From: Yuval Neeman
Sent: Monday, January 08, 2001 5:55 PM
To: Yuval Neeman's Direct Reports
Subject: FW: Software Agenda

Attachments: SOFTWA~1.DOC

SOFTWA~1.DOC
(56 KB)

---- Original Message ----
From: Jim Allchin
Sent: Monday, January 08, 2001 7:43 AM
To: Brian Valentine; Paul Flessner; Mike Nash; Will Poole; Yuval Neeman; Dan Neault
Subject: FW: Software Agenda

Please make sure you read this and distribute it to your managers.

Although it isn't completed, it is worth reading, commenting on, and planning for.

jim

---- Original Message ----
From: Bill Gates
Sent: Thursday, January 04, 2001 6:22 PM
To: Senior Leadership Team
Cc: Rick Rashid
Subject: Software Agenda

This memo is one of the ones I have discussed doing.

It is not the Roadmap memo.

Rather it is my view of the key issues.

This will be a live document. Some of the points are incomplete as you will see.

We will have some time to discuss where I should focus more attention during the SLT meeting tomorrow.

Software Agenda

For companies like Oracle where they have a single technology and a focus on a single customer set its easier to understand what their priorities are Microsoft is building a much broader platform for a broader set of customers. Our strength comes from the singularity and popularity of the platform. Even we can't afford multiple overlapping messages especially when developers are moving to Linux and Java. Our platform pieces have to solve major problems for customers and be extremely popular.

This memo is an attempt to draw up the list of areas where progress can make a big difference so that we can focus on these and track our progress.

In addition to improving this memo a complimentary Roadmap memo needs to be created.

Software distribution

We need to be able to go to endusers and corporate customers with a service for keeping their software up to date. For consumers I believe this should be included in our low cost subscription offering which you get 3 months of free membership in when you update Windows or get a new copy. The service needs to federate through the corporation so they can control what goes out to their end users. Allowing updating to be federated will help us know how to federate other elements of the system which contain Microsoft URLs or connect to our services (help, crash dump, IM, custom home page, SIP communications, remote storage...). Federation means giving IT a way of reviewing and controlling what goes out as well as using the infrastructure for their own software distribution. We have many formats and approaches to software distribution today. The question of when "setup" gets done affects distribution. SMS should go away as a separate product as we provide this service as a part of Windows. Windows update needs to move to sending deltas instead of whole files. We need the Drizzle feature that Mars provides. In many cases we need to move away from DLL replacement to patching using the PPRC technology. Fusion needs to be embraced broadly. Windows Update has to be a lot less intrusive to the user and a lot richer in content. Corporations have to feel a real loss if they are not getting the fixes that only come by belonging to the update service. We need to articulate the vision for software distribution and get ISVs involved as well. OS updates and Driver updates need to share the same infrastructure. Windows needs to pick someone to architect this and make sure it is a central element of our platform. The business benefit of having great software distribution will be huge because it will give corporations a reason for staying up to date with us without viewing every new bit that comes from us as creating gigantic overhead. It will also allow us to up the quality of the products our users experience dramatically. This is the most important reform I list. The Windows group needs to drive the creation of this service for all Windows users and ISVs.

There are parts of SMS which could get moved to a "PC support" overall package (like software inventory) but that would only make sense if we bought one of the PC support packages to sell as an application.

Quality

Microsoft needs to know when users experience crashes or other errors that come from software bugs. Using this feedback loop we will be able to fix problems in our products quickly. Combined with the distribution service described above this will do more to improve our reputation than any new features we will provide.

Both Office (Dr. Watson" and Windows (Bluescreen) have systems for taking system crash information and publishing it to a Microsoft URL so that we can analyze the crash. Already these systems have uncovered a number of important crashes that our normal systems had never caught. Today we have no way of taking a system that is experiencing problems and being able to remotely patch in code to gather more information. Using the work done in PPRC we can add that capability. Today the crash logging systems aren't able to involve a third party ISV or PSS or a customers IT group in the process of gathering all the information and solving the problem – these dumps are coming straight to the product group today. The current systems are restricted to dealing only with crash situations – they should be broadened to support general error conditions including having the application call on these services when something is wrong. These crash handling systems need to be changed to work in the unattended server environment.

