



# [WORKING THROUGH THE BUZZWORDS]

*< Let's understand the "What" and the "Why" behind **NSK Corporate Software Development Recommendations** to see how the **NSK Development Group** accomplishes the planning and deployment of NSK business applications. >*

Let's examine some of today's corporate technology buzzwords, to discover any common principles for corporate software development. We'll provide guidelines for investing in custom development of business tools.

To do this, we'll take apart Enterprise Architecture, Enterprise Portals, Open Source and Open Architecture, Service Oriented Architecture (SOA), and Web Services. We'll clarify Web 2.0, XML, SOAP, AJAX, and SQL. We'll see if Microsoft's got anything new which may make our decisions easier. We'll unravel .NET and the "Windows vs. Web" dilemma. We'll briefly explain Apache, Perl, PHP, Python, and Ruby on Rails, and then we'll talk about what has happened to Java, J2EE, and Oracle, and what is happening to tools such as Delphi and 4D.

Lastly, we'll describe NSK's internal goals for long term software investment, and we'll explain why our total cost of maintenance on a software application is low, only when our client's total cost of ownership is low. We'll show you how business buzzwords happen.

**Keith Mitchell**

[Senior Developer, NSK Inc.]

March 18, 2008

## [WORKING THROUGH THE BUZZWORDS]

*< Scalable software doesn't blink when you add 10 users, doesn't slow when you add 200,000 records, and won't crash when you request from NSK 20 more features than it is currently doing. >*

### OF ENTERPRISE AND ARCHITECTURE

Sometimes, “enterprise” software is simply software that has enough features for a large business. Microsoft SQL Server Enterprise is feature-filled. An average developer, faced with the challenge of writing a complex application, may believe “enterprise” means complex. But to a good developer, “enterprise” simply means “scalable”. Software which scales, expands as it is required to do so. Enterprise means that when you program good user security on a database in accounting, and then put a similar database in another department, the two will be able to talk to one another, while still using that good security (instead of, in spite of that security). Enterprise means that “3/14/2008” means “March 14” in New York, yet won't crash a server in London because there's no third day of the 14<sup>th</sup> month. Truly Enterprise software requires very high quality architecture—it needs a careful design, which is documented, and which has been well considered. Software which is architected, or designed, to be scalable, has three very important features.

Scalable software can talk to other software easily. Scalable software can be expanded easily. And scalable software is faster than it needs to be, or as fast as is presently possible with technology. Almost all the big buzzwords in business technology have to do with scalability.

#### “Open Services”

— The only new idea.

Gone are the days that the DEC computer in the sales department won't talk to the IBM in finance. Computers can talk to each other. But pulling an RSS feed into a system on a different server can still take time (and cost money). What if applications talked to each other so easily, that you could expand your software by plugging on new pieces? Modern software design takes the first two foundations of scalability (open communication, robust expansion) and combines them. The revolution in *Enterprise Software* has come from expanding complex systems, by developing task-specific pieces, which communicate easily. To grow, you develop applications to perform a particular service, or you buy them, and then someone plugs them in and they work.

Scalable software can talk to other software. Scalable software can expand. And scalable software is faster than it needs to be. Almost all the big buzzwords in business technology have to do with scalability.

## [WORKING THROUGH THE BUZZWORDS]

*< Service Oriented Architecture is a design of business systems where applications which store, manipulate, or use data, provide a mechanism or service to other applications in the system, for getting at that same data. An open architecture is a business system where applications “expose” what they do, and the data they use, to other applications or network services. >*

### OPEN ARCHITECTURES AND SERVICES

Web Services started a lot of things (moderately well).

A web service is a web page that another computer can go to, to get information. It's that simple. By the time web services were popular in certain functions (weather info, product vending by middleware and B2B, and stock quotes), they had been very standardized and defined. That standard was clumsy to implement and was sometimes too slow for the speed of business. Much work went into creating lightweight wrappers for the data which was given out by web services, so that they would be fast. SOAP is a very lightweight data container for web services data. What's good about SOAP and web services? Data wrapped in a SOAP wrapper can move like a text file across networks, through firewalls, from a Windows machine into a Linux machine into a Mac, and then into a library mainframe. Data can be easily taken from one database and added into another, or made into a report. This was revolutionary several years ago. Web services tell other computers what they do (what “service” they provide) and what data to expect. However, the expected explosion of computer-consumed data didn't happen with web services, except in some e-commerce and distributed corporate environments.

