[illegible]

To:	Bill Gates, Mike Maples, Paul Maritz, Brad Silverberg
From:	Jim Allchin
Date:	8/21/92
Subject:	Systems: State of the Union (Chicago/Cairo and more)

1. Preface

I feel I have been lax in not articulating in a better way the problems that I think we're facing. I do not have the answers to all these problems, but we can solve them if we understand the big picture and fix some ownership problems.

We face both technical problems and organizational/ownership problems. In a surprising number of areas these two aspects are intertwined. Why? Because one focused team on a problem is the right way to solve it. Dual (N?) (slightly or radically different) developments cause interoperability issues, time spend syncing, fighting over everything from ownership to the recruiting of candidates, etc., etc.

I do not say the following lightly. We are at crisis stage with these problems. We could have accomplished so much more if we had addressed these organization and ownership problems earlier. Problems are not getting solved. People are getting burned out over the stress and lack of ownership. The problem is *much* worse now than when I joined the company.

I will not cover the technical problems here. I want to discuss strategy.

2. To Do List and Dependencies

People now understand the laundry list of all the things that Cairo (and now Chicago) needs. Here's the basic list (ignoring all the distributed system stuff and base OS stuff where there seems to be (generally) clearer ownership):

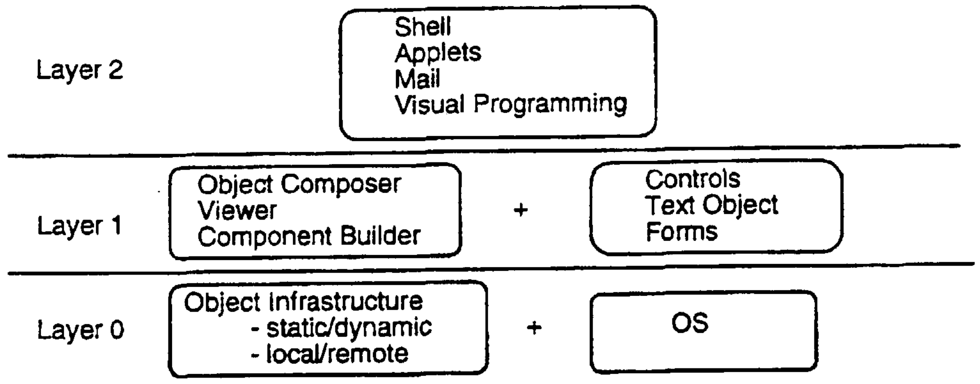
To Do	Explanation
controls	standard implementation of base UI facilities (e.g. buttons)
text object	standard implementation of a text box (e.g., a mini word processor)
forms	standard implementations of controls grouped together which are used widely (e.g., dialog boxes)
viewer/help	standard implementation of a displayer of rich text
object composer	Interface Builder equivalent (e.g., used to glue objects together)
component builder	the environment for creating components to be available to the object composer
applets	shell applets
mail	the client side integration with the shell UI and object model

visual programming	the "scripting" language for the shell. This is the end user's tool for making agents for automation of tasks, etc.
shell	the shell should use the object composer in order to build many of the parts of the UI (e.g., property sheet customization).
object infrastructure	*the* foundation for all of the above. It's the model and the implementation of how the binding, activation, static/dynamic, local/remote, etc. aspects of objects work in the system.

I've been asking for resources and trying to get attention to get people working on some of these for Cairo for some time. (I have not been alone -- edwardj/stevem and their people, for example have both worked hard to educate people on the need for Cairo support.) It has been a major uphill battle.

Since the off-site I have seen much more attention on trying to address these. Unfortunately, what I'm currently seeing is Product Group by Product Group proposed solutions. (This should go to Chicago, this should go to OB, this should to Cairo, etc.) This is not the right way to make the decision for two reasons. First, the above list has a natural dependency relationship which should be used to group people together. I don't think this has ever been discussed. Second, I question whether our current division of Systems "Product Groups" is correct. (I will return to the second point later in this paper Section 4.) I want to focus on the first point now which is true regardless of any decision on the second point.