We need to have at least 100,000 clients and 10,000 servers where we totally support the machines to have a complete profile of problems. As part of our subscription offering being able to log what goes on on these systems, back them up and solve any problems that come up will guide our work. We need to understand the costs and complexities users experience with our systems.

Beyond the "error" feedback loop described above we need an overall focus on quality issues including someone full time who looks at the business, partnership and technical issues that could significantly improve the situation. People like the specialized publications and Intel could play a role in helping us measure what needs to improve. For example a great feature of Whistler is the ability to connect up with someone else in order for them to help support you including letting them take control of your machine. We need to fix the firewall problem that stands in the way of all PC real time scenarios for this feature to see its full use. We need to continue to evolve this capability so that support costs for PCs go down significantly. This feature needs to work for ISVs and IT. Another key element of quality is to make sure that buggy third party code creates less problems for our users. Central to this is code signing and forcing addon code to go through a testing process. Our update service should be able to notify users that a piece of code they have installed has had its certificate revoked. An ongoing jihad here is critical.

Productivity

The man-years of work per feature created at Microsoft isn't good enough. Good architecture that leads to powerful shared infrastructure can help with this but there are dramatic gains that need to be made beyond that. The total number of bugs we are creating and then having to track down slows everything down. Even a 10% improvement would make jobs more interesting, cut costs and allow us to get higher

quality more powerful software to market quicker.. Potential advances can come from tools, processes, and source language. The recent move to SourceDepot was a big success and reinforced the importance of great tools. Likewise the Prefix work from PPRC has been extremely beneficial. Still our tools overall are poor. The tools we use internally should be a pure superset of those we offer to the market. Our source codes should be structured XML documents that allow specification and code to be more closely coupled. Part of the PPRC vision is that navigating information related to execution and testing with the source code should be easy. Source codes should be easily accessible online. Part of Simonyi's vision is that the source language itself should be extensible so that domain specific constructs are simple to create and understand. User interface's should be so easy to create that you create them and then document them. Ideas for improvements will come from PPRC, Simonyi's work, Research in general, Linux development, other outsiders and parts of the product group including some initiatives in David Greenspoon's area.

I think VS8 should embrace the idea of source code as an XML document with an environment that combines the vision Intentional Programming and PPRC have suggested. This will be a huge advance for our customers as well as for internal productivity.

We need a specific agenda in this area – things we are implementing and things we are exploring. We need regular checkpoints for discussing best practices and new ideas.

Openness

Our most potent Operating System competitor is Linux and the phenomena around Open Source and free software. The same phenomena fuels competitors to all of our products. The ease of picking up Linux to learn it or to modify some piece of it is very attractive. The academic community, start up companies, foreign governments and many other constituencies are putting their best work into Linux. Although we cannot make Windows free for commercial use we can do dramatically more to make it accessible including parts of the source code. We can make it free in restricted areas. One important idea is to be able to source debug any running copy of Windows by connecting up to an Internet hosted symbol table with PPRC technology allowing you to patch the code. This means that you don't have to get the entire source and learn how to build it to debug and add on to Windows at a source level. Although some parts of the source would have to be made opaque to the general public, some of the source could be there for all people and all of the source could be there for some people. We need other creative ideas to allow Windows to match the viral nature of Linux.

Storage innovation (including caching & replication)

The Microsoft file system and related protocols have stood still for almost a decade. During this time HTTP/URLs/DAV and Email have become important competitors to the file system. Users are very frustrated that even the most basic property query capability is not available for files. Even inside Windows itself we create special stores for music, fusion, passwords and every new object that needs property based viewing. Office has created a storage system called Sharepoint based on HTTP. MSN created a free "storage"

service called MSN communities. Office has had to abstract away from just using our file system APIs because Windows did not connect out to rich enough servers and so Office embraced some form of DAV. IIS is a kind of storage system with its own naming and security and code invocation capabilities. Our customers are finding it easier to install storage appliances than Microsoft servers. Storage systems are being abstracted away from the application servers. Both our servers and CAL revenue are at risk.
The best solution is to pursue an architecture that is over a decade old and that is to make the storage system richer – rich enough so that email, music and things like printers, fonts, etc... are able to be queried and stored without using special applications for each type. This means taking our next generation SQL technology – Yukon and making it the next generation file system.

