Erik Stevenson (LCA)

From: Aaron Contorer
Sent: Friday, February 21, 1997 12:47 PM
To: Bill Gates
Subject: Memo material

This is 1 page of new material,
followed by some closely related material I wrote and sent you on 2/12.

Today we face the largest threat Microsoft has faced since the success of Windows. For the first time, there is a really credible threat to our position as the leading platform for ISVs to write to. Windows faces challenges in satisfying end users and IT organizations, but we have a lot of smart work underway to address these problems. By contrast, we are not executing on a strategy that lets us maintain our leadership position as the people who define the platform for ISVs.

Owning this platform is *the* Microsoft asset. It is the difference between growing to twice our current size in the future, or shrinking to much less than the role we enjoy today.

There are three possible ways to address the threat of the Java platform. One is to do nothing and gradually die as others innovate around us. The second is to join the parade of people who are saying "let's kill Microsoft and share their market among us" - good for everyone else, but reducing us to the much smaller role of a common software company like Lotus or Borland or even Symantec. That's a great way to make all our stock options worth zero, even if we would not technically be out of business. The third choice is to make major innovations to our platform so people still prefer to write to us instead of some tepid cross-platform Java layer. This is our only real option.

For over half a year I have been upset that some people at Microsoft are apparently working hard on plan 2 to destroy the value of the Windows API. Of course I agree that we must win against the Java platform, but a belief that we have to just match everyone else's actions one for one is fundamentally misguided and wrong - it makes us a commodity player, one of the pack, instead of the leader.

Centralized Computing

Sun, Oracle, and Netscape are all pushing a new model of [almost] centralized computing. They all acknowledge that Microsoft holds tremendous sway over the desktop platform, so they all want to quickly strip as much value and spending as possible *off* of the desktop and onto the server where they can charge premium prices and push their own platform offerings.

At the same time, they know this is fundamentally wrong. There are good reasons why a big company in the 1990's uses thousands of small and midsize CPUs instead of one giant Cray supercomputer to do all the work. Centralized machines have poor price/performance when they get too large; they have high latency for ordinary interactive tasks like typing and even worse latency for multimedia (unless you literally spend a fortune on your network); and they fail to take advance of the principle of colocation - putting the processor close to the inputs and outputs it needs to work with.

Our competitors are not stupid, so they are pushing the Java platform as the solution for programs that really need to run closer to the user. Sure, it's a half-assed solution and isn't compatible with anything and in fact scarcely exists, but hey, at least it's not Windows. With Oracle and HTML-generating code on the server and a browser with Java on the client, you have a very crude, complicated, but functional platform for developing line-of-business applications - more specifically *distributed applications* which take advantage of all the interactivity and media-richness that purely centralized mainframe apps never had

Fortunately for us, this solution is an incredible hack. Real applications require work in Oracle and Java HTML and GCI, and except perhaps for DNS, no unifying architecture ties the whole thing together. If you

want to write an app like Amazon.com or a comparable intranet app, you are on your own. Even a cool tool like Visual InterDev merely serves to paper over this disastrous platform, not to fix it.

This situation leaves open a huge strategic opportunity: to provide a better way to write distributed applications.

We need to make clients and servers more powerful and functional. But more crucially, we need to ensure that our platform - the thousands of person-years of proprietary code that we license to customers - makes it incredibly easy to write real business applications in all their richness and complexity.

- *end of brand-new material* -

Switching Costs

In economics there is a well-understood concept called *switching costs* - how much it costs for a trading partner to change partners. Our philosophy on switching costs is very clear: we want *low* switching costs for customers who want to start using our platform, and we want to provide so much unique value that there are in effect *high* costs of deciding to move to a different platform. There is a name for this: it is called Embrace and Extend.

Embrace means we are compatible with what's out there, so you can switch to our platform without a lot of obstacles and rework. You can switch from someone else's Java compiler to ours; from someone else's Web server to ours; etc. Customers love when we do this (as long as we don't spend our energy embracing extra standards no one really cares about); our competitors are not so sure they like it because they prefer us to screw up.

Extend means we provide tremendous value that nobody else does, so (A) you really *want* to switch to our software, and (B) once you try our software you would never want to go back to some inferior junk from our competitors. Customers usually like when we do this, since by definition it's only an extension if it adds value. Competitors hate when we do this, because by adding new value we make our products much harder to clone - this is the difference between *innovation* and just being a *commodity* like corn where suppliers compete on price alone. Nobody builds or sustains a business as successful as Microsoft by producing trivial products that are easy to clone - that would be a strategy for failure.

If we fail to embrace, we *can* lose because there are big barriers to buying our products. But if we fail to extend, or do only humble work that is easy to clone or to surpass, we *automatically* lose because our competitors will spend literally billions of dollars to clone our work and replace us.

The Windows API

Windows was a very successful embrace-and-extend move. People already had DOS machines and DOS apps, and we were able to go in and say "add this to your machine and it will just get better." Wow! What a deal! It seems to have worked out all right so far. NT is a very similar move; although it's not trivial to upgrade from Win95 to NT, in general you can use the same computer, same apps, and same APIs as before, *plus more*.

The really big win in Windows is the API. An app that calls the Windows API is effectively calling upon thousands of person-years of engineering work to help their app get its job done in a very specific way. You could argue that the API is too hard to use, that not every library is as fast as it should be, or other serious imperfections, but the fact remains: if you took away Windows, that application would no longer work.