Here's my cut at the dependencies:



Let me explain this chart. Things boxed together have a natural synergy. If these functions are separated, it will make the design *much* more difficult. Frankly, I think we may have a mess. We could discuss whether design could be separated from development. Depending on the box, it may be possible, however, I do not recommend it.

Each layer represents a conceptual separation where dependencies are from higher layers to lower layers. Finally, each layer also has dependencies. I have grouped them left to right. For example, the Object Composer requires the UI controls, text object, etc. in order to be useful. That means that the shell is closer to Viewer work than to the UI controls work. (Breadth first walk of the tree.) They don't have to be done by the same group, but there is a strong tie here. It will be *much* harder if they are separated and not under the same technical manager. Having different layers owned by the same group is the least important issue. However, the layers juxtaposition means there is a dependency, so a closer working relationship is required.

Conclusion? We should group projects together according to this chart and establish a management owner for each box. (For example, visual programming should not be separated from the shell UI work.)

3. One Owner

I can't emphasize enough the problems with having multiple (or unclear) owners for things. The result is massive frustration and very slow progress. (It is a committee based approach.) I can name many examples in the company today:

- OLE/Cairo
  Who *really* owns the object model? Who makes decisions about versioning as a case in point. OLE/Cairo work closer together than most groups in the company. For example, Cairo did the DocFile design and development for OLE 2. Both groups have worked very hard to ensure that we are compatible, etc. The point here is that is has been very painful and slow. We have burned many engineering hours after the fact once we've learned one of the groups had gone done a different path. This is very inefficient.

- Forms: OB, Access, AFX, etc.
  Unless something changes immediately the shell won't be using any forms. Property sheets, etc. will have to be built by hand and there will be no modification architecture for them. Whether the problem here is lack of ownership or lack of delivery is unclear. I personally don't even know who to hold accountable.

- Object Composer: AFX, OB, ?
  App Studio isn't close to what we need. It doesn't follow the object model of Cairo/OLE and is dependent primarily on MFC (it does support VBX integration, however). However, App Studio is a begining. But, both OB and AFX may believe they own this problem. (I think that the Object Composer should *not* be tied to OB. It should be able to tie together a component written in any language if it follows the object model. However, no one has really made this clear.)

- AFX/OLE
  Who owns the C++ definition of the system API? Should OLE define C++ interfaces or not? What is the role of the AFX group if they do?

- NT/Cairo
  What is the role of NTFS vs. OFS? Why are we building two file systems?

- etc. etc. the list is very long.

Let's look deeper at two areas: the object infrastructure and the shell. The object infrastructure has been split between OLE and Cairo. It has been a monumental effort to try and keep these in sync in two different groups split across the campus. It would have been so much more efficient to have everyone in the same building. Being in the same group would have been that much better. It has been clear to many people that OLE was attempting to define the new programming model for the Operating System. They were not in sync with the history of Windows. I don't understand how that group can be separate from the OS/Systems group. Remember OLE has two levels: infrastructure and applications interfaces. I am specifically separating the infrastructure from the applications interfaces; the app interfaces should remain within a central group in applications where the interfaces are agreed to by the groups doing the code sharing. Bottom line: Cairo and OLE infrastructure should be in *one* group.

The shell has been hard because it has been challenging design wise -- no excuses here. However, a fundamental problem the group has had is the buy-in required from *N* groups. This situation has recently improved greatly with Chrisgr having more respect from the Application product groups. Before this, we had to bring together (and we did!) the different program managers from Apps. What a way to try and make decisions! It wasn't clear who counted in this process -- APPA or the individual product groups. And, furthermore, did certain product groups count more than others? Of course, we can't forget Tandy's group. Things here also are working much better recently, but the fact that Tandy's group did not have a clear charter and ownership left another group for the shell people to deal with. It wasn't clear who counted again.

Conclusion? We need to be *hard core* (absolutely ruthless) on who owns what. Clearly, many people want to own things. They all cannot. We then need to move physically everyone associated with the effort into one group headed by a strong manager. This will ensure high bandwidth communication and maximum focus on the problem at hand.

4. Systems: DOS, NT, and Windows

We are organized wrong in Systems to accomplish what I understand our goals to be. I believe the organization is not only inefficient and frustrating, but it is on the verge of lying to the marketplace. Even though not exactly the same, it is still very similar to the OS/2 disaster. Here's why.