The new strategy will only work if we get a class of applications that take advantage of the new capabilities of the store system. Capabilities like basic Document management should be intrinsic to our new file system. Serving up bits over HTTP and serving them up over SMB should not be different. The name spaces and ability to invoke code should come together. HTTP listening and HTTP efficiency are key capabilities – one listener and one talker.

We need a vision of how storage semantics and systems will evolve. What is our response to huge SAN networks of disk supporting the commodity protocols? Will we make Windows systems easy enough to set up and price attractive for these markets? What kind of high end features are we missing?

I have pushed the Windows NTFS group to think of Sharepoint as a storage system that they should ship and then work to provide something that is integrated and better than NTFS alone or Sharepoint alone.

There is a question about Yukon subsets. Whistler+1 should probably align all of the various system stores to a subset of Yukon like DirectDB. We also need a compatible subset for PDA devices. Blackcomb is the release that has the Yukon file system active once it is booted and is used to simplify the interaction with all information on the system

The future of storage systems has to be considered in light of the intelligent caching that will be present in the Network and in the client devices. There is no reason that Geo-caching should apply to HTTP information and not to the Windows file system. Our whole Message bus strategy should apply to caching file system information as well as generic messages so that the work we do on tagging, local recomputation, and push/pull are shared. We should be the leader in defining the new protocols for caching, cache invalidation and reporting usage of cached information. Our client level caches give us the ability to influence Websites to do things our way.

Making replication of information is another critical goal for storage reform. One of the key scenarios where the PC shines is in offline use that is not going away. Today the user is forced to manually do the replication of all the information they might want offline. The interface is different for Mail, Directory, WebPages, Files, and Code. For files we

have Briefcase and Intellimirror. Windows2000 brought the different replication commands into one place but that reinforced the problem. Our file system should treat caching of information – all information – as one of its native capabilities. We will need a short term plan for replication advances as well as the plan for Blackcomb. We have to make it easy to move between PCs either whether or not a user connects to our services or not.

The new storage system has to run as a service in the cloud at very large scale including hooks in Windows to make it easy for subscribers to store data. Even before we get advanced storage we need to provide a rich replication service between all our devices.

Collaboration/Workflow on SQL

Strongly related to storage reform is having email show up as an item in system storage. We have already taken the step of combining the Exchange and SQL groups with this goal in mind. Yukon is being influenced by the requirement of hosting the next version of our email server. There is a key question about what different servers we should have on top of Yukon for Knowledge workers. The Windows File Server needs to use Yukon and offer much richer features which subsume Sharepoint. We will continue to charge for email capability which we need to enhance with Gmail capabilities as discussed in the subscription memo. Unclear is whether Workflow or Portal Servers are separate and what access is paid for by having an up-to-date Office license. I believe we should take the Biztalk platform and use it for Office workflow but this requires a visual front end. The technology needs to relate to our Information Agent work.

The separation of Office from our servers has held us back in solving problems for knowledge workers. No matter what the organization structure is Office needs server support for rich Document Management, Workflow, Conferencing, Gmail and other knowledge worker scenarios. We should also combine the efforts to have a rich environment for Team source code development with the desire to have a Vignette like high end production tool built on these servers.

Whatever servers we create we will want to operate them as a service for customers signing up to our Office.Net service.

