Erik Stevenson

From: Jim Allchin
To: Mike Maples
Subject: FW: win/x
Date: Saturday, February 20, 1993 10:36AM

fyi,
jim

From: Darryl Rubin
To: Bill Gates; Jim Allchin; Paul Maritz
Subject: win/x
Date: Friday, February 19, 1993 16:13

Here are my thoughts on the win/x proposal. I'll comment separately on technical and business issues.

Technically, I love the proposal. It does a much better job than Scottra's original WOM proposal in terms of truly just extending the existing windows architecture in a graceful (and quite minimal) way, as opposed to providing a parallel architecture that just has a windows look/feel (which is how I'd describe the WOM proposal).

As is well recognized by now, existing OLE is a parallel architecture that doesn't even have any of the windows look/feel. This has resulted in a massive reinvention of the wheel, in terms of duplicative conventions for memory management, notification, registration, etc. The fact that UI elements like windows and controls are in a different world from component objects has also created a lot of ugliness and confusion. The win/x proposal solves all these problems and I see no defects in the design approach. It is the most rational design document I've read in at least the last two years.

I'm enthusiastic about the proposed type model. This is much better than the interfaces model. Dealing with interfaces has proven to make coding much more cumbersome, especially due to the need to do queryinterface calls all the time and to reference count the interfaces. I think win/x provides sufficient type safety. If there is a tradeoff, it is just that Microsoft must administer ISV id assignments, and each ISV must internally administer type and message number assignments. Administering message number assignments within a large organization is the bigger problem, although it is not a killer.

Another thing I like is that the win/x model is more language independent than OLE. There's no developer penalty for using C (or whatever) instead of C++. OLE is very C++ oriented.

The proposed simplification to monikers is good. I had suggested basically the same idea to Gregw when monikers were first being designed, but the OLE team rejected this on the belief that having to conform to the system pathnaming syntax would be too constraining. I don't see that as a problem and think that the win/x moniker scheme makes dealing with persistent objects much more comprehensible.

Another possible plus of the win/x proposal is that it's easier to see how we cram component object capabilities into our very low end (mobiwin, winpad) environments. Right now OLE seems too big and complex for that.

Regarding concurrency...I agree with the statements about the problems being the same for messages versus rpc's. However there is also a consideration at the application programming level. The windows messaging model is more complex to manage than a purely procedural model because once you've bound two objects messages can fly in either direction at any time (actually, messages can even come from objects you haven't bound to but let's ignore that). This is just a way of saying that windows is an event driven programming model, which the interface model is not. To a large extent this should be a non-issue, since windows programmers have by definition already learned to deal with this programming model and win/x is just talking about leveraging it. The question to be careful about is how much harder does it get to implement under the event driven model when the number and kind of objects an app is connected to gets much larger. This probably isn't a killer but it could use more thought.

Another consideration related to this is that most of windows programming today deals with lightweight, runtime only objects that exist in a strict parent/child relationship. You don't really have to worry about

keeping track of who's talking to who--you can pass hwnds around without worrying about binding, ref counting etc. However, with component objects it's a greater concern for the object itself to understand when it's still really needed since it's implementation and the resources it consumes (and the persistent resources it's connected to) are opaque to the apps that are using it, and there will be broader sharing of objects among multiple callers. The win/x proposal doesn't seem to address this, but that shouldn't be hard to take care of.

Clearly, if this was three years ago, there'd be no question in my mind that win/x is the way to go, and the result would have been a much faster exploitation of OLE-like capabilities. (We might even have solved the forms problem by now!) However, should we change now? This leads to the business issues.

We've been able to win with our Windows apps partly because we've been the leader in exploting the systems platform. If you look at us today with regard to OLE, this is still true. Microsoft is way ahead of anybody else in leveraging OLE. Of course this is largely due to how complex OLE has proved to be. The result though is that we're very far along in exploiting it and very few other ISV's are.

The possible gotcha in switching to the win/x proposal is that suddenly we may find ourselves far behind, instead of in front of, our competitors! This is because few of them have deeply committed to OLE coding, and so for them win/x creates a new, simpler alternative for implementing OLE-like features. They can just get started and do it. We on the other had would be set back as we tried to redirect our coding efforts, and also did the extra work to make our new win/x support interoperate with existing OLE-supporting app versions.

A big question would be, if we did win/x, would we try to make the change for chicago, or just cairo? If chicago, then we derail other very OLE-dependent projects, including the shell, and mapi. If we do not, then we establish this very high volume platform that ships with less than a year's delta from cairo, but whose whole architectural underpinning from almost the system level all the way up to the shell is OLE rather than win/x.

Another thing to think about is ISV and customer perception. We are coming into the timeframe when we may face competition from alternative object-based systems, and in a somewhat different sense, from Notes. Redirecting our architecture could cause a lot of uncertainty about how stable our direction is. A lot of energy would be focused on understanding the differences between the old OLE and the new win/x, which could detract from commitments to implement either one. The door could be opened wider for people to look at competitive solutions since as long as people are now back into evaulate things mode...It's even possible that some isv's and the press would portray this as yet another piece of clever misdirection calculatingly foisted on isv's by the evil empire.

A basic question we need to ask is how much will the ugliness and complexity at the heart of our system impede developers, especially versus the possibility of a system like Taligent hitting the market which in the worst case for us turns out to be beautifully simple and elegant? While I have no doubt that the complexity has already impeded a lot of development and will continue to do so, tools continue to emerge that hide the underlying grunge (VB, AFX, other kinds of high level wrappers and development aids). The OLE/Windows dichotomy will probably become moot over time to most corporate and vertical market developers. It is mainly the hardcore horizontal apps developers who would continue to suffer the pain, and in certain ways that's good for us near term. Where it's not good for us, of course, is in the additional design and development inefficiency it creates for us long term, and the risk that the hardcore developers get attracted to a much simpler alternative platform.

I must admit I'm very torn between the choices here. Technically, win/x is totally the right thing to do and I think a switch would bring a huge benefit in terms of making the system smaller, more rational, easier to develop for, and easier to implement good general purpose tools for.

On the other hand, such a switch could set back both systems and apps schedules enough to lose critical market timeliness versus Taligent, Notes, etc, and it could stall ISV commitment to our new platforms while people try to sort out what it all means.