Why not? New and better ways are always being created for software systems to become more open. This is a very important benefit to corporate software customers because new functionality can be added inexpensively. Have you noticed how quickly RSS newsfeeds became widespread? Compared to SOAP web services, RSS is very fast and easy to work with. Similarly, AJAX, without any SOAP wrappers at all, has been used to replace web services. RSS and AJAX are not just “Web 2.0”. They could be referred to as “Web Services 2.0”, as well.

Modern software design takes the first two foundations of scalability, and combines them. The revolution in *Enterprise Software* has come from expanding complex systems, by developing task-specific pieces, which communicate easily.

## [WORKING THROUGH THE BUZZWORDS]

*< Linux/Apache as a platform has been doing web-based networking, backed by large-scale community development, for a long time. The most mature SPAM filters, email handlers, RSS tools, web data-miners, and web automation tools are Linux/Apache based. For those tasks, that's what we may recommend. These services or components can be integrated easily with Microsoft Exchange and other Microsoft servers and systems. >*

### Are we talking about “Open Systems” or “Open Software”?

No. We're really trying to discuss the important business move toward open architectures. Enterprise or service architecture is not related to “open systems” or “open software”. For open systems, think “UNIX-like systems which have been standardized”. For open software, think free Linux software, free licensing, and community development.

### Is Linux more “open” for business software architectures?

No, What matters are your software applications, not your systems. Linux is important, however, because most of the web runs on Apache web servers running on Linux, and the internet (worldwide) and intranet (corporate) ARE vital to open architectures.

## WEB 2.0, AJAX, XML, THIN CLIENTS, AND YOU

**Web 2.0 and AJAX have *already changed* web programming and business application development, to the same extent that managed care has already changed the healthcare industry, and boxing gloves have already changed boxing.**

Most web programmers learn quickly how to use a submit button, which allows a user to wait for the next web page to come back to them and tell them the results of their action. Most accomplished web programmers have learned how to display a table of data, and to sort the data by waiting for the page to refresh. Most accomplished web programmers have also explained to clients many times, that they can't type into a dropdown list; they can only choose one of the values in the list, because “that's the way the web thing works”. Now, web programming isn't so simple. Forms can be updated and data can be sorted, faster than you can read the confirmation messages. No one has to wait for pages to reload any more. New descriptions can be added to a list, and stored into a database, simply by typing them into a dropdown box. A web form can enter and correct data faster than a user can type. The admission requirements for the web programmer club are getting higher. Microsoft and Adobe are writing tools to make these tasks easy, but these tools are still in their infancy.

# [WORKING THROUGH THE BUZZWORDS]

## What is AJAX? Does Microsoft do AJAX?

Ajax is “Asynchronous JavaScript and XML”. If you’ve ever used Microsoft Outlook Web (OWA), then you’ve used AJAX. Microsoft wrote OWA with hidden embedded web requests, so OWA could get and display new mail without refreshing the page all the time. Within a few years, everyone was doing this in several different ways. Microsoft was a pioneer, but AJAX is not Microsoft technology. JavaScript (the modest little webpage scripting language) was one surprising ingredient. JavaScript was combined with something we got from Web Services: XML. That’s right-- Microsoft’s hidden browser function was combined with JavaScript and with some of the fast, light, network-ready data formats that we all learned from Web Services and SOAP, to create AJAX. Like magic, XML was being sent into web pages, “from behind”, to display data and reformat pages. The best part? It’s fast.

“Atlas” was Microsoft’s first AJAX toolkit, released in 2007. When Microsoft renamed it to “AJAX toolkit 1.0”, many Microsoft developers were unhappy because they lost the cool name. But Microsoft knew what they were doing, because we’ve already been asked by clients why our AJAX is working, even though they never installed Microsoft AJAX on their server! Unfortunately, Microsoft AJAX toolkit is not yet a tool of common choice, and many Microsoft.NET programmers compare the AJAX toolkit to early versions of FrontPage (they assume it will get much better before being replaced with something else entirely). As of right now, Microsoft AJAX Toolkit doesn’t easily stand up to such libraries as Prototype, Dojo, Scriptaculous, and General Interface, in the same manner that Front Page (which has now been replaced by SharePoint Designer) didn’t survive well in the market against Adobe Dreamweaver. Before MS AJAX arrived, these other toolkits were listing clients on their websites from Gucci to NASA to Apple to ESPN to Sony, and even Microsoft! Developers who use only Microsoft tools may say AJAX is slow, because Microsoft AJAX can be slow. AJAX tools are already far more sophisticated than web service tools ever became, and the simple truth is that good web programming now requires qualified, experienced pros.

