From: Greg Stein
Sent: Tuesday, August 06, 1996 4:08 PM
To: Hank Vigil; Brad Silverberg
Cc: Arnold Blinn; Rob Price; Tom Johnston; Greg Stein
Subject: RE: Gravity or Anti-Gravity

Since we've passed the Beta hurdle, I've been thinking a lot about where we'll be going once we hit Beta2 and/or RTM. Obvious directions are integration w/ Normandy, Denali, IStudio/Frontpage, Viper, etc.

A rather less obvious direction is the componentizing that Brad mentions. I believe we can (rather easily) turn our purchasing process into a fantastic component (available thru COM) that can be used by the Normandy team (for billing), for micropayment systems, for soft-goods purchases and delivery, and for custom development Denali-based applications or other web-based content systems (Slate and MSN). There are a couple other components in our system that be pulled out into COM objects (our shopper, administration, basket, and database handling modules), but I don't see anything nearly as important or broadly usable as the purchasing process.

Even more so, as I mentioned to Brad at the Workshop yesterday, I believe that the purchasing process is very patentable and we should push for international patents on it. It is a truly unique system for managing a purchase, yet it is incredibly simple to code. Given one of our Merchant manuals (let alone a copy of the US patent), a rather lame programmer could code it up in a day. Of course, the question here is whether we see this as strategic enough to try to cover it with a patent. I think a simple answer is this: we want to be the focus and implementation of all electronic purchase transactions; we have the simplest and easiest model for customers (merchants) to use; we want to protect that.

-g

From: Hank Vigil
Sent: Tuesday, August 06, 1996 1:31 PM
To: Greg Stein, Arnold Blinn; Rob Price; Tom Johnston
Subject: FW Gravity or Anti-Gravity

food for thought.

Henry P. Vigil
hankv@Microsoft.com
(206) 936-3123 ph.
(206) 936-6037 fax

From: Hank Vigil
Sent: Tuesday, August 06, 1996 11:47 AM
To: Hank Vigil Direct Reports
Cc: Satya Nadella
Subject: FW: Gravity or Anti-Gravity

fyi. I want to discuss in our offsites this month, which I'm currently scheduling.

Henry P. Vigil
hankv@Microsoft.com
(206) 936-3123 ph.
(206) 936-6037 fax

From: Brad Silverberg
Sent: Monday, August 05, 1996 3:33 PM
To: Paul Maritz; Hank Vigil
Cc: Anthony Bay; Anthony Bay
Subject: RE: Gravity or Anti-Gravity

GOVERNMENT EXHIBIT 157

From: Hank Vigil
Sent: Monday, August 05, 1996 2:35 PM
To: Paul Maritz
Cc: Anthony Bay; Brad Silverberg
Subject: Gravity or Anti-Gravity

In thinking about the server apps business models, and on how we are structuring our product strategy relative to netscape, several questions have popped up. I haven't sorted this out yet into a clear set of answers, but I want to get these questions out and discussed.

Aggregation vs. Disaggregation of services

This started for me as I realized we are banking on the force of gravity, pulling levels of functionality into the OS. Our model is that at a certain point, key areas of functionality like browsing or HTTP server become generally important enough that they belong in the OS. This thinking extends even to marketing issues, where there is gravitational pull on the server apps to fold into the BackOffice brand.

What is interesting, is that this dynamic seems to come at a cost of speed to market, competitive focus and pragmatic, truly customer based innovation. Let me elaborate:

Once something has been pulled into the OS, the requirements of quality, breadth of compatibility and scale mean that lots of dependencies and trade-offs happen. The net result is that the monolithic code base ships on long cycles after lots of testing. There is also a tendency to meet all needs: be everything to every customer.

Despite the advantages of integrating more and more functionality into the OS, there seem to be areas that can/would benefit by breaking out of the OS so that they can develop richer functionality faster. This allows for groups to discover, re-define and exploit customer needs in ways that are hard when teams believe that OS gravity is the central law. This seems to be a question about distinquishing between the OS and the platform in a more explicit, rigourous way. Put another way, how can we disaggregate appropriate fucntionality while not losing the integration upside?

This is a very good and important point. To me, the optimal strategy is something in between: key components evolve and improve and get delivered independently of the OS release cycles, and then synch up when there is an OS release, providing additional integration. Clearly the needs for many components require that they release in much faster cycles than the OS itself can. The most obvious example is the browser. Yes it will be integrated into the os, and ie4 integrates deeply enough that it takes over the os's UI; but it is on a much faster release schedule. We would be dead if we had to synch with OS's. Look at how long, for example, it's taking to get OSR2 out the door and how little content there is (yes, IE3 is stretching it even further but the basic point remains). It has been *so liberating* for my team to not be tied down to OS releases; we've been able to accomplish an order of magnitude more as a result. [Note also the accusation that Andreessen makes us against us: that our stuff is tied down to OS releases and we have those every 5 years. It's a lie of course but the point is made.]

