o Exploitation of EMS, Cairo
o Coord. of WG features into apps

MAC?

. process
other priorities
- schedule

To: Jim Allchin, Jon De Vaan, Tom Evslin, Chris Graham, Roger Heinen, Pete Higgins, Mike Maples, Paul Maritz, Nathan Myhrvold, Chris Peters, Darryl Rubin, Steven Sinofsky, Greg Whitten

From: Bill Gates

cc:

Date: July 23, 1993

Subject: Microsoft Development Synergy Agenda - DRAFT

I believe there is wide agreement that we have not done a great job setting a company wide technical agenda and getting the benefits of our scale. At a recent retreat I agreed to document high level goals to help us develop an agenda. A meta-issue is how we improve the bandwidth of the architecture dialogs at Microsoft to avoid missing opportunities or trying to solve problems that already have a long history. We need to document and communicate technical visions, strategies and architectures. We need to share code massively between our various development projects in order to be efficient and to promote standards in user interface, data formats, and feature sets.

As the company has grown we have found that synergy across Business Units has to be negotiated and orchestrated at a very high level, often with my direct involvement. Unfortunately, the issues and technology are complex enough that it is not possible for me to resolve these issues in an efficient manner. I look to senior management, development, and program management to push forth a common technical strategy. Crucial to achieving this strategy is a process for solving problems that span Business Units, as well as a set of shared technologies and an architectural vision. I would like the issues of user-interface consistency, code sharing, and adherence to architectural synergy to be given priority by product teams. Synergy should be part of performance objectives and should be measured, with the idea that individuals will be genuinely concerned with achieving these goals and will be rewarded for doing so. As we go through the three year planning cycle, synergy across applications and systems will be a key theme. Also, there will be a follow-up retreat prior to the three year planning process.

decision makers

Technically, I believe the "object oriented" approach is crucial to achieve this synergy. Although everyone has a different idea of what this term means, these differences are minor compared to the major shift in thinking it will take for us to obtain the benefits of these techniques internally. Object-oriented means defining architectural quality interfaces across product components, encapsulating the implementation of those interfaces in a reasonable manner, and reusing that implementation across applications. For controls, object servers, and major application components this means using the OLE 2.0 model of IDispatch for automation and Interfaces for programmability. For low-level components these interfaces must be architected using C APIs and Windows messages. I am expecting us to share more and more code using DLLs and OLE servers. There are many places in our applications today where

and CRL

Form

fairly isolated portions of code are duplicated in applications because there was no defined interface or because there was no incentive to share. Sharing code is also important to gaining competitive advantage by reducing the working set of our suite of applications.

We will define a number of these object-oriented "interfaces" and implementations of those interfaces. Examples of these interfaces include: controls, forms (a container for controls), hierarchy viewing, table, query, text handling, and various storage interfaces. Some of these will exist at multiple levels of richness. We need to identify the key set of these objects and assign responsibility so users and implementors know how to proceed to get the highest leverage. It is important to realize that although the power of separating the user-interface from the implementation makes multiple implementations easier to manage, it is still far better to share the same implementation than to rewrite a new one.

This has become an urgent issue. Frankly, our competitors have started to catch up with and in some circumstances surpassed, our ability to innovate Windows applications. We need to be the leader in innovation. We need to send a clear message to developers, including how to build on our applications. Additionally, users are buying suites of applications and have high expectations of how those applications should work together. Our competitors are doing good work that we need to surpass:

- Lotus SmartSuite. With SmartSuite Lotus has shown that it is possible for one company to put together an entire line of productivity applications. They have gained substantial mind-share with their *Working Together* campaign. Their features like SmartIcons and other standard user-interface elements such as the live status bar show that they are serious about corporate synergy.
- Adobe Acrobat. Document interchange and on-line viewing need strategy. I do not think that their approach, relying on the printed page as the common denominator, is the best one, but we need to use our standard setting ability and our architectural thinking to advance this area.
- Folio Views. This product shows the power of on-line document viewing. They have done some interesting work with navigation and using document structure that needs to be part of our products.
- PowerSoft. A relatively small company has gained significant mind share by selling development tools that cover an entire class of problems. We do not have a good answer for these users. Visual Basic is making substantial headway into this market, but areas such as group development go unchecked.