## How does NSK do AJAX?

NSK uses an in-house AJAX library, which was specifically designed to do three things. It performs as well or better than equivalent Windows-based applications. It is learned easily by developers, so work for our clients can be done quickly and inexpensively. And our framework is useable on many browsers (in addition to Internet Explorer), and is 100% compatible with all current major server and development technologies, including Microsoft.NET. We’ve seen development turn-around as short as eight months, on financial industry projects of considerable complexity.

NSK created an AJAX framework which is learned easily by developers, so work for our clients can be done quickly and inexpensively.

## [WORKING THROUGH THE BUZZWORDS]

*< Employees enjoy having secure access to documents, reports, real-time business information, and management tools in their browser. Portals work with the latest web and data standards to deliver useable, flexible and inexpensive software applications, to large groups of people in diverse locations. New applications require no deployment costs and little training. >*

### What can Web 2.0 do for me? What is a thin client?

A thin client is one of the things Web 2.0 can do for you. A thin client is something which is already on a user's computer, or something which is very small and easy to install, which allows a user to access a multi-user system, such as a large contact database or a banking transaction system. Web 2.0 is called "Web 2.0" instead of "Corporate Services 2.0" or "Map Delivery System 2.0" or "Email 2.0", because the only thin client required by all these new technologies—is a web browser. Imagine— your application is already installed everywhere.

H&R Block's newest office system was designed with Tibco General Interface, a very mature AJAX and web services tool. Suddenly, H&R Block no longer had to regularly install software on thousands and thousands of office computers. This is a well-publicized example of Web 2.0 success. Such secure corporate Web 2.0 systems require only that a user sit down at a web browser. NSK's latest systems automatically log users onto the data system using their Microsoft Active Directory information, automatically talk to their Exchange mailboxes, and are capable of sending calendar and appointment items within the network or all over the world. Our intranet applications are aware of user permissions, security policies, and group memberships. For clients with even stronger security needs, virtual desktop environments from Citrix and VMWare can be used to deliver intranet workspaces to users, with complete control over customization of the desktop work environment.

### MICROSOFT SHAREPOINT AND OTHER PORTAL SYSTEMS

Microsoft SharePoint is a good intranet gateway, or corporate portal. But is it open? Yes, it can be very extensible if setup properly. It can talk to things, better than things can talk to it. SharePoint is improving every year. SharePoint is perfectly compatible with Web 2.0 and AJAX programming, just as Microsoft.NET applications work with AJAX programming.

There are many open-source portal or community platforms, but most open-source developers prefer (and will recommend) custom intranet solutions which feature modular add-ins.

SharePoint is a great tool for rapid Portal development.  
 Portal pages can easily become custom business applications.  
 In the open-source community, there are many alternatives.

# [WORKING THROUGH THE BUZZWORDS]

*< XML is the closest thing to a universal data language for software systems. HTML (for the web) is a type of XML. All significant SQL databases have XML input/output features. Web Services, and now AJAX and Web 2.0, use XML. XML capabilities should be a part of any software investment. >*

## XML AND DATABASES

What do Microsoft Server 2008, Microsoft SQL Server, all Microsoft Office products, and AJAX have in common? They're all designed to use XML. XML is "extensible markup language". It's like HTML for data. Microsoft's web server, IIS, uses XML for configuration. Microsoft has a huge investment in XML. So does the Internet. You should, too.

## MS-SQL, MySQL, Oracle, and Legacy Databases

I'll tell you a secret. If you put 5 developers in a room, at least three of them will know some SQL database language. They'll be proud of this fact, and they'll each believe that everyone should have to suffer through SQL. This is a strange love-hate relationship. The SQL database language is a mature international standard. But it's crude, almost ugly. And under that skin, it's fast, and you can do almost anything with it. SQL queries to a database are like big cannons. You aim them carefully, and then you push a button and something truly spectacular happens, very quickly.

Access data reports are easy to do, but a common, complex Access report can require an experienced expert. Fourth Dimension (4D) is a fast and wonderful RAD (Rapid Application Development) system which works on Windows and Mac, but 4D queries are an art form common only to the 4D developer. SQL, on the other hand, is generic, and it scales like no other database in the world. Very complex queries can sometimes be very simple to achieve, in SQL. This means, SQL is the least expensive to maintain, least expensive to expand, and least expensive to develop, of any data platform. And, SQL is very good with XML. Finally, there are many report generators for SQL, and MS-SQL has web-based reporting built in. There are few default decisions in technology, but this comes pretty close.