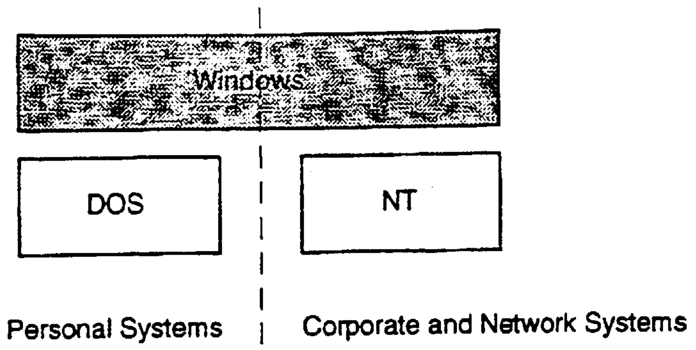
Basically, we want two things. First, we want to tell the marketplace that Windows is the product and we have two kernel implementations: NT and DOS. We want everyone to focus on Windows. But, we don't have the groups organized that way. We have them kernel organized.

Consider APIs. We certainly did the right thing defining the win32 API so that it would be common on both kernels. The reason that worked was that *one* group (NT) owned defining that API set. (Even though it required close coordination with Windows 3.1 since they added APIs.) I have assumed that Cairo owned the central coordination for defining that API for the next generation. On our current path however we will have both the Chicago and Cairo groups defining APIs with people not even working in the same building. Again, *one* owner needs to be defined. But, it's a lot more than the API here: it's the UI, the end-user programming model, the mail integration, the macro language, etc.

My point is simple: these higher level features make up the product -- not the kernel. We should not separate this functionality if you want present a consistent model to a programmer or end-user.

Second, we want to eventually drive to one kernel. (It doesn't matter which kernel.) The current product group structure will ensure that will never happen in my opinion. Why? It's just natural that each product group will continue to enrich their products so that driving to this will be impossible. The current situation is very concerning. We our beating the drums about NT, NT, NT. (Remember OS/2?) We are telling everyone to write new device drivers, transports, file systems, etc. etc. At the same time, we are quietly working in Brad's group on improving VxDs and changing the structure of the system so that we'll want Novell, Banyan, etc. all to write to the new model here also. There is a new IFS mechanism, new VxD support, etc.

If NT's advantage was great, then this strategy might be OK. But, what I see is a set up. When Chicago comes out, the advantages NT has over DOS will be few: portability, security, and SMP. Very few people are beating down Microsoft's door for these. If the product groups were set up differently, then perhaps this would be fine and it wouldn't matter as much how confused we are about the kernels, but today we have a structure which is *kernel focused* not *Windows focused*. This kernel (and organizational) focus leaks over into the Windows design focus. I don't buy that there are such special requirements about the DOS version that aren't true for the NT version. Performance is performance; small size is small size; ease of use is ease of use. Everyone wants these. These are typically poor market segmentation vehicles. Nor do I buy that Cairo has different requirements than what is required in the DOS version.



Today, we have chopped our baby (Windows) up. There is no central control of changes. I just noticed that NT added special support for admin controlled program groups. I don't know, but I expect that Personal Systems has thought about this for Windows for Workgroups and I doubt if there was much communication about the change. Today, the effort is set up so that the NT group is basically building a new kernel. But they continue to slowly evolve the UI. As we move on to Cairo we have to decide where the central focus of the design is. Should our baby be divided into across product groups? The design focus should be driven by a *Windows Product Group*. And all the people who deal with the UI, programming model, etc. for today and tomorrow should be in that group.

Where is the vision of Windows coming from if we divide up the respnsibility? It was on paper the Cairo group, however, we were not organized in order to make that succeed after Window 3.1 shipped. If we divide Windows up along kernel lines, I can assure you we'll never have a consistent model. We'll have duplicate people in the different groups repeating/competing between them for features to be added to the product. Unless we name a single person for the vision there will not be one.

