

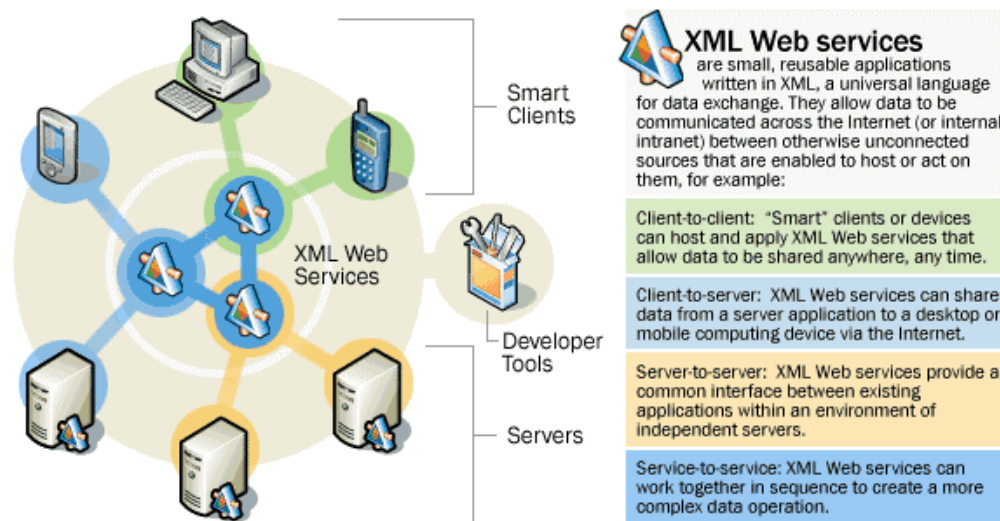
What is Microsoft .NET?

The evolution of the Internet has prompted many technology companies to provide development environments and servers for building Web services. Most vendors have based these environments on Java with the objective of creating a single language that runs in almost any environment. Technology veteran Microsoft Corporation has taken a different approach by building a development platform that supports the creation of Web services in a variety of programming languages. Microsoft's .NET Platform is a complete set of products including development tools, a Common Language Run-time (CLR), and a variety of servers (operating system - Windows, database - SQL Server, Web - Internet Information Server and integration - BizTalk).

Microsoft defines .NET in this way:

Microsoft® .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web services: small, discrete, building-block applications that connect to each other—as well as to other, larger applications—via the Internet. .NET connected software delivers what developers need to create XML Web services and stitch them together. The benefit to individuals is seamless, compelling experiences with information sharing.

The following diagram, provided by Microsoft, helps illustrate the .NET Platform. This diagram and further descriptions can be found at www.microsoft.com/net/basics/whatis.asp.



Why Microsoft .NET

Business Requirements

Epicor Software Corporation chose Microsoft .NET to build its next generation customer relationship management (CRM) applications based on Web services. But why?

This section examines the requirements of small to medium enterprises looking to integrate enterprise applications such as CRM, and how the Microsoft .NET platform fulfills those requirements.

The requirements that will be addressed are:

- Rich user experience
- Accessible
- Reliable and Scalable
- Affordable
- Connected
- Flexible

Rich User Experience

Usually discussions of user experience and ease of use are excluded when addressing architecture. Rather, these facilities are based on the design of navigation, visual cues and other techniques, which are generally independent of architecture. Yet architecture can influence user experience and ease of use in many ways. One example is that of a Web-based architecture that is designed only for a browser-based user interface. The use of a browser-based user interface places certain restrictions on the types of navigation that can be used. It also forces consideration of how standard browser navigation features (forward and back buttons) will affect the user interface and workflow.

A Web services architecture provides the powerful ability to have multiple user interfaces interact with the same Web services. Microsoft .NET extends this by allowing both rich client and browser-based interfaces to be built in the same development environment – Visual Studio .NET (VS.NET).

While browser-based user interfaces have gained significant popularity in recent years, many expect that the tools available in VS.NET to lead to a resurgence in the development of rich clients that interact with Web services over the Internet. Microsoft has begun to promote the development of rich client applications with VS.NET.

For more information on smart clients visit

www.windowsforms.com/whitepaper/whywindowsforms.aspx.