Oracle ruled the world at one time, and Microsoft had no database. When the web exploded, Microsoft SQL was already as fast as Oracle, and cost \$2000, compared to up to \$200,000 for Oracle. MySQL was free, but not yet a strong database. PostgreSQL, was already strong and fast. Today, these four SQL databases rule together. Most web development today is probably done for MySQL, and almost all corporate development we see here at NSK, is done with MS-SQL, with a small amount in MySQL. We encounter Access database work, but only in smaller systems, and quite often in systems that should or must be migrated into SQL.

What of "combination" application and database systems such as 4D and Delphi? The facts above remain. You should probably be data-warehousing with MySQL or MS-SQL. This is standard advice we would give to any anonymous client.

## [WORKING THROUGH THE BUZZWORDS]

*< NSK has a strong history of investment development expertise, but as NSK grows, we gain diversity from developers with experience in many industries. We can advise intelligently for development in many languages. >*

### DEVELOPMENT LANGUAGES

#### Open Source gems. Perl, PHP, Python, and Ruby.

Every web server needs a scripting language, a database language, and a database. The web server for Linux is Apache. Three languages common to Apache/Linux are Perl, PHP, and Python. Notably, the history of these languages is tied to the history of the web, and they continue to grow in popularity. There are many web tasks best suited for development in Perl, PHP, Python, or a combination of the three. Also, Ruby, and Ruby on Rails (RoR), deserves a special note. Ruby has an evangelical following right now. But Ruby is slow, and does not scale. The official site quotes RoR as “only 50% slower” than simpler systems. Whew! Some large Ruby projects have been switched into other languages.

#### Object-Oriented “Big Guns”. C++ and Java.

Java and C++ are “object oriented” languages, very hip and modern. We’ll include C++ with .NET and Windows, below. So let’s talk about Java. For years, it seemed every programming job posting in the daily newspaper was for Java, or J2EE (Java 2 Enterprise Edition), now called Java EE. What happened? Microsoft .NET happened. First, Microsoft put all their languages on a common platform (.NET), and then Microsoft came out with a “Java-like” language, C#. Then, Java practically committed suicide, through imitation. Java attempted to expand outward, to provide Windows development tools equivalent to Microsoft’s own extensive library. The language became complex. Then, Java attempted to expand downward, creating “applets” and “Java beans” to attract new programmers away from Microsoft’s fun Visual Basic .NET (VB.NET). The hungry market was filled with new Java programmers who didn’t know what they were doing. The result was the worse heap of badly written, broken, un-readable code ever left in America’s corporate corridors as a bad investment write-off. Only Microsoft can recover from something like that. Many Java applications have been re-written, many were thrown away. Java has survived its moment in the limelight, and remains a vital and viable platform, but there is no longer a premium placed on Java development over other languages. NSK does not currently deploy Java.

#### Microsoft .NET.

Microsoft .NET is a healthy standard for corporate development. We keep a very strong, experienced VB.NET and C#.NET staff, and we have increasing C++ experience. Our primary development platform is currently Windows Server 2003 and MS-SQL 2005 with Visual Studio 2005, and we are also developing on Windows Server 2008 and MS-SQL 2005 with Visual Studio 2008, We enjoy the luxury of Virtual Machine testing environments.

## [WORKING THROUGH THE BUZZWORDS]

*< NSK knows the difference between clean, maintainable code and bad code. We produce re-useable, well-documented tools and programs. >*

### Microsoft .NET advantages.

NSK development requires a wide range of experience with Microsoft products, currently including: Server 2000; Server 2003; Server 2008; Windows XP; SQL 7; SQL 2000; SQL 2005; Exchange; Access, Excel, Word and other desktop applications; and Visual Studio. With such a wide array of skills required by our development group, areas of expertise between developers often overlap. What does this mean for our clients? If they are running software we developed using Microsoft technologies, we may often have multiple resources available for the client. At times when quick response is important, we will more often have someone available who can respond intelligently to project problems, and we are able more effectively to provide solutions quickly. There are advantages to being in the majority!

Microsoft has built the .NET Framework into Microsoft Server technology, and the .NET Framework allows applications written in any language in the .NET Framework, including VB.NET (Visual Basic) and C#.NET (“see-sharp”), to run directly and safely within the Microsoft Server operating systems. Microsoft .NET Framework programs make full use of such things as Windows system event logs, monitoring tools, debuggers, and optimizers.

Additionally, the .NET Framework is a “managed code” environment. This means that programmers produce code which is able to clean itself up, fix its own memory leaks, prevent some types of improper coding or simple typing mistakes, and generally insure proper program execution. The .NET Framework uses Visual Studio, which is one of Microsoft’s best products. NSK’s .NET Framework developers all use the same editors and tools.