- Lotus Improv. At the recent Word and Excel retreats one subject came up repeatedly and that was how powerful having a structured view of a document can be. Improv offers a very structured view of a spreadsheet which for some tasks is very useful. Improv has also made a number of advances in charting and visualization.
- Claris Works. The low-end integrated category is one that we owned, yet Claris was able to improve both on features and in the metaphor for working in these products by using a document-centered approach. We need to both innovate this category and gain synergy with our desktop applications.
- Borland Paradox and InterBase. With the combination of a local database engine and a server database engine Borland has a great leverage point.
- Notes. Our inability to coordinate or counter strategy to Notes is the strongest example of lack of technical planning across groups. They are defining workgroup for all leading edge customers.

The boundary between systems and applications should not hold back any of this synergy. Microsoft is allowed to innovate across product boundaries, including joint design and implementations. Microsoft evangelizes ISVs broadly by providing them with information and seeking their input. A close relationship between our applications group and systems will improve the system for everyone. I expect Microsoft applications will help drive standards for UI, shell, integration, data format and programmability.

Forms

Our forms architecture will be the next generation of a Windows dialog, as well as covering classic forms. A form will include OLE controls, as well as standard Windows controls. We will also have a forms editor that will allow the user to wire the user-interface elements to behavior, in a language-independent environment. Our users will want the ability to place controls on all documents and use VBA to attach behavior to these controls. Today we have this to varying degrees in a number of our applications. Efforts of both end-users and our developers need to be shared across Microsoft products.

Rather than have different forms editors and control architectures for Fox, Access, Visual Basic, Works and Mail, we will have a single forms architecture. This forms architecture will be used for building all of the user interface elements of our applications, replacing today's solutions such as SDM. We are in the process of finalizing a control interface designed as an extension of OLE 2. We need to define a form interface which is independent of the language it connects to. We will have at least four implementations of the forms interface: a lightweight interface most suited for user-interface elements such as dialogs, and

more capable implementation suitable for Access/Fox including some dynamic layout, Word documents, and Excel spreadsheets. It is presently unclear how the forms work going on in Cairo and Visual Basic relate to our overall goals. How are they different? Which can be used with C/C++? Isn't the Cairo/Chicago shell based on forms technology and a superset of Navigator, WREN and our mail front ends? Mail, including its shell extension piece, needs to handle anything that conforms to the forms interface. Our language developers still maintain a special interface between the language and the forms editor, however we need to abstract this interface. There are a lot of architectural issues relating to forms: multiple views of the forms-a form editor versus a package that just wants the data from the controls, hidden fields, field layout properties, namespace of fields, forms versus generic objects with properties, forms packages bundled with Windows.

The cost of not having a common forms solution has lead to overlapping development, user confusion and substantially higher localization costs.

process.

Owner: Roger Heinen and Jim Allchin are in charge of driving this forward.

Data Storage

Current storage systems include: MAPI 0, MAPI 1 local and SFS, DocFiles (compound document files in OLE 2.0), Fox, Jet w/Red, Blue, SQL Server, Cairo OFS, EMS directory, EMS message store, NT ISAM for configuration, Chicago ISAM, WinPad, Works database and a number of file systems (FAT, HPFS, NTFS). For every storage system users need and we develop tools to administer, diagnose, secure, browse, query, and program. We are starting to add *trigger* capabilities to a number of these storage systems. We have several different replication strategies and implementations being implemented today. We are starting to define hierarchical view interfaces with rich cursors, and we need to do this one time. There has not yet been adequate communication between all of the groups working on data storage issues.

For example, our mail application is essentially a simple browsing tool for the Mail storage format. At present, no other query tool can examine this format and programming to this format requires understanding MAPI and using a MAPI driver. The mail front end is special purpose with its own unique user interface characteristics. We need to unify all of the "object" focused storage handlers and APIs around Cairo standards (Mail, EMS, WinPad). We need to unify all of the ISAM-level handlers and relational table handlers around a standard that inherits from Jet and SQL Server and includes local and server implementations. This unification includes a common API. We need to determine the relationship of the "object" oriented stores to the more traditional record based storage systems so that things like triggers, data dictionary, security, links, programming interfaces, are shared where they can be.

