From:	Steve Madigan [stevem]
Sent:	Tuesday, December 28, 1993 3:21 PM
To:	Tony Williams

BUSMOD.DOC

1. What sells Apps in the Future

In the section on Containers and Applications I touched on what could constitute the basis for a totally new business model - a strong shift in the user model. In short, as users move towards templates as the basis of the new things they create, there will be a corresponding shift in what a template is, and what corporate developers do to build them -- and the vertical applications which are their extreme form.

I will now step through some logic that takes us directly to a very different basis for competition that results from this shift - the part of the paradigm shift that has the potential to totally reshape the basis of application purchase decisions especially in corporations. I do not pretend to fully understand this, and you should read this as a scenario - it is one of several possible outcomes, the one I happen to understand partially and believe at this point. More thinking is needed on this.

As a starting point, it's clear that our applications are becoming a development platform, and this is just good. When Office releases with programmable Word, Excel, and Powerpoint, the number of objects, interfaces, commands, and properties in our Office product will dwarf Win32 - and even Cairo - by a long shot. This is just the beginning - this will eventually become a delta measured by an order of magnitude or two.

But as we look at the programming environment, the environment for constructing customized, corporate applications, we are *still* presenting a model of customizing "an app". Yes, OLE has made this better since you are now able to use different kinds of content when you customize the app. But we are not yet to the point where you can just build your custom app using VB and a bunch of controls derived from our application data types. There are a number of missing pieces that must be filled into this puzzle before it's possible, most notable among them being the forms work, as this is key to a huge class of corporate applications. Another one of them is this: if you build your application purely out of components, not using some MS Office app as the container, then what IS the container? The answer *might* be a form, it might be a (CDE) folder, it might be both, it might be something else. It IS some structured container....but we don't have such things, except as manifested in our applications, where it's tied up with a bunch of other stuff that may or may not be useful.

If we look at this market in corporations and elsewhere for custom applications built from application components, it should be obvious how important *structure* as a component of those custom applications becomes. Assuming we can achieve what I describe for reuse of structure, or componentization of structure, then we are really close to a complete platform for custom applications. We have the containers, the content, and the programming language and environment to put them all together. And we have "best of class" in all of these categories. We have the best platform for the development of corporate vertical applications, and have expanded what can be done in this domain by great leaps and bounds. That is, of course, the success scenario.

If we step back from this description of the new world order and look at what it means to do business, it becomes clear that the basis of competition in this custom application market is quite different from what it is today. The software companies that succeed will be the ones that provide the best *platform* for development by corporate IS departments. And that means containers, content, and programming. More succinctly: the basis of success for Office in the corporate market in the very near future will be its ability to function as a full fledged environment for vertical applications development. This *must* be the focus of integrated Office if it is to succeed in this paradigm shift. And if we don't succeed at making Office the basis of this development environment, then we miss the paradigm shift, and you know what happens then.

I don't even begin to address what this might mean to packaging, pricing, and marketing of our applications wares, or what this might do to profit margins for various technologies in various markets. I doubt that the simple approach of selling individual containers, content, and environments quite cuts the cake. More thinking is needed here, and it's really important stuff.

So how important *really* is it for us to actually pursue this "other half" of componentized software, the reuse of structure? Can we continue to succeed under the existing architecure and business model as a company? I believe that this work may very well be the cornerstone for our success - or failure - in the paradigm shift which Cairo is creating.

It's incredibly important to understand our competitors as we think about this. I strongly believe that in the Windows and Applications businesses we really have ONE competitor: Lotus+Novell, viewed as a single entity. Yes, Novell is a competitor in their own right, but in this paradigm shift, they are not by themselves capable of participating fully in the paradigm shift, and thus competing with us. Lotus, on the other hand, is absolutely capable of competing with us on their own without Novell, via their applications and Notes. Novell + Lotus is by far the scariest combination, and I find it terrifying to think of Notes on Netware with Ray Ozzie telling Drew Major what he needs to do in the Netware file system and services to support Notes and Lotus Apps. It's not a big leap to think of a Netware/Notes bundle on the server, with corresponding bundles on the client of a unified Netware/Notes client runtime and Lotus apps. They may as well merge as far as we're concerned.

Why is Lotus the key competitor? Because they have great entries in both the container and content markets, and they understand the network (in the person of Ray Ozzie). Notes is a fantastic implementation of a structured container, complete with services necessary to make it work great on a network. They are still an "old model" application, with a strong tie to a primary data type, their note. But, they *know* this, and they understand the power of decoupling the container from the Note data type. And, they are *doing* what's necessary to accomplish this decoupling in the coming releases of Notes! They absolutely will build support into their applications to better support the Notes container model - they *are doing* an equivalent of "OLE for shared collection containers".

In short, Lotus understands the paradigm shift, and are working on stealing it from Microsoft. In talking with both Microsoft applications and Lotus applications people (NOT Ray) at design previews it's just clear they understand what's going on at a level way beyond what our applications people understand. We should not judge them on their current implementations (e.g. SmartSuite), any more than we would judge ourselves by current implementation. We should judge them based on their capabilities, and man are they scary. We are absolutely NOT in a position for a sneak attack paradigm shift here. This is a race, people are figuring out the rules, and they don't all work for Microsoft.

It's interesting to note that the thing Lotus is missing that we have is VBA. I assume they are working very hard to fix this. But for now it is a key competitive advantage, a strategic asset, that we should be exploiting the hell out of!

And a final note - you can probably guess who is likely to be most ready to address the programming tools market needed for professional development of component applications. God forbid Borland should hookup with the Lotus/Novell world. We're definitely making progress in our languages and tools products, and Cairo is working to develop it's first generation tool set, but I can say with great certainty that the model described here is not the model on which our current tool's business is based right now. More thought required here as well.