For our clients who deploy applications and programs written with the NET Framework, this provides better trouble-shooting, faster problem-solving, and better error-handling (which means, less crashing). This also means less development cost, and less development headaches! The .NET framework helps us to be most responsive to you, as well as less expensive to you.

Microsoft Visual Studio can create Microsoft Windows programs (programs which run on the desktop, not in the browser). The libraries (code collections) used for Windows are still changing every few years. Current choices include MFC, ATL, win32, and WinForms! The last one, WinForms, requires the .NET Framework to be installed on a user’s machine in order for your program to run.

# [WORKING THROUGH THE BUZZWORDS]

*< Business technology buzzwords are created by the promise that, with a given technology, the corporate consumer's Return On Investment will go up, and Total Cost of Ownership will go down (instead of your server). >*

## WINDOWS VS. THE WEB

### Corporate Software Then

Eight years ago in 2000, writing Windows applications with Microsoft tools was almost rocket science, except messier, and more prone to error. Businesses (and developers) were still using tools like 4D, C++ Builder and Delphi to develop corporate Windows applications which were (relatively) inexpensive, secure, and bulletproof.

Eight years ago in 2000, e-commerce systems were already running complete customer service, cash register, credit card processing, shipment processing and mail list management tools, on their big e-commerce web servers. These “admin systems” ran in the browser in a subdirectory on their dot-com sites. But most businesses couldn't afford to run business applications in a web browser. They didn't have a web development team, and security and user control wasn't good enough.

A best decision would have typically been: Use 3<sup>rd</sup> party tools like 4D, C++ Builder and Delphi to develop for Windows.

### Corporate Software Now

Now, the Microsoft .NET Framework has greatly simplified Windows development. It's taken Microsoft a decade to catch up, but Visual Studio is now capable of producing reasonably quick and well-behaved Windows applications, in a reasonable timeframe. The .NET Framework must be installed on the user's machine in order for the application to run. Windows development takes slightly longer than equivalent web development.

Now, there are no problems with running corporate business tools from within a web browser. There haven't been, for many years. However, perceptions are changing FAST! Remember that for most developers, “Web 2.0” and AJAX truly arrived only last year. Web/AJAX applications are fun to use, behave perfectly as expected on the desktop, require no installation for new users or new computers, and have reduced training times. Web development is slightly quicker than equivalent Windows development.

But wait, there's more, regarding web applications running in a browser. Applications can all be located in a central place such as a corporate web portal or intranet. Applications from several vendors and markets can be launched from the same interface. Arrangement of applications can be changed easily by editing a web page. Applications can be re-branded, have logos updated, or can be re-styled all at once with corporate CSS style sheets, at almost negligible expense.

## [WORKING THROUGH THE BUZZWORDS]

But wait, there's more, regarding web applications running in a browser. Applications can co-exist with modern business information and data such as RSS feeds, document repositories, email and wiki messaging systems. Application organization and integration with standard office tools is again, a matter of web page editing. Applications can easily be converted to web services for delivery of information to alternate devices and locations. Internet "web mashups" can be easily inserted into web-based applications, so that users have access to familiar and required tools such as Google, trip-mapping, street-level mapping and GPS, flight informations, stock quotes, and the list goes on. Dialing and telephone software can easily be integrated into web applicatitons, to automatically highlight phone numbers for dialing, and to make free or inexpensive calls over VOIP internet.

We could wrap it up, but wait, there's more. Web services or purchased databases can be used to auto-complete addresses, check and replace shipping addresses, provide geographical distance searches on contact data, perform real-time stock, credit or D&B retrievals, and even data-mine for free-text phrases on the web. All by editing web pages, rather than loading the source of a compiled Windows application into a professional developer's editor, making careful changes to forms, and then re-distributing the modified application back to every user's machine. I think we are communicating the difference.

There's one thing to remember, here. When talking about open architectures which share data and manipulation-tools between your purchased business applications—it's not just the 3 or 7 or 12 or 42 applications you have installed in your office. It's thousands, all moving at the speed of business. Who knows what valuable product or service is going to hit the market for your business, tomorrow? Now, we can say it:

A best decision would be: Use appropriate languages to develop for your server base (Microsoft or Linux) and develop for the web. Integrate your applications with your corporate intranet and public web site, where appropriate.

At NSK, we're excited about all these technologies. Sometimes, we're a bit intimidated (we'd be crazy if we weren't), but we're very good at rising to the challenge of our fast-changing landscape and exceeding expectations for the quality and performance of our delivered products.

Gone are the days, when software developers locked their customers into closed, proprietary packages and ransomed updates and fixes. We're focused on delivering applications which stimulate cost-effective software systems expansion in our clients' business processes.