Authentication and Directory

We need to move over to SQL as the store for directory and metadirectory as soon as we can. We need to have security ACLs they aren't dependent on Ids which are tied to the hierarchy chosen for names. Companies should be able to move users around in the hierarchy whenever they want. By using SQL our tools will work with the directory information. We need to articulate the importance of a Metadirectory and take leadership in having connections to all the applications that use directory information. There will be a lot of key schema work that we need to do in partnership with other leading companies to make this work. We need A.D. to support public keys and allow for federation with Passport. One of the systems should be renamed to have the same name as the other. Our metadirectory should be usable independent of the OS it is running on for intranet scenarios. The Directory team needs to work with Yukon to get the data model and

protocol issues resolved so that the semantics for authentication replication can be done without special set up. We need to make sharing of all kinds of information across corporate boundaries easy by federating with Passport. We need to make it easy to authenticate the sender of information the same way. Directory design should not be a fragile and gating item for the deployment of Windows systems. We need to make sure our authentication system allows for signed code. We need to eliminate vulnerability that the way our admin account passwords are handled creates. Our customers should not have to go to third party solutions. Our certificate architecture should be aligned with our ACL/Directory/PKI work. Customers shouldn't have to buy add on security products like Entrust. The Windows group needs to drive a roadmap for this big change in how we do Directory.

We should be able to charge extra for Metadirectory and the Federation services that allow the Extranet services to work through Passport.

We should have Office and H.R. Apps vendors work with our MetaDirectory people so that the common scenarios relating to finding out about employees provides a great front end experience. Today trying to find out who works in what part of the org or where they are located or what their backround is far too complex.

Management/Setup

Provisioning and monitoring Windows systems needs to be far easier than Linux systems. Our management infrastructure has to use the eventing/logging/filtering APIs that we are defining with the SQL Message bus. The vision of the BIG group where you can describe a set of systems to perform a task and then monitor those systems through an XML document is key here. An abstract description of a server should allow a service to fully provision the server. Customers respond favorably to the idea of having ASP resources available on demand for peak loading and disaster recovery. BIG and AppCenter and Management to me are really one set of problems. By using all the SOAP/XML protocols and the Yukon infrastructure we will be able to have an extensible management environment. Windows has to work in a headless environment including reporting crash information over the network to enable the automatic provisioning scenario.

One of the reasons that appliances are so attractive for dedicated functions is that Windows is too hard to setup for specific profiles. Although for high visibility profiles we should have special packages it should be very easy to choose a profile like "Load balancing" or "RAS" and a few parameters and simply ask the server to be setup. We need to merchandise how this approach can be more flexible in terms of staying up to date and providing flexibility than a pure appliance approach.

In the past we tried to do setup and monitoring using MMC. MMC wasn't a model – it was simply a shared display service. The Setup and Management Console capabilities need to be put onto the XML runtime as that gets done so that we get rich viewing and navigation. I don't know what interim runtime the Management people are trying to use – this should be discussed.

Some level of management needs to ship with Windows and some level can be extra charge. We need a rich framework/schema around the Yukon infrastructure that third parties use to add management capabilities. Tivoli excelled as a framework with third party value added before they actually shipped rich functionality which is a model for what we need to do several parts of management. We should make sure firewall issues don't make it hard for people to manage their systems across the Internet. We should consider offering Remote management services under the new framework.

Leadership in management will be measurable from customer feedback, revenue, and comparison with other high end vendors. We have invested a lot in this area over the years but our solutions have suffered from having their own infrastructure. Windows cannot be a strong server product without having excellent management.

Presentation reform

Our "presentation" layer today consists of a number of disconnected technologies including DirectUser, GDI Plus, DirectX, Trident, PTO, VS Forms, Webforms, Ebook, Office SDM and Forms3. An innovative presentation layer is critical to showing off why local application execution provides a better experience and keeping the PC vibrant versus dumb devices. We want most applications to download to PCs and run there receiving XML data and XML SOAP calls over the Internet. A rich presentation environment that allows forms and UI to be specified and easily edited and allows XML data to bind to each is critical. The lowest level of our Presentation system should present a single driver interface and allow exploitation of new graphics chips. Turner Whitted in Research is pulling together thoughts on what requirements this creates.

