable technology. The popularity of Web Services in general, and the Open GIS Consortium's work on service specifications such as Web Mapping Service (WMS) and Web Feature Service (WFS) in particular, have been important drivers for this project.

Web Services provide developers with the tools to both access and provide access to GIS functionality and data. A developer using MapInfo's MapXtreme 2005 location-based development environment can take a portion of their windows or web forms application and easily publish a portion as a Web service. For example, a developer wishing to share planning information within their organization can quickly create a programming interface (Web service) with a variety of spatial search and map display methods. These simple methods, when combined with any number of land-use data sources, would provide for the exchange of information between systems and people across an enterprise.

The ability to both publish and consume open standards such as WMS and WFS are similar in that they too promote the exchange of information. An organization that wanted to share, for example, real time traffic or weather maps might deploy a WMS server for this purpose. This would provide access to any WMS-compliant client regardless of the software manufacturer, and thereby relieve the provider of writing their server for any one consumer. A developer that needs to incorporate a "search within polygon" operation in their application, but lacks the polygon data itself, might utilize a WFS server for retrieval of features for their query capability. In both cases the support of open standards based on XML communication makes the distribution of functionality and the sharing of data possible.

Whitepaper

Along with the support of XML based communications, the .NET Framework opened up new development options as part of the common language runtime.

Figure 3

Graphic created by Microsoft®

The ability to choose any .NET language means a GIS analyst with some Visual Basic experience can work on the same solution as an IT specialist with a background in JAVA programming. The framework, including the Visual Studio .NET environment, appeals to a wide variety of developers and organizations that have significant investments in development teams and the skills these team possess.
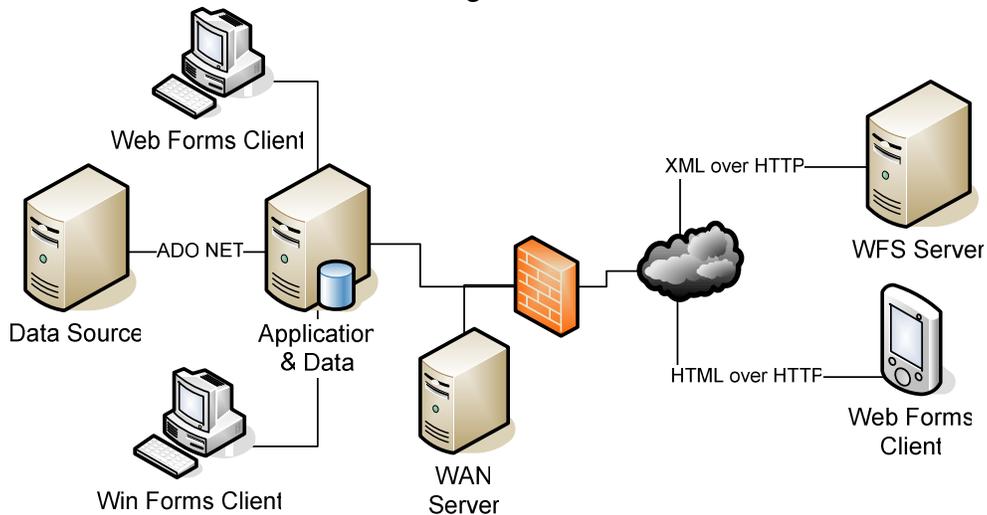
Deployment of the Technology

As depicted in Figure 3 above, the framework also provides developers with a consistent programming interface regardless of the clients they plan on building. For the MapXtreme project, this allowed us to integrate what were formally two products: MapX, which was for the deployment of desktop applications or Windows Forms-based clients; and, MapXtreme Windows which was for the deployment of Web applications or Web Forms-based clients. Looking at the project from this perspective it is easy to see a parallel philosophy between MapXtreme and the .NET Framework on which MapXtreme is built: namely, the separation of business logic from presentation. This philosophy has translated into a single development environment for both desktop and web based deployments as well as a single object model for both desktop and web-based deployments.

As highlighted above, the integration of Web Services within the .NET Framework provides additional presentation options to the application developer. Be it Web Forms, Windows Forms, or Web Services, the spatial application developer has a variety of methods for accessing and providing access to their geospatial capabilities. As significant is the fact that Web Services, in particular, provide more deployment options than ever before.