To read an independent analysis of smart clients by Gartner Group visit

www.gartner.com/reprints/microsoft/104982.html.

Accessible

Utilizing a smart client to provide rich functionality does not eliminate an organization's need for easy, flexible access to CRM information. This need is often expressed as requiring "Internet-based applications," meaning they need access options to support a variety of user types. These users may be on a local area network (LAN), or they may be part of a remote office with access to the corporate network via long distance dialup lines, frame-relay, or VPN (wide area network - WAN). There may be times when a user may be remote and need to connect to the corporate network from home or a hotel room. For remote users, the Internet is usually the easiest way to connect. In any case, providing Internet access to the application is a viable solution for providing easy access to a variety of users. One of the greatest benefits of the .NET platform is the ability to build applications based on Web services that have either a smart client or a browser based user interface, or both!

The last type of access requirement is for the disconnected mobile user. The disconnected user takes a portion of the data with them and then works while disconnected. This is particularly common among sales personnel or field service technicians. One common drawback of enterprise applications that use a browser-based interface is the simple fact that they do not function in a disconnected environment. The smart client approach provides a natural user interface for both connected and disconnected users.

Reliable and Scalable

Downtime is very expensive. All organizations want applications that work and keep working over long periods of time. Reliability and availability are directly related to the quality that a software vendor builds into its solution. However, some architectures lend themselves to increasing reliability and availability by allowing for redundancy and failover. The Microsoft .NET platform, which includes a family of servers and is based on Web services, provides a framework for an IT infrastructure that includes redundancy and failover of Web services to keep CRM applications running even when individual hardware failures occur.

It is rare to encounter a small or medium-sized enterprise that has no intention of growing. This is not surprising given that one of the primary goals of any CRM solution is to help grow business. But the last thing an organization wants to do as they grow and expand their reach is to replace a critical system such as CRM. This leads to question of scalability.

Microsoft .NET and the Web services architecture provide a simple solution to scalability issues that also addresses another important concern - a low total cost of ownership. With a Web services based architecture, instead of scaling up by buying bigger and much more expensive hardware, you can choose to scale in two ways. The first is to scale vertically, which is simply upgrading the current hardware in place by adding additional CPUs, more RAM or adding additional hard disk space. The second option is horizontal scaling, which is achieved by adding additional machines and then distributing processing requirements across all machines. The modular nature of Web services creates an environment in which organizations to quickly, dramatically, and cost-effectively scale up enterprise applications by upgrading existing hardware or adding additional machines and then distributing the application across multiple servers.

A f f o r d a b l e

There are many different variables in the total cost of ownership equation. Microsoft's .NET platform will help companies lower the total cost of ownership for enterprise software by providing several benefits that keeps costs in check.

Enterprise applications such as Epicor's Clientele CRM.NET will be able to initially deploy on a single server and then scale by adding inexpensive hardware – rather than replacing the initial server with one that is more expensive.

The use of standard Microsoft servers will make finding skilled and experienced administrators easier and less expensive. Microsoft helps by providing certification programs – so certified professionals can be easily found. Applications based on the .NET platform will use Microsoft servers such as Windows Server, Internet Information Server, and SQL Server.

The use of Microsoft's VS.NET development environment as the standard customization tool means finding developers with the appropriate skill set for customizing enterprise applications will be much easier compared to other enterprise applications using proprietary toolsets. Thanks to the Common Language Runtime (CLR), part of the .NET platform, organizations can leverage their existing investment in development skills by allowing developers to use any .NET compatible programming language. This means that a software vendor can develop their solution in one language (such as C#) and allow customers to customize the solution using another language (for example, VB.NET). In addition, vendors other than Microsoft have introduced .NET versions of their languages – making it even easier to find qualified developers.

For more information on other .NET compatible languages visit <http://msdn.microsoft.com/vstudio/partners/language/default.asp>.

F l e x i b l e

The previous section covered how customization using standard tools reduces the total cost of ownership of an enterprise application. This section focuses on other advantages of using the standard tool for Microsoft .NET development –VS.NET.