A good design for this system will allow us to layer in PDF compatibility including an annotation layer. A good design for this system will allow us to build an animation layer which is competitive – today Flash is playing this roll. Video and Audio will play a major role in upcoming interfaces and they cannot be relegated to just run in a player application. Also screen updating should be done more smoothly using offscreen memory

I want us to be able to construct rich UI easily simply by editing an XML document. I want us to be able to bind to XML data easily using the new forms environment. We have compatibility issues to consider with all of the above systems. Trident will be a piece of the solution as the HTML displayer but it will likely not be at the center of the system.

One important consideration is to make sure that display remoting using our proprietary WTS protocol/Netmeeting (same or different?) continues to work for applications with extensions to support high quality audio and video.

There are a lot of compatibility constraints but also a need to innovate in this level of the system. A key question is: Can we make applications look better if they use the new approach?

Its unclear to me how much symmetry we can have with the server based forms model – Webforms. Ideally we want XML payloads to rich clients but developer's don't want to think through their forms logic twice and completely duplicate that code.

Our strategy is to make standards based HTML look increasingly obsolete and not have to give away the underpinnings of our presentation environment to a standards group.

There is an ongoing dialog between Ted Peters and all of the constituent groups to try and come up with a proposal on where we go with this. HP is potentially a partner for printing related aspects of the new approach.

Applications platform

Our applications platform message is quite confused today. Pieces like CLR, WMI, MSMQ, XML runtime, Biztalk, MTS, IIS, ASP+, Load Balancing, Message bus, SOAP, UDDI and Yukon are not consistent and reinforcing. Basic standards like eventing, logging, and filtering have to be established. The disconnection of these products make our message when trying to win back the developers who like JAVA and J2EE very difficult especially when we have the limitation of being only on Windows and those technologies are supported on many platforms by many companies. Although we have waited a long time for the shipment of VS with the URT that doesn't give us anywhere near a complete consistent platform story.

We have talked about many of these problems but not pulled things together. MSMQ is a bit of an orphan. Our transaction strategy isn't getting any traction while BEA has established an $800M per year business around that technology. We did a good job on MSMQ and MTS but they couldn't thrive on their own. Our decision to make Yukon the center of gravity and to connect Yukon to the URT should give us the clear starting point. We may need to be able to package Yukon so that it doesn't feel like a database if all you want is a Message bus. We may need to create some subset implementations of things like Queuing for size and speed reasons. However the API set should be consistent. We may need to be compatible with some of the J2EE apis.

I think that between Paul, Yuval and Eric's group with some help from Rick Rashid we should be able to go through another iteration on this (like we did with NGWS) and come up with some clear answers.

The strength of this platform and the innovation around it is the key element in preventing commodization by Linux, our installed base and Network Appliance vendors. We are in the best position to define the distributed application model that allows work to be moved out into the Network. We don't have enough research our product group people pushing this agenda but we have the best opportunity. This is what it takes to seize leadership in caching, load balancing and protocols. I think between Management/Setup and a vision of how our platform is Distributed we give ourselves a chance to lead in all the Level 7 networking pieces. I almost included this as a separate item but executing on these two technical pieces will give us what we need except for packaging, marketing and sales force.

There is a major packaging question once we get architectural coherence. To what degree should we package or charge for the rich so called middleware pieces separately from the rest of the platform? Are there advanced forms of some of these pieces that cost extra? Most of the API set we want supported in the base server with understandable advanced services costing extra.

We are discussing with IBM a joint effort to agree on most of the Application server pieces so that companies have a choice of our two implementations. Although this would be an unexpected partnership I see a lot of advantages for both companies. I think they can help with parts of the architecture. The current view is that we do not share any code between the companies.

We also need to drive Microsoft to use the new platform to prove it out and show it off. Our Services need to use these architectures so that our tools make them easy to extend.

PC Excellence

The PC has to have all the advantages of being a simple dedicated appliance without giving up the ability to run many applications and support a variety of peripherals and update the system software. Walt Mossberg and our satisfaction data say we haven't done enough on this.