Our complexity here has cost us dearly in competing with Lotus Notes since we keep hesitating about which store will have the front end tools, replication, flexibility, and programmability to allow us to do good workgroup applications. The cost of this delay is very high. DDT is trying to unify its various storage engines. Cairo is defining its storage APIs. We need a complete picture including determining how to fit Ren and WinPad into the picture. This picture also includes the administration tools and programming issues described above.

Owner: Roger Heinen and Jim Allchin need to determine how we move this forward.

Text Handling

Today, Microsoft has text handling development going on in Word, Help, Viewer (Advanced Technology), Publisher, Ink, PowerPoint, InfoDoc (Cairo), Mail, Works (Mac and Win), C editor, VB editor, Write and perhaps many other places. At the lowest level the number of *text controls* implemented inside applications and externally is also very large. This leads to substantial costs in adaptation for double byte and bi-directional, as well as lots of incompatible user interfaces and overlapping development. Our wasted cost on localization alone exceeds $10M per year. We shouldn't have to change any code for bi-di or DBCS support. Although a single solution will not suffice we ought to be able pick a few packages and standardize on those. We certainly want to implement the WritePad *applet* using a standard, system-supplied text control in Chicago.

An architect is required to think through these issues, including: What do the language products need to use external technology? What are the differences in editing technology required for each of the client applications? Can we do a better job using the Ink code, for example, in Works? Utopia is the project at Microsoft which has done the most work on trying to wrap existing Microsoft code, including text handlers, as objects to make them part of Utopia. For Publisher, PowerPoint and the word processor part of Works there is a major question of whether they should be developed as special adaptations of Word rather than as separate products. The savings that this approach could generate would be very dramatic. We need to urgently investigate this. Groups that find themselves implementing text handling solutions should question their efforts immediately since there is a high likelihood they are duplicating effort and creating a problematic situation.

The solution to the forms problem will help a lot with low level text handlers since we can avoid recreating several text controls for each forms package. We need an ambitious architect to help us move forward on this. Staying with our current approach will force us to adapt every one of the existing packages for pen, voice, new user interface, 32-bit, RISC, text indexing, on-line viewing, programmability, UNICODE, device independent layout, advanced linguistic utilities and new locales. Several people have suggested that some layout logic should be included in the operating system itself, as

Apple has done. This would allow a great deal of flexibility in allowing outsiders to write text controls without duplicating layout logic in an incompatible fashion.

Owner: Chris Peters will work with me to determine a plan.

Programmability

In the object oriented world where all of our applications are OLE 2.0 enabled we can think of an application as simply a large run-time where the user interface has been defined using our graphical form tools and a rich set of controls. We need the programming interfaces of the applications to be rich enough and fast enough that a high percentage of new features can be written in VBA and shipped as interpretive code. This will allow our applications groups to organize more efficiently since it enforces an architectural boundary. It should make user interface changes and localization fully leveraged off of the advances in our programming tools. It insures that the programming interfaces connect into all of the key events and include the ability to add new properties to persistent objects. Excel has taken the first step by including VBA inside. We need to evolve this implementation as we add VBA to Word so that it merges the user-interface models and insure that it has adequate performance.

Our applications should share a consistent architecture for building such add-ins. As an example, for me this means if I write an add-in that formats a table, I should be able to re-use that add-in in both Word and Excel. This requires an add-in architecture. We need to understand how we can expose the internals of our application data types in order to enable a much richer level of customization and reuse. We need to provide a way for users, and ourselves, to replace portions of an application. This will require our applications to expose and support some event model so add-ins can be notified of changes in state before, during, and after a command is executed. End-user customization needs to mean more than custom toolbars and menu arrangements.