There are two important types of customization for enterprise applications such as CRM. The first is the ability to change existing capabilities – adding or subtracting fields, re-arranging fields to support workflow, or changing the business rules to ensure processes are followed. By employing VS.NET as the customization environment, application vendors can build applications that allow for the above changes without exposing the original source code or creating unnecessary difficulties with future upgrades.

All VS.NET languages are object oriented. One of the central characteristics of object-oriented programming is inheritance. Simply stated, applications are built as code objects. These code objects are called classes. Code classes are built in layers with the foundation layers referred to as base classes. Each succeeding layer can inherit the functionality of the layer below it. The advantage of this is that the customer and vendor can work on different layers simultaneously. As a software vendor, Epicor can continue to add functionality on the base classes without interfering with customizations executed on successive layers. Because .NET has a “multilingual” compiler, these simultaneous customizations need not even be in the same language.

The second type of customization occurs when the purchaser of enterprise software chooses to extend the functionality of an application. Again, the object-oriented nature of VS.NET provides advantages in this area. In particular, customers can inherit from base classes for common functionality and user interfaces. These base classes provide a dramatic head start in creating new application functionality that integrates tightly with the rest of the application being extended. New forms or Web services follow the same model and know how to interact with existing Web services in the application. This inheritance model will significantly reduce both the amount of time and resources needed to build additional functionality.

Connected

One of the most important elements of the Web services architectural model is the potential for integration between different applications. By using XML as the format for sharing data, and Simple Object Access Profile (SOAP) to communicate in a way that is programming language independent, it is possible to create Web services that talk to Web services built by other vendors, or that have been created internally.

Integration between Web services is not automatic. XML – the text-based language for exchanging data - still needs standards. Like HTML, XML uses “tags” to designate what a piece of data means. Today, organizations like WS-I (Web Services Integration) are working together so that the tag for “Customer Identifier” is used consistently across applications. Creation of standards will allow vendors to create applications that use standard XML tags – making integration even easier.

XML is only part of the integration equation. The second part is the “methods” exposed for interaction. Today there are no standards for naming these methods and what functionality should be exposed to other Web services. For instance, if two Web services are to talk to each other directly, they need to know the names of the methods available and what they do.

While integration servers are not required to link to Web services, they can help reduce the amount of work necessary to integrate applications. These integration servers, such as Microsoft’s BizTalk, allow a vendor to create a “connector” to the integration server. The connector contains the definition of the module interfaces and XML streams. Using the integration server, it is possible to map two XML streams together and create rules for what to do when the connector is activated.

Thus far discussed integration within a single business, but Web services have the potential for integrating with commercially published Web services. A technology called Universal Discovery Description and Integration (UDDI) provides a Web-based distributed directory that enables Web services to publish information on the Internet and discover each other, similar to a traditional phone book. Companies can use a commercial UDDI directory or build their own internal directory. The goal of UDDI is to allow businesses to connect to Web services provided by external business partners.

Focus on Building Application - Not Infrastructure

The final advantage of Microsoft's .NET platform to be discussed is an indirect benefit for buyers of enterprise software. It is not a standard "requirement" for architecture that most customers would name – but it does have potential for significant impact on our customers. The advantage is that the vendor building the application can focus on the application – not the platform. Many enterprise software vendors build their own application platforms. Building a platform takes many resources and must be supported by fixes, patches, and improvements over time. These resources are expensive and while they make functionality possible – rarely provide capabilities directly experienced by the end user.

Epicor Software Corporation is a leading provider of integrated enterprise and eBusiness software solutions for mid-market companies around the world. Epicor recently announced Clientele Customer Support 8.0, one of the first CRM solutions available on the Microsoft .NET platform, which will be available September 2002. With more than ten years experience and over 3000 CRM customers, Epicor provides small and mid-sized companies with everything they need for a successful CRM implementation: quality products, experienced professional services and excellent support.

The use of .NET will allow Epicor's customers to easily deploy Web-enabled applications that give businesses new levels of flexibility, control, and inter-enterprise integration. Epicor's new Web services architecture takes advantage of all the .NET Platform has to offer, including rapid deployment, ease of application integration, flexibility, customization, and extensibility through Microsoft Visual Studio. NET.

For more information about Clientele Customer Support 8.0 and the Self-Service Portal call 800-356-0912 or visit www.epicor.com.