One critical issue is boot time. I spend minutes of my life everyday rebooting my system at work and at home It has gotten slower as I have moved to new releases of Windows. Just making a big advance in this area would cause a lot of people to upgrade Windows. The Windows team is focused on making Standby and Hibernate much faster including the BIOS piece that requires OEM work. We need to make sure we improve it even a lot beyond where Whistler will be partly with software advances and partly with hardware. We should have a flavor of Boot that assumes not big changes and uses the same logic and de-hibernating.

WinHEC is the forum where we get to send the message to the hardware vendors about what we are focused on. We need a clear message on power management, removable media, microphones, video decoding, graphics, that makes the PC a moving target.

Data Center

Its critical that we be able to handle the most demanding applications – even those that don't partition naturally across many servers. We need to lead with partitioning technology but also make sure our hardware partners are providing single server systems that match everything SUN is doing – memory bus, I/O speed etc.. We will continue to find software bottlenecks in these scenarios. There are a number of software features we still need to match up to Mainframes and Solaris in the Datacenter scenarios. Many of those we can use partners to solve. Key hardware partners are IBM (particularly if they will put x86 support onto their latest chip technology), Intel, AMD, Compaq, HP, Unisys

and a number of Japanese vendors. We have put a lot of energy into supporting Itanium and we need to evaluate at some checkpoints how well Itanium will provide what we need.

Reliability and Manageability are also critical to success here but those have already been mentioned. The Data center does bring special requirements to those issues.

Real time communication

Part of our vision is that communication in the future will be multi-modal. People will have a screen – either a PC or a PDA type screen – and anytime they are interacting with the screen they can use speech for commands or navigation or dictation or talking or voice mail. Notifications/email/filtering. Screen call. Collab. Ozzie-Groove. Multimedia. Capps vision. Screen Call extended – all calls coordinated as speech and data. Milestones.

Asynch Communication

Email client. Storage. Gmail. CRM. Information agent. Stored Annotated meetings. Scheduling. Multimedia. Future of Outlook. A piece of this was already in the subscription memo. Notifications versus IM versus Chat versus Mail.

UI approach/Schema

This is the hardest one to write. I need to put a lot of time on this. I have a lot of thinking I need to share on this.

Office code base

Development platform. Extensibility. Future of Word. Future of Frontpage. Future of Access. Future of Works. Future of Publisher.

Tablet/Ink

Reading/Annotation

Meeting/Learning

XML runtime

(forecasting, management,...) Future of Excel.

Speech platform and Natural Language

Over the years we have invested quite a bit in natural language and speech recognition/synthesis. Recently some people have asked when it will really enhance our products and whether there is a customer/developer need that will be huge around this technology. While some parts of the speech stack we could have developed by partners the "API" that says how people allow speech to affect their application is something we need to lead in and own.

As described in our RealTime and PC vision, speech input will be a mainstream part of using a PC. Likewise typing natural language as part of a conversation (type in line) will be the most efficient interface for many things. This means that all applications including websites will want to be able to describe how to "react" to natural language. For example when you connect to Amazon.Com with a PC or a PDA your commands should help you navigate. Part of Windows/.NET keeping a lead as an applications platform requires us to have the architecture and tools that make it easy for people to language enable all applications. We need to innovate in how these grammars or slot filling or tree diagrams deal with probabilistic input. This includes having the client side and server software that can execute the applications. A degenerate case is where the device is a voice only device. Since Nuance and Speechworks dominate people creating voice only interaction it is natural for them to be strong in websites wanting multimodal interaction. Also IBM, Philips, Lucent and various research labs will like to give away speech technology so our asset needs to focus on tools and API allowing other speech engines to be used.

Speech interaction will change the way applications are defined and finally bring the idea of social interface back to the fore.

I think we can define some scenarios that we need to lead in and track our progress in being the primary platform for language enabled applications.

Summary

It very exciting to see how many cool things great software can do. Things that everyone really cares about – Knowledge workers, IT, Developers and Consumers. G