The Windows API is so broad, so deep, and so functional that most ISVs would be crazy not to use it. And it is so deeply embedded in the source code of many Windows apps that there is a huge switching cost to using a different operating system instead. You can't just take a Windows app and stick it on some weird Java NC from Oracle, for example, and expect it to work - the guts just are not there. For many customers, the cost of reworking all their apps would be huge.

It is this switching cost that has given customers the patience to stick with Windows through all our mistakes, our buggy drivers, our high TCO, our lack of a sexy vision at times, and many other difficulties. People have tried to clone Windows, but it is just too hard to do well. Customers constantly evaluate other desktop platforms, bit it would be so much work to move over that they hope we just improve Windows rather than force them to move.

In short, without this exclusive franchise called the Windows API, we would have been dead a long time ago.

The Java Platform

So along come Scott McNealy and Larry Ellison, saying "hey, we've got a good new programming language called Java." Fine, we like programming languages a lot. After all we *are* a software development company. The problem is that very quickly they also said, "we've got a whole new platform, a whole new set of runtime libraries and APIs, to go with it - so as long as you are writing your apps in a new language, you might as well write to this new platform that we say lacks the flaws of old Windows." In other words, they are saying, switching costs will never be lower than they are right now - the barriers are low - so join us now.

You would think it would be our top priority at such a time to (A) fix any serious flaws in Windows which could push customers over to the Java platform, (B) add so much new and unique value that this vaporous "Java platform" doesn't sound very attractive anyway, and (C) make damned sure that our new value is really hard to copy so it doesn't show up tomorrow in Sun's or Oracle's offerings.

We are doing all of this. We are fixing TCO and further improving our dev tools. We are providing new value such as Viper and great multimedia and unified storage. We are making sure that Windows, *not* some new platform, is the most attractive place to run apps written in this now programming language. We are building the best virtual machine in the world, and optimizing it to run on Windows. We are even making sure you can run your Windows apps remotely on an NT server if all you have on your desk is a GUI terminal. As if all this work were not already hard to copy, we are also getting a bunch of patents to further protect it against cloning.

Following the Java Parade

So it is with some amazement that I listen to a number of people who just don't get it - who think we should do work that actually makes it *easier* to copy our work and to run apps written for Windows on other platforms. That flies in the face of everything we are trying to do - it's almost like a suicide attempt. The philosophy here seems to be "our competitors' products are getting more press than ours, so we should kill ours and build copies of theirs instead." This is foolish. Since when did we start *believing* our competitors' press releases instead of rebutting them?

Let me be clear; we have no problem with the Java *language* or with running Java apps really really well on our platform. But we are explicitly *not* in the business of making it easy for people to write apps that get all the features of Windows on a non-Windows platform. "Pure cross-platform portability" is another way of saying "commoditize the OS." In this vision, every OS is just an engine for running this layer called Java as fast as possible, and adding any value below the Java layer is explicitly against the rules.

Sun has already figured this out and has launched its "100% pure Java" marketing program, which literally certifies apps as running the same on any client OS. Programs that call a Windows API or use ActiveX or DirectX, or *any* platform-specific feature, are by definition *not* 100% Pure Java, and are therefore evil. Hey, if you were Sun, you would say this too!

Both Sun and Oracle make their money primarily on servers. (Sun still has some workstation market share, but NT is inevitably eating away at their share and their profit margins on the desktop.) So these companies have every incentive to turn the desktop platform (aka Microsoft's main business) into a cost-driven commodity and focus all the high-margin business onto servers where they (especially Oracle) have a real fighting chance against us.

This is all the exact *opposite* of what we want to happen. It is critical to us that application writers choose to take advantage of features that are (A) part of Windows, and (B) extremely hard to clone. Therefore it

would be a huge mistake if we (A) spent all of our energy just embracing other companies' innovations, or (B) asked key groups to do extra work that makes it that much easier to replace Windows, such as making Visual Basic apps run on the Java virtual machine.

We have enough people trying to kill us without us helping. It is our goal to make them *lose* while making ISVs and customers very happy by delivering great benefits.

Making Real Progress

With technologies like DCOM, Viper, and client-side persistent caching, we are just starting down the long road to the distributed world. There is a lot of design work to do, and a lot of intelligence to build into the OS and the network and the tools. And critically, each part has to be managed be the person or program who knows how to make the best decisions. Web site designers should not have to design their whole site around the latest statistics on who has what browser. End users sitting at desktop machines should need to do *nothing* - no Setup or anything else - to get computation to happen on local machines, just as they do nothing to enable the server apps or Web sites they connect to today. Business system designers should not all have to be experts on variable-speed wide-area networking. Library administrators should not care if a student brings in an app from home and wants to run it on a public kiosk machine for a while. Users with laptop machines should not have to know or care how the right things from the server magically get replicated to their local disk before they leave for a trip. An engineer who needs a big calculation done should not have to care which machine has spare CPU space, and an artist who needs to save 800MB of images should not have to manually hunt around for disk space. The list of "shoulds" goes on and on.

None of this is provided today by the Java platform, but one by one each of these features in being worked on by many people at MS and at our competitors, and each will get properly implemented by someone. We have an opportunity to make many of these advances part of the Windows platform we get paid for, or part of the Java platform that is given away for free. As a shareholder, which do you want?