We want our interpretive language to be pervasive in many ways; it is an enormous asset. We want it to be available as part of query processors, triggers, editable files, and display files (print or on-line). This means that not only field calculations but also layout decisions in Word documents or other forms packages can be expressed in code. A subset of the VBA interpreter will have to be included in the operating system. We want to be able to download program fragments in a compressed form dynamically as part of our on-line protocol. We want to run on any processor type. We want developers to use VBA for all sorts of applications, including multimedia titles. Some elements of the language, including how it binds to our objects and the protocols for sending it, need to be patented. Our language needs to support visual programming front ends that combine the best features of high level recording, "visual programming" a la Nextstep, and Metaphor capsules.

We need to unify the various formula languages we have into a coherent family. Specifically field formulas in applications, spreadsheet formulas and VBA formulas need to move towards a common set of functions and conventions.

We need to compare our languages with efforts like Script-X, Tele-Script, LotusScript, Borland Script, Dylan, Applescript, and Smalltalk to make sure we have the best interpretive language. Extensions like garbage collection for rich data structures or a built concept of time may be important. Our interpretive language should be looked upon as a great technical achievement.

Owner: Roger Heinen has the responsibility for making sure VBA is a great language. Pete Higgins should insure that our applications share a common programmability and add-in model.

On-line viewing

Reading of documents will increasingly be done on the screen instead of on a printed page. Our efforts with InfoDoc, Help, Viewer, Word, MSDN, PowerPoint, cue cards, multimedia publications tools (several) and CBT are all attempts to deal with this problem. The most general thinking about this problem is being done in Advanced Technology but it needs to be brought together with work in Word and built so that it extends VB. It is a crime that Word is not a good tool for authoring help and that help has to be slowly batch compiled to be tested. Some of the difficult issues include dynamic layout, runtime size, and user model and interface. We may need a special runtime viewer for special operations.

Owner: I am expecting Advanced Technology and Word to help us define this problem including all on-line authoring for our internal efforts.

Drawing / Painting

Drawing is a key technology. Whether it is done by outside acquisition or internal work we need to have world class drawing capability for our applications, including the ability to do visualization by connecting the drawing up to constraints. In a compound document model the layered/structured drawing engine would allow for much greater simplicity in the design and implementation. Making this rich enough to handle all of today's graphs, future graphs, process depiction, maps, seating charts, etc. and still allowing direct manipulation user-interface is a considerable technical challenge but one we need to tackle. This *constraint based* drawing is an exciting feature for lots of visualization tasks.

Owners: The Excel/Charting group and the PowerPoint group should work with Pete Higgins to move this forward.

CD-ROM

During 1995 the majority of all PCs will be shipped with a CD-ROM drive installed. This will change the packaging and delivery of software as well as dramatically reducing the space constraints we now operate under. We need a strategy for how we lead in this world; what does it mean for our current software categories in terms or value added capabilities?

We will create a value added version of Windows that I believe most PC makers will want to license. An element of the license might require them to pass through one or two CDs to the user without change and without requiring a royalty back to Microsoft.

CD distribution is good for Microsoft in the sense of allowing us to get distribution for a broader number of our titles, including small things like fonts. It is also good in terms of simplifying the ongoing annuity relationship with existing customers since installing new versions will be easier and having new versions have breadth of value to justify the yearly fee will be easier. On the other hand, CD distribution will greatly simplify software distribution for competitors who don't have large sales forces and brand recognition, somewhat neutralizing the position we have built up.

Owners: Paul Maritz for Systems and Pete Higgins for Applications need to drive this.

Pen

We believe the pen will become important in the next several years. For example when writing on a white board the board should be able to create an electronic document with certain portions stored as Ink and other portions recognized. Although we have relegated the extensions of Excel for Pen to slate for the time being this will become mainstream technology by 1995 and will need to be addressed across the board by internal development. Our Mobile Windows group is continuing to invest in advancing the recognition technology and will be the focal point for defining user-interface ideas and evangelizing this opportunity.

Owners: Mobile Windows will drive this issue, with the product groups moving more aggressively to include this technology in our applications.

User Interface

We have too many interface elements today: menus (with and without hierarchy), dialogs, toolbars, popup menus, short-cut keys, status bar, direct manipulation, various unique controls on dialogs, etc. There is no clear guidance about where any of these should be used. Often it appears as though new interface elements are added solely for a coolness factor. Interestingly, each additional command

execution technique makes it more difficult to provide an insulated environment for solution providers, which is a key to the success of the Office *runtime*.

