From: David Vaskevitch
Sent: Friday, May 30, 1997 2:22 PM
To: Bob Muglia (Exchange); Brad Silverberg; Jim Allchin (Exchange); Ben Slivka; Paul Gross; Paul Maritz; Bill Gates; Nathan Myhrvold; John Ludwig; dgmsmi@davidson
Subject: RE: Moving beyond Java

Two broad messages for the market, equally important

1. Microsoft offers CHOICE, PRODUCTIVITY, SECURITY, AND SOME PORTABILITY

2. Microsoft LOVES THE JAVA LANGUAGE and is committed to making BETTER AND COMPLETE for developers

As I read Bob's proposal, which is an important one, it is primarily about message number 2. We should just be careful to always keep #1 in mind, too.

Broadly the world factors into two communities that only partially overlap

Hard core developers love the Java language: it is C++ without the tears. J++ should convince them that finally Java becomes useable for most programming tasks. Ideally it positions Microsoft as a true innovator and a true leader in the language space, for Java too. Actually, true language innovation would be a big deal for us, period (heretical statement: I believe pointers are very much at the center of any proposed innovations; while we're thinking about it we should consider BOTH fast pointers (eg memory references) AND a new class of SAFE POINTERS which could then be put into C / C++ and maybe even VB as well

Managers, CIO's, ISV's and Solution Developers as well as many hard core developers love the NON LANGUAGE part of the Java message. The services which are built into the Java environment and invoked automatically. The security inherent in their interpreted environment. The idea of being able to send code across the wire both to other servers and to workstations. And to the extent people believe, the concept of cross-platform. Our opportunity is to offer these benefits while also offering a choice of languages. The biggest whole in our strategy for getting there is figuring out what to do about C. I claim there is only one possible answer: some form of p-code, an interpreted C. The p-code's will be different from and lower level than Java byte codes, but ought to be considerably higher level than x86 instructions. Perhaps p-codes are an answer. In any case, if we can complete our plan in this area, offering a complete set of services and a machine that is language independent, that offers productivity through services, and security is a big win all by itself. It allows code to be shipped across machines too. Then we just need to decide how many of those services we make available on Unix to have some degree of cross-platform-ness

The most important this is that these messages have to generally ALWAYS BE GIVEN TOGETHER. Just the Java message carries the risk that people might say "if I'm already going to Java why not stick with the original leader" and also "isn't Sun more likely to be cross platform than you even if you do make Java slightly better", and end with "even if you put something cool into Java, won't Sun be right behind you". Just as bad is the services / choice message alone because that puts us in the "you don't believe in Java after all" camp. The two message together though are very powerful and build on each other

-----Original Message-----
From: Bob Muglia (Exchange)
Sent: Friday, May 30, 1997 7:32 AM
To: Brad Silverberg; Jim Allchin (Exchange); Ben Slivka; Paul Gross; David Vaskevitch; Paul Maritz; Bill Gates; Nathan Myhrvold; John Ludwig; dgmsmi@davidson
Subject: Moving beyond Java

We have been struggling with the issue of how to recapture the developer mindshare that Sun has gained with Java. It is clear from Bill's consistent statements that we need to build our message around Windows.

I think that we can do this in a very clear way by changing the context of the discussion to A NEW LANGUAGE Microsoft is creating called J++. J++ builds on Java but adds the features to make it useful for Windows programmers. While J++ certainly supports downlevel "pure" Java, it relegates that to a role similar to ANSI compliance in a C compiler. To be productive on the high-volume Windows platform, developers will very much want to take advantage of the full features of J++.

J++ is both a language and a development tool product. The 2.0 release of J++ will be one of the most productive development tools on the planet. It builds on the RAD concepts of VB but it is a very modern object-oriented language. The roots of this language can be found in Java but there are important extensions such as pointers, structs.

J++ contains a powerful class library. This class library is the equivalent of MFC, but of course it is optimized for J++. The class library only runs on Windows.

We should announce the J++ language at the PDC and enter a wide beta program for the development tool at the same time. The tools ships with the rest of the dev tools next year.

MSS 0205755
CONFIDENTIAL

Many attributes of this proposal are already POR. The dev tools team is building a world-class product called J++. The language extensions are under investigation. There are ongoing discussions to merge the AFC group with the dev tools team and re-vector their efforts to a Windows specific class library. We would need to make sure that this new class library can be finished in time for next years J++ release.

What I am suggesting is a change of mindset: Java is old stuff which we treat as a compatibility thing. Our efforts are on a new Windows-focused language and product. That is where the big change comes in.

This would also mean that we truly treat all future Sun enhancements to Java as a compatibility tax. While we do support them, we certainly don't do anything to help them run well in our J++ environment.

ooo