PLAINTIFF'S EXHIBIT
2563
Comes v. Microsoft

EXHIBIT
DH 7/27/01
49
Peters

From: Steven Sinofsky
Sent: Sunday, October 06, 1996 8:01 PM
To: Jon DeVaan; Chris Peters
Subject: RE: Access, Internet studio, VB and other overlapping products

I'm scared because I think adam and I agreed a little bit. Here is sort of what I would say...I won't send this. Basically at this point, I am realizing two things:

- I still will not sign up for another dependency on adamb.
- I can't do what one has to do and just ignore bill. It hurts me too much personally to see him get frustrated because we are not doing "literally" what he says, and I can't work with that guilt.

So basically I don't know how I can make progress on this trident thing.

There are a number of interesting things to think about in both bill's mail and adam's reply. We are trying to work through many of these issues right now as we plan Office9 and "new office", so it is hard for me to speak for everything we are doing as if there is complete concensus or agreement, so what follows is my personal thoughts.

I do not disagree at all with the assertion that our applications (Office9 or new office) should target HTML and scripting and that is what we will do. However, I think Adam and I agree that this is much more easier said than done.

We are at a point when each office application is set to be made obsolete by various web-based applications (over the next 3-5 years). As adam aludes to, this is not something that will happen overnight, but will eventually happen in some form or another. The two products that are the subject of this mail (VB and Access) are very clearly going to lose out in the web tidal wave, and very rapidly since they rely on developers, who are very rapidly making the shift to developing in HTML and JavaScript. Even right now, I would be hard pressed to tell a customer to write a VB or Access application instead of an HTML form-based application, even with the severe/crushing limitations of the HTML world--anything that uses a server database benefits enormously from being HTML-based.

Word is clearly threatened by some form of HTML page editor, most ironically our very own (Trident). Excel is threatened by the fact that in a short-sighted way it seems far easier to prepare a variety of static reports on the server and dish those out over HTTP, rather than let people use Excel connected to a snapshot of the live data. The use of Excel as an ad hoc tool will be perceived as Important, but not worth the cost (of course this will put IT back in the hot seat as a bottleneck). PowerPoint is already viewed as a tool for the minority of users. As adam points out all of these products have a huge amount of domain specific knowledge in them that will continue to remain important and transfering that knowledge to a new runtime could be a huge win.

I agree with building on HTML and Scripting as the runtime--that is not very controversial. I think by doing so we will find a lot of things we do today will not be possible, and likewise we will be able to do a lot of new things. On balance this will be a difficult transition for Office9. I think the focus of this will be out of necessity still be on "publishing" to HTML. This is distinctly different from "going native" since it is not clear to me what that means for PowerPoint or Excel or Word. Just taking those three-where does something as simple as interactive slide sorter view, worksheets, or just plain old printing come from? These are pretty difficult services to program when you do own the runtime, and even harder when it is immutable.

The separation of design time from runtime continues to remain a thorny problem that is not made any easier by Trident. The two products mentioned above, VB and Access, are good examples of just how hard this problem is. Both products have runtimes which are very large. The Access runtime is the same as the access design time, though through a bunch of checks at runtime the design environment is not accessible--yet we still ship the design time and most of the code is necessary for an Access developed application to run. VB's runtime is similarly large, but because the design time is a very elaborate application (the compiler, debugger, project manager, etc.) you can save a lot of code. But the fundemental aspects of VB are in a the large number of VBRUN DLLs in the \SYSTEM directory. PowerPoint experiences this same issue--the "player" for PowerPoint is nearly all of the product.

It is easy to say this is an architectural shortcoming, but it is hard to find an example amongst all of our products collectively where there is a very rich end-user editing environment and featureset, that can also be easily separated to allow viewing. I'm not saying this can't be done, but I am saying this is very difficult and there is substantial cost in terms of size of the document and speed of rendering, not to mention speed of loading and saving. Many like to talk about the separation of display from data in HTML, but that is just "good old days" thinking--from the day the <BOLD> tag was added this has not been true. Scripting with events makes this worse since you are sending a specific description of what the document should do (and look like) when viewed, and Trident makes this even fuzzier by making it possible to have a "self-modifying" document.

For me, this means that targeting HTML+Scripting as a runtime will be a huge challenge. It will be very difficult to do this for any sort of application that has a pre-defined notion of how features should work (i.e. Office9), since there is such a good chance that the new runtime will prove to be a difficult environment to get the same behavior. I liken this to the difficulty DOS programmers had with Windows early on, when they kept looking for DOS-like ways of doing things that did not make sense in a world of event-driven programming. This is especially true since we talk about HTML+Scripting as a PLATFORM.