A move towards modeless elements that can be shrunk/expanded and can respond to the current selection may hold promise. I was intrigued by the toolbar/dialog control utilized in Improv.

We need to redefine the relationship between the shell and the applications. We need to synchronize our efforts to do a new shell user-interface with our applications' interface.

Further in the future we need to eliminate the difference between a document being edited and a saved document. This is a confusing distinction although today programs use it to have a compact format versus a format for quick operations with additional state for both speed and user convenience. Users are used to the idea of having a saved version where save is like a commit. If you are just opening and closing views on data you don't want to have to consider starting and stopping the application.

Ideas about changing/improving user-interface do not have a focal point where everyone looks for leadership. We have continued to tweak the user-interface without improving the underlying user-model. In fact, the proliferation of user-interface elements has degraded the Windows model over time. Our competitors have expanded the desktop metaphor to the workplace, office, etc. A unified shell team working closely with the Office group has an opportunity to become the focal point.

We also need to identify some high reach goals for user modeling that can be pursued in Advanced Technology or Research. The idea of keeping track of everything a user does and recognizing patterns and using that for suggestions and automatic customization is a deep one that probably involves using the Bayesian logic engine work in Research.

Owners: Paul Maritz will determine the next step. This should be done in coordination with the Interoperability group and the unified shell team.

"12-24" (Ship at Same time

In order to gain revenue from the installed based we want to be able to ship products that are compelling on a predictable schedule-every 12 months with every other release being major. Users have to feel the new release will not confuse them or overwhelm their system resources. Revenue from our installed base will become more important than revenue from new users as we saturate potential users. A company that determines how to deliver on a predictable schedule will have a major advantage in selling subscriptions and having users expect the new release.

The sooner we are able to move to a model where we use our unique interpretive language, VBA, to implement features and add-ins for our applications the more predictable these releases will be.

The systems groups have been able to work on multiple versions of our key system software. Our applications groups are just starting to work out the issues associated with this. I realize there is a significant challenge. Critical to the success of this effort, will be high bandwidth communication among the product teams as they specify version *n* and *n+1* in roughly the same time frame.

Sharing concepts

There are a number of other application features that are, at the highest level, functionally the same across the entire Office. For example, if the sophisticated page layout available in Word was somehow available in Excel far more complex spreadsheets could be designed. Thus from a customer perspective leveraging features and usability models across applications is beneficial. Pagination, print preview, named areas, styles, outlines, formulas, and wizards are among the things that exist in our applications in different forms that might benefit from unification. A more extensive list of these overlaps needs to be developed.

Owner: The Interoperability group needs to be the focal point for identifying these areas of leverage.

Futures

In addition to the above technologies that I see being architected and developed over the next six to 18 months, there are a number of areas that both our Applications and Research groups are investigating. We need to exploit these technologies in an efficient and timely manner. The recent Word retreat highlighted a process where the product groups will allocate specific resources to work with the research group once a technology has reached the point where it needs a product engineering focus. The Word group is doing this with natural language processing today, and I feel very good about this.

A number of technologies deserve a closer look. I will assume that product groups will identify ownership and coordinate their efforts with research as needed.

- Blackboarding so that probabilistic input systems like handwriting and speech can work together with the application to determine the correct input.
- Recognition and natural language processing.
- Personality user-interfaces as being investigated by both research and Utopia.
- History tracking gives us the ability to provide a runtime customizable user-interface.

- Applied intelligence in a more architected manner will allow us to consistently across applications provide a *do-what-I-mean* interface. Today's *ad hoc* approaches work very well, but have limited extensibility and synergy across applications.
- Structured documents will be one tool for implementing better intelligence in our applications. The use of structure came up recently at both the Word and Excel retreats and there seemed to be no shortage of ideas on how to use structure. A number of challenges were identified in terms of elucidating structure from the user's input, thus eliminating the need for the user to understand structured input.

MS5048489.11
CONFIDENTIAL