Whitepaper

One of several goals for the MapXtreme project was to fully support the next generation of Web based communication. This allows for the deployment of open systems using Internet protocols such as XML and SOAP. At the heart of this is the ability to integrate both external and internal systems and, for the application developer, the ability to leverage these systems in the creation of their solution. Take Figure 4 below, as an example.

Figure 4

In this hypothetical deployment, the application developer utilizes a number of systems in the deployment of their solution. Technology such as ADO.NET provides access to spatial attributes somewhere within their enterprise LAN while a WFS client provides access to spatial features located somewhere on the Internet. The use of other data access technologies such at ODBC add additional layers of data to the application from a local data source, in this case, a database. With a broad set of supported deployment types, all of the above data sources can be combined easily into one solution.

This same flexibility holds true for the capabilities of the application itself. A variety of both thin and thick clients can be developed and deployed as appropriate. A Web browser interface might suit certain users while a more robust Windows Forms client may be necessary for advanced users. Descending deeper into the application, the business logic can be deployed in many different ways. The WFS data source not only provides access to features it can also provide spatial capabilities to search and retrieve select features. An enterprise with a number of remote locations can force some logic to reside on the LAN and some to reside remotely on a wide area network server. As above, with a broad set of deployment types, the processing itself can be deployed where most appropriate.

Whitepaper

Finally, with the introduction of .NET came the introduction of assemblies. Assemblies can be thought of as the next generation of DLLs in that they are distinct sets of functionality. In the past, shared functionality provided in DLLs was often difficult to manage. One application might install one version of a DLL and then a second might install a different version of that same DLL on top of the first. This could lead to unexpected results. With .NET it is now possible to install, side-by-side, several versions of the same component or assembly. Applications are able to run more in isolation (e.g.., a pure .NET application does not rely on local registry settings as older DLLs had) and are therefore much more portable. A milestone for this project was to convert a significant number of un-managed DLLs to managed assemblies thus making the job of deploying geospatial application much easier.

Functionality Inherent in the Technology

If development and deployment options are the 'how' of a development tool, the spatial functionality could be thought of as the 'what' of that tool. Spatial functionality is not how the tool is developed or deployed, but what problems it can solve and what functions are provided.

For the MapXtreme project, the true .NET advantage was that MapInfo engineering teams were provided a technology that solved problems 'out-of-the-box'. Previously, it was our task to solve these problems. The support of XML communications, a common language runtime, assemblies and the like, meant that more time could be spent on developing the spatial capabilities of the product. These spatial capabilities are our core competency and the competitive advantage of MapXtreme 2005. The most significant feat accomplished was the replication of technologies we had developed for our older MapX and MapXtreme Windows products. This was a must for this project to succeed. In addition, engineering teams were able to concentrate on several areas of new or enhanced functionality including:

- a shift to a table centric data model. A fundamental difference between the previous MapX technology and MapXtreme 2005 is the separation of Tables from Layers and Datasets. A table in MapXtreme 2005 may be mappable (contain a column of type FeatureGeometry) or non-mappable thus providing standard SQL-3 access to both spatial and non spatial data.

- a totally new hierarchical geometry model. Geometry and Styles have been separated and as a result provide great flexibility to access and manipulate each component independently from the other.

- the ability to control layer display grouping and order as well as the separation of the draw order of labels and themes from base layers thus allowing greater cartographic control. Layer modifiers (ranged and individual thematic options included) have control over all layer display options including the ability to set an automatic style based on an expression or value of an attribute cell.

- the introduction of a standard cartographic map scale.

- the introduction of more extensible and more functional map tools.

CONCLUSIONS

With the release of the .NET Framework by Microsoft Corporation, MapInfo Corporation was able to build and release the next generation of spatial application development tools. Leveraging industry standard methods and technologies translated to significantly more user options in the areas of development, deployment and functionality. The goal of delivering an open development environment to the geospatial community was reached when MapXtreme 2004 was released in the spring of 2004. MapInfo continues to build off of this accomplishment including a new set of features and functionality in the just-released MapXtreme 2005.

REFERENCES

MapInfo Corporation. 2005. MapXtreme 2005 Overview.
http://extranet.mapinfo.com/products/overview.cfm?productid=1849

Microsoft Corporation. 2002. What .NET means for IT Professionals.
http://www.microsoft.com/net/business/it_pros.asp

Microsoft Corporation. .NET Framework Fundamentals.
http://msdn.microsoft.com/netframework/programming/fundamentals/default.aspx