

Title: **Taking a Good Look at .NET for Finance**

Summary: The features of 64-bit systems making higher application performance possible translates into more realistic models for quantitative analysts charged with portfolio modeling or risk analysis or similar applications. Knowledge of the .NET programming environment will be important when the move to 64-bit systems triggers consideration of whether it is time to re-write or re-engineer systems. This paper explains what .NET is and why it has special value to systems development in finance firms.

"If it ain't broke, don't fix it!" may adequately sum up the historic hesitation of most finance firms to adopt new programming languages and hardware platforms. Those in-the-know, however, sense tremors that may ultimately disrupt historic predilections for one programming language or another. This anticipated earthquake is the commercial availability of 64-bit operating systems for Windows™ slated for summer 2004, which should cast the .NET programming environment in a new light.

Previous columns have discussed the advantages of 64-bit systems for financial quantitative analysis in some detail. For a quick recap, consider the features of 64-bit systems that make higher application performance possible: greater bandwidth, and therefore greater speed moving instructions and data to and from the CPU; access to much more memory, increasing the likelihood that the next instruction or data needed will be in memory; and the ability to address significantly greater file sizes, enabling direct access to large datasets, such as mountains of stock market tick data. From the point-of-view of quantitative analysts charged with portfolio modeling or risk analysis or similar applications, all of this translates into more realistic models.

For most finance firms, however, it is the confluence of .NET and 64-bit operating systems that is most promising, if and when moving to 64-bit systems triggers consideration of whether it is time to re-write or re-engineer existing systems.

What exactly is .NET and why does it have special value to systems development in finance firms?

First, .NET is not just a variant or new incarnation of Java. True, both .NET and Java are touted as systems for web services exploitation, and .NET was apparently designed with the experiences of Java in mind. Since the Java and .NET creators kept both eyes firmly focused on the Web, the two are alike in their reliance on byte code (halfway between human readable code and relatively fast binary code) and run-time environments built into every machine. Because hardware has progressed, the excess code that used to be built into all programs and weighed them down can now be handled readily by what you can think of as the "ozone" of every machine. The just-in-time (JIT) compilers that you find on every machine make both .NET and Java approaches possible.

But how .NET and Java get to this lighter code scheme is not the same, and how they match up to the needs of finance firms differs yet again.

Java code was designed for platform independence and ease of analysis by byte-code interpreters and JIT compilers. The portability that Java affords is especially important to Independent Software Vendors (ISVs) seeking to write code once for multiple platforms. Another key advantage is that Java is geared to

handle complex, high-volume, transaction-based applications, such as keeping track of the hundreds of billions of daily trades on the New York Stock Exchange. Unfortunately, when it comes to incorporating sophisticated analytical code into Java applications, there are still plenty of challenges. If you then want to crunch data from the daily NYSE trades, for example, Java's numerical limitations quickly become an area of concern which has prevented most quantitative analysts from incorporating Java applications into their work. Performance can be a problem when it comes to numerical computing in Java. As part of its research efforts, The Numerical Algorithms Group has compared the performance of moderately sophisticated statistical routines in pure Java, Java-"wrapped" C code using the Java Native Interface, and pure C. Our results suggest that the pure C code will be five to 15 times faster than pure Java while C code utilizing the Java Native Interface will run approximately 90 percent as fast as pure C. Putting Java on the same footing as current languages will require more effort and some new features of the Java VM implementations.

Like Java, .NET also uses byte code and relies on JIT compilers supplied with new systems. Also like Java, .NET's ultimate goal is to provide users access to their information anywhere, anytime and on any platform or device. However, unlike Java, .NET does not deliver the same level of platform independence.

What .NET does provide is the ability to interoperate in multiple languages. Of interest to most quantitative analysts in finance is that there is a C/C++ compiler under .NET. While Java is a language, .NET is described as a programming environment.

How does .NET work?

First, program in any one of 20-plus languages incorporated in .NET. When compiling to managed code, the compiler translates source code into Microsoft Intermediate Language (MSIL), which is a CPU-independent set of instructions. MSIL includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations. Before code can be run, MSIL must be converted to CPU-specific machine code, usually by a JIT compiler. Because the common language runtime supplies one or more JIT compilers for each computer architecture it supports, the same set of MSIL can be JIT-compiled and run on any supported architecture.

The most practical outcome of the .NET framework is that you can take routines written in the languages that are most appropriate for their function, and take existing routines irrespective of the language they were originally written in, and know that they will compile to the same intermediate language and use common types that can then be assembled together. Finance applications written in C/C++ can be combined readily with routines in C# and Visual Basic. These languages all play nicely, along with various legacy applications that you similarly want to throw into the .NET environment.

So, if the advent of 64-bit computing has you asking how you can deploy applications created for desktops most efficiently, this is where .NET will likely come to play a part. You either bring your many applications into one environment where you program once, or without .NET, you would need to reprogram to get routines to work together. This means that even with the usual learning curve and time that goes into mastering a new system, .NET will likely save development time and money because it paves the way for easier and safer application development. The wide selection of .NET-based components and tools also contributes to a reduction of development and maintenance time. With .NET,

you get to write in the language of your choice, yet integrate into one running system -- a convenience that should not be underestimated.

In the finance arena, .NET's attractiveness once 64-bit desktops are common is more of a projection than current reality. Given that financial application development, like most of the computing world, is moving to the component approach, .NET's ability to mix languages without any problems is a big draw. But don't expect IBM, Sun and others who have invested heavily in the Java platform to sit back and let .NET become a Microsoft stranglehold on the market. And, in the fast-moving world of software development, a new twist seems to pop up daily. In the course of writing this article, Mono, an open source project supported by Novell, began offering the beta version of .NET for Linux, Solaris, and MacOS on multiple chip architectures. With this, we can begin to see the outlines of a environment permitting both platform and language independence. That prospect has to be an exciting one for financial systems developers.

By Rob Meyer and David Sayers, Numerical Algorithms Group.

Originally published by Financial Engineering News, July/August, 2004 (www.fenews.com).

Numerical Algorithms Group

www.nag.com / info@nag.com (North America)

www.nag.co.uk / info@nag.co.uk (All Others)