But these difficulties can be addressed in some form or another by enough problem solving thought and cooperation, as well as just giving up on a lot of things our applications do today. But there is still the issue of document creation--how does this work in a world where HTML+scripting is the runtime?

This is really the most difficult question for me to answer and I don't know. Trident is an editor, but it is an editor that will create a new document type--one that is mostly an interactive mail message or a database form. This is not a presentation, not a spreadsheet, etc. From the development tools point of view (Access and VB) using Trident as the editor seems very straight forward since the value of the tool will be in the connection of various trident elements together (connecting the controls to a database or events, for example). Similarly, Outlook using Trident is a total no-brainer.

However, for a tool like Excel, Word, or PowerPoint the value these tools provide is in the editing environment that is tightly coupled to the "runtime". In fact, not having a runtime is what makes Word and Excel so easy to use. PowerPoint, as I mentioned, has a very intertwined runtime and design time. The domain specific behavior adam mentioned is in the editor, not just in wizards. So we are basically back where we started two years ago -- "word just becomes Forms3" and I don't know how to build Word if we are not building the editor.

I'm at a loss....

-----Original Message-----
From: Adam Bosworth
Sent: Sunday, October 06, 1996 2:10 PM
To: Bob Muglia; Bill Gates
Cc: Aaron Contorer; Richard Fade; Steven Sinofsky; Paul Maritz; Nathan Myhrvold; Brad Silverberg; John Ludwig
Subject: RE: Access, Internet studio, VB and other overlapping products

I'm not competent to address whether Access is or isn't getting good architectural attention so I will not address that. But I will talk about the issues with roles for Front Page, Internet Studio, and Access. Indeed I'm surprised that you didn't include Powerpoint in there because I believe that Powerpoint faces challenges very similar to Access.

Here's the issue. It is critical that we develop quickly applications that are great domain specific authoring tools for HTML and shortly for what I have called "Chapters" (memo reenclosed below for those who missed it). A product like Powerpoint is still badly needed because it understands the user model for authoring slide oriented presentations. But the runtime IMHO would be Trident, not some other binary. At most it might include some Java/ActiveX compoents to allow special effects on pages at runtime. This powerpoint like product should know how to exploit Trident's HTML. I'm routinely building pages and mail internally using Trident these days that animate text, dynamically expand bullets and collapse them, and cause text to change its look and feel as the user hovers over it. This tool should allow mere mortals to do this without having to write the Javascript code I use to author it. It should make it easy to build simple animated scenarios of the sort I'm going to demo at the PDC. Similarly, a product like Access is even more badly needed, namely an authoring tool for building applications to collect and display and search data, but this time with Trident as the runtime, not some custom binary runtime. Again, some components might be included in the pages to help, but the runtime would be the browser. This product would/should be bought to author a site and manage one with domain specific knowledge about what people want when they are authoring data-centric applications. Ideally, this tool would also build the server side component necessary for interacting with the user. Both tools would be far more mere mortal focussed than VB would be. Lastly, there ought to be a tool which lets programmers author pages with script behind them, lets the programmers also author components in whatever way they want, and then lets the programmers assemble these pages/components into sites. This tool is for programmers. This tool would be like a VB that also let me build SQl Server Stored procedures today, only in tomorrow's world it would help me build the Denali intermediate component as well. I'll call this tool the VB-like tool.

So we have a need for 3 tools that are like 3 tools we have today, Powerpoint, Access, and VB, but yet are all totally different in that the runtime is the IE shell plus Trident plus some simple components authored by the tool plus active sites on the server (if necessary and it often will not be especially on the Intranet!). It is easy to say that our current 3 tools should just be these tools.But there is a huge problem here. Backwards compatibility. At 2 levels, usability and programmability. For a long-time to come, there will be things that the current tools make easy for the user to do at runtime that may not be easy in the Trident runtime. Sure components could be written to solve this problem, but that begs the point. Who helps showcase the HTML extensions/runtime we've enabled and why not exploit it. Much worse, tho, is the issue of programmability. The programming models for the current products are going to be fundamentally different from that of the new product. Not a little bit different. A lot different. Pages are documents. HTML isn't Access or VB forms or Powerpoint slides. Don't under-estimate this work Bill. It is as big as building these products.For that matter, a lot of the more artistic of of the Web site authors don't want to program in a scripting language at all, they want something like TerraCotta. And they think today that Director and Shockwave meet that need.