I should point out that at Apple the Mac group apparently has the structure that I'm proposing here. They have several different kernels that they deliver on with slightly different requirements (just like we do). However, there is a central UI group, programming model group, etc., etc. That group is the Mac group. They are much better organized than we are in my opinion. I realize that this may appear to be radical. However, you have not been in the trenches trying to get focus. This would force focus on the right things. There would be no confusion for applications on who they should talk to. Further, our story to the market place would be less confused.

So how would we implement such a strategy? One person needs to be named head of the Windows design. Everyone dealing with UI, programming model, etc. should work for this person. The kernels could remain separate groups. I think we have some options dealing with Windows marketing. Probably the right way to do it would be to combine all Windows marketing under one person. This would ensure that consistent messages are given to everyone. There are some differences in the positioning of the different Windows systems that could be addressed in a number of ways either within a central group or by having it segmented. One of the biggest problems with a strong segmentation is that I think Windows and Windows NT will overlap segments a lot. (Of course, having them in separate groups today, we will guarantee confusion.)

The DOS group would continue on shipping DOS. This makes sense because we already have everyone together in one group who designs and markets it.

Don't think that this would overload the head Windows design person. *Someone* must own putting together a coherent, consistent Systems story. If we separate it, all it does is make the problem harder. (That person cannte be Paul because we are simply split across the wrong divisions. Paul would end up trying to rationalize the products.) Frankly, we will compromise every step of the way and the Cairo vision will not be implemented.

I am not the only person who thinks our current approach is incorrect. So why don't we change it? One interesting comment that came from JeffH was that steveb told him last fall that we couldn't because brad, paul, and

I were all senior guys who needed to run a big chunk of the business. This is not necessarily a good reason. It certainly isn't a business decision and it wouldn't be the one that I would make.

5. Conclusion

1. We need to organize around functions that have interdependencies.

Even if everyone disagrees with my analysis of the Product Group definitions as we have them today, it is important to address the dependency issues I raised. Assuming no Product Group changes, then I would propose that Stevem own the Shell grouping. Jeffh own the Object Composer grouping. Someone in Brad's group should own the Controls grouping. And Edwardj should own the Object Infrastructure. I would propose that stevem, jeffh, and edwardj work for me. I would move asmusf (from NT) and gregw to brad's group to work on the text object. I would move bill mitchell and chris westin to the address visual programming. Neilk would continue to work in the controls areas. (Additional help would proabably be required here.) I would assign david stutz to the Object Composer area. I would move Bob Cooke to work on Applets. I would move Chispin Goswell to work in the Object Composer area. I would raid other areas as necessary to staff the Component Builder area.

2. We need to address the Product Group problem we have in Systems.

Doing (1) above is not optimum at all. We really haven't faced up to the real problem if we don't reorganize the Product Groups themselves. We have several choices:

(a) We put all the "Windows Product Group" into one of the kernel groups. Clearly, Brad's area seems the most likely. I should then be reassigned: another division (e.g., tools & db) or ???.

(b) I take over the Windows area (design and marketing) with Brad owning DOS (development and marketing) and NT development.

(c) I take over the design of Windows. Brad owns the business side. NT and DOS development could be handled in a number of ways. Either I or Brad could pick it up -- or even maybe Davec should pick up development ownership of both.

(d) Brad takes over Windows (design and marketing) and I take over the DOS and NT kernels.

I believe my lost from the future design of Windows would be great. Therefore, probably (a) and (d) don't make as much sense. It is unclear that (b) leverages Brad and may overload me given the design issues that we face. That leaves (c). Maybe there are other options. One thing is for sure, I'm convinced it's broken pretty bad today. It's not that Microsoft won't come out with a good product if we leave things alone. From where we are today that isn't too hard. But, we could create a much better product, faster, with less frustration by changing. I see the change as inevitable.

This situation isn't that much different than the issue of having NBU separate from the operating system (NT or OS/2). That organization of Product Groups was also incorrect. Networking needed to be apart of the OS group. Our products will be *so* much better because we made that decision. It was painful at the time, but the result will be an order of magnitude better networked OS than having separate Product Groups. There is little difference with the Product Group mismatch described here. It's the same problem. We will fail both to build the best product and we will confuse the marketplace because we don't have a central focus.