The browser is the obvious case to deliver on faster cycles than the OS. We're doing it with the shell too. We're going to have to do it with IIS (your next example) if we want to keep up -- we have to get Denali, out, for example, and can't wait for an OS release (NT5) for that to happen.

Examples

A potential example of this is IIS, where now that it is part of NT, the focus is on synching with NT schedules and longer term OS plans. One could argue that shipping quick competitive versions that incorporate things like Public Key, that would position IIS as the leader in key emerging areas. This isn't a criticism of the people, but rather a question of whether IIS is the platform for Internet app developers or NT. If the former, I argue we should disaggregate the product development effort (into product units?) so we can move quickly to win, or minimally, protect our franchise. It should be noted, that this doesn't mean the pricing/packaging need to be seperate from NT. We could still adopt a penetration pricing strategy for some pieces, and still create effective bundles with the OS.

As an counter to the above example, let's examine "security". Prady's team has been disagregated in the sense that the trade-offs the team thinks about are pushing ahead of NSCP and implementing technology for Access control, value transfer and secure channel. Disaggregation creates intense focus. The strategy is based on open protocols and Public key, so legacy integration with the OS, while important, isn't the driver. I'm not sure this would be the case if NT was the sole delivery vehicle for our security architecture. Prady's team has NT as a the key customer moving forward, but meeting the needs of NT happens on a schedule that allows the team to also compete with NSCP aggressively.
My guess is we could say the same thing about directory. Today, the OS group is focused on a very rich strategy that is executing response to NDS, Banyan etc, while also trying to get hands around the LDAP groundswell. Becuase of this, we will likely follow vs. lead with LDAP. If we disaggregated our directory strategy along the lines of Security, we could build faster without giving up the requirment of integration. It just that integration isn't thre driver, but meeting the custemer need for simple directory access sooner.

There is no question we should have disagregated Directory. We are way behind now. I wonder how many other places there are like this?

Competitive Pressure

I think Netscape is pushing on this approach very smartly. For example, Netscape's concept of Full Service Intranet includes services for Email, groupware, Security, Directory, Database access, Network managment and commerce are positioned as the more interesting, "valuable" part of the platform for appplcations developers. NT for them provides scheduling, I/O and the file system. Of course, this is way too simplistic, but this appeals to the Internet developer. We have to acknoledge this appeal and i question whether the gravity strategy is a compelling response.

The 'interesting' new value is in these middleware layers... By being ahead, they get mindshare, leadership, and marketshare; and put us in a reactive position.

If not compelling, how else should we compete against this? Is it NT? Back Office? Normandy? Or rather, do we need to restructure our offering so that we can "embrace and extend" this full service concept. The fact that it comes from Netscape shouldn't deter us from embracing this approach. At the heart, the full service network concept is a customer centered bundling of functionality that avoids as much as possible the traditional platform decision process. People are pragmatic and don't feel equipped to navigate the complex shoals of a "platfrom decision". They just want to publish content on the internet and figure out how to get paid for it. Slate is our latest example.

The trick is to lower the barrier to making a decision, while providing the benefits of integration. When we make things 'low barrier' decisions, customers need less compelling reasons to adopt and we do better. For example, MS-DOS 5 and 6 were viewed more as "utilities" than platform changes. By contrast, Win95 was viewed as such a major platform decision, it was not an easy decision -- it required lots and lots of studies, justifications, and plenty of easy reasons to say No. This lead us to decouple, for example, Nashville from being a release of Windows. As "win97" it's another platform class decision. But add-ons are low friction decisions....

Our Response

Our model to respond should be to componetize these functinnal services. As a customer, you aren't required to buy the whole enchilada, but can buy piecemeal, without losing the option of more pieces and great integration moving foward. The key here is flexibilbity and openness without surrending the integration upside. The component services includes functional services like email, directory, security, payment, membership, replication, news, chat etc. This also opens up our acquisition strategy. We have lots of cash; can we use it to quickly fill holes or take leads. eshop is an example of this. We could still assemble applications (service bundles?) built from these functional components, with a clear customer need as the focus. Merchant is an example So are Netscape's , Publishing and Community apps. So, our pricing should reflect a component pricing scheme that balances the need for the components to make it on their own, while still offering bundle pricing or unique customer pricing per Cameronm's currend Public Network model.

I fully agree. We need a collection of basic components, all of which integrated well together, and which we put together, along with some target specific code, into customer-specific solutions (systems). One of the efforts going forward in IPTD is to make merchant and normandy more componentized and better integrated together -- so merchant can use personalization, membership services, etc.

This is full cicle, since thoughts about pricing drove thinking about our product strategy. Again, I haven't provided answers, and there are obvious simplifications here. However, I do think we need to squarely address where the OS seperates from the component platfrom, and think through all the implications. We need to do this quickly. NSCP is.

Henry P. Vigil
hankv@Microsoft.com
(206) 936-3123 ph.
(206) 936-6037 fax