So we have a problem. We see Internet Studio off building something that I think is supposed to evolve to be both the Powerpoint and the Access replacement which, frankly, I think makes it hard for it to be great at either. We see Front page off rapidly building a great replacement for word in the new paradigm, but not really focussing on the other issues. And we don't see anyone really worrying about the VB replacement except VB, but they appear profoundly unwilling to focus on the page authoring problem as opposed to the component authoring problem and they also are trapped by backwards compatibility. You run the risk if you mandate that there be just 3 products of getting none to market quickly enough and having none of them be great at what they do.

Personally, I'd vote to focus. Have Powerpoint stop working on Powerpoint, forget Escher, and just work on building an equally great tool to exploit what Trident 1 and Trident 2 will enable along with cool components and Tera-Cotta to let mere mortals do this. Let Internet Studio focus on letting mere mortals build data-centric applications for the Web. This is terribly important and we need this tool to showcase the Active Data Connector and Trident Dataawareness architecture as well as it is working on showcasing Denali (more on the PARAM's in a moment). Let Bob build Developer Office and "TRICS" to be one and the same thing, namely a developer's tool for building components, pages, Chapters, and sites. And please, do it swiftly! Let Access be a legacy product that can "publish to HTML" but has its own runtime as well and is backwards compatible for the 12 million users of Access today.

Lastly a comment on the PARAM's. It is clear that what is needed as soon as possible is a model for extensible grammer (aka DTD)/rendering in the HTML model so that the data required by components within the page is shown inline and grammatically for these components with extensible TAG's and Attributes for each such component. This will not happen in Trident 1!! Indeed, my architect, Gary Burd, thinks that we should do it in a rewrite of Trident in Java that Don and I might collaborate on and he and I can explain why Java o all of you. It may not even happen in Trident 2 therefore. until there, we're going to have to live with ugly obviously glued in components. The big risk is that someone builds a Java runtime that supersets HTML, but does exactly this using Java classes, call it JTML which isn't constrained by being identically backwards compatible to HTML, and makes HTML obsolete because of its easier authoring and extensibility.

---------

From: Bill Gates
Sent: Sunday, October 06, 1996 11:54 AM
To: Bob Muglia
Cc: Aaron Contorer; Richard Fade; Steven Sinofsky; Paul Maritz; Nathan Myhrvold; Brad Silverberg; Adam Bosworth
Subject: Access, Internet studio, VB and other overlapping products

Recently I was trying to figure out what Internet Studio is and thinking about the future of Access.

I am very worried that Access is not getting good architectual attention because it is sort of part of tools and sort of part of Office. The last time I met with the Access group I challenged them to explain to me what it would take for someone to be able to write Outlook or Project in Access. In particular I talked about being able to right click on a data row and see a set of actions as an example of something that should be very easy. I have never seen any response to this challenge.

Access certainly has to move to use Trident as a runtime like so many other products we have.

Access also have to think about what roll HTML plays inside the product. Access is our product for letting people define reports. Access should allow those reports to be browsed interactively - diving in on detail data simply by clicking on it like an outline. Access should support most of the richness of HTML output in these reports.

Why is the difference between Internet studio and Access? I can't detect any reasonable difference. Internet studio has taken an approach of putting onto HTML pages the most ugly Microsoft garbage ever seen since COM/OLE programming in C++ was declared a success in order to block langauge invocation. I am still blown away by seeing all those ugly PARAM statements in the HTML totally confusing anyone who tries to do anything. If something isn't part of the WYSIWYG output then it should be succinct and understandable. This was the opposite of that.

We all know that Internet studio on Trident will be better but what is the difference between Trident by itself, Front Page on Trident, Access on Trident and Internet studio on Trident? My answers are:
a) Only enough editing to do common email scenarios/compete with Netscape and bundled with OS
b) Much richer editing capabilities including the ability to make rich forms
c) The ultimate tool for generating dynamic HTML from data with queries and reports
d) A product I don't understand.

The basic approach we are on now is to take a data tool -Access and allow HTML content to be part of what it generates for reports (including interactive reports) and queries. The thinking here is moving slowly. At the same time we take an HTML oriented tool (Internet Studio) and start to embed report/query generating commands in the HTML. The result is 2 products that both do a poor job and completely confuse anyone who wants to figure out what we are trying to tell them to do.

A simple question is: say I want to take a database of sales data and let people browse that data - what approach does Microsoft reccomend.

I need to see new vision for Access. I also need to see quite a different vision for Internet studio before I agree it should exist as a different product at all.