29

Some Ideas on Exploiting Product Synergy in Word

Mike Mathieu, Word Program Management, 7/1/93

Introduction

During the past few years, we have been able to steal significant market share away from WordPerfect. An important lesson to learn here is how relatively easy it was for us to do this. It should serve as a warning for Word. In today's market, word processing might be called mission critical, but no specific word processor can be called mission critical. As long as there are low transition barriers, it will be all too easy for a small, new, low-priced competitor to come along, errode prices, and steal business from us.

An important goal in the 3-5 year timeframe is to increase the "mission-criticalness" of Word--at least to the same level that an app like Excel is considered mission critical. In the past, file formats, macros, unique features, and app-specific user interface conventions were all barriers to transition. In today's competitive world we must go far beyond these former barriers and exploit synergy with business system components that will help entrench Word. This document serves as a starting point for the general discussion of how to better entrench Word, and offers some ideas on the potential areas for developing synergy with other products.

Email Integration

Besides a true word processor, the next most common editor used by the average person is their email editor. If we can make Word so that it can take the place of the email editor, then users will be more satisfied--they get a more feature rich mail editor and it's totally consistent with their word processor--and we will be able to leverage off of the company's choice of email system (a mission critical system) and link it to purchases of Word. Today we don't really integrate any better with MS Mail than somebody like WordPerfect does.

The question then becomes whether we exploit private interfaces in Mail to make this happen, or whether we rely on public interfaces--which our competitors can use for the same purpose. When the email system comes with the OS, this becomes a bigger problem. One possible idea is that this extra level of integration only comes when using the distinct Microsoft Mail product, as opposed to the mail client built into Windows. (Is the current plan still to ship two separate mail clients?)

Some specifics to consider are:

- Using Capone's custom forms feature to register certain Word templates as a form. (Who handles textizing?)
- Use Capone's (more robust?) routing features, rather than DAD app-specific routing.
- Using Word's File Open dialog to go into the LMS file and open any Word documents, or even preview file attachments in other previewable formats. In general, be able to use the LMS as a regular extension of the file system. e.g. Find File, links, Save As, Move, Create Directory, etc.

Forms Integration

There are several plans circulating around now about ways to take advantage of Word's forms and layout capabilities as a component of a more generalized enterprise forms and document database system. The idea is to provide some unified front end browsing tool that lets you get into a database, and then use a variety of forms tools like Access, VB, or Word, to display the data in the most appropriate manner.

It is interesting to note that Microsoft Eforms really has no special level of integration with Word or other desktop apps. We'd have to be able to create a new positioning, perhaps based upon the notion of a document database, where Word could serve as one of the value-added forms components for this mission critical document sharing system. (In this sense, Word would be a superior competitor to the front end document forms client that Notes provides. It would be up to another group to come up with the approriate backend system to compete.) We'd want Word to be able to tie document fields into backend databases, provide features like lookups, and some sort of security or data protection scheme.

Shell Integration

The idea here is that once a user has bought into Windows, aside from the ability to run other Windows app, the shell itself is probably the most important component of their user experience. (This in itself should serve as a warning to the Chicago team. Too big of a change would lower the barriers for competing shells.) Having Word integrate perfectly with the shell would give us an advantage over those word processors who didn't integrate as well. As in the email case, this area of integration is not able to provide our apps with a unique advantage that we could sustain. As soon as we integrated nicely with the shell, another app could some along and do the same thing. This would just extend the feature wars into integration feature wars, rather than making Word more mission critical. (Of course, if we're in the lead in the integration feature wars, this is a way that we could stay ahead of the competition without having so much obvious "feature bloat" or reduced ease of learning.)

Some obvious targets for shell integration includes making MS Write's file format identical to Word's file format; providing document previewing filters; OLE drag and drop scrap support; OLE Automation support; an Explorer handler dll to provide enhanced browsing into components of a Word document (e.g. master document pieces.)

Word as an Online Document Viewer

Particularly with the growth of email, the use of rich online documents is likely to grow significantly in the next five years. This area talks about a set of related ideas like making Word into a replacement for the Multimedia Viewer, or making Word into the authoring system for Help files, and perhaps even a display engine for help files.

Help Files

Today there is a whole cottage industry of products either built around Word or around other word processors to allow someone to easily create help files. The manual process is tedious, and so software developers look to these tools to ease the process of creating the help files. Creating a help file today requires a RTF file with special tags to indicate specific objects. This file is then put through a help compiler, finally producing a .hlp file that the help engine can use. Since help files are mission critical to software products, we could make Word a perfect help content editor, as well as a great viewer. Hlp format could be a standard output format by combining the processing of the help compiler with the interfaces of Word's export converters. Having Word read hlp files would help solidify the hlp format as the standard format for online documents (do we want that, or do we want .doc files to be this standard?) Some additional issues to consider are:

- Would we stop improving or selling the Help Compiler?
- Would ISVs balk at the idea of having to buy Word to make help files?
- Would customers want Word to support hlp files, or just want Word to be able to easily create documents with easy popups, secondary windows, hypertext jumps, and per-page layouts?

Multimedia Viewer

The notion of using Word as a replacement for the Multimedia Viewer implies some pretty big changes in the way we sell and distribute Word. For instance, today we give away MM Viewer in order to sell the content. We've never done anything like that with Word.

If we could come up with a business model that allowed us to give away some viewer-like version of Word, that could help make Word's document format the de facto standard for online information. (There are tons of CD's and online services around today, and they all use different formats for the data. We need to think about potential advantages to owning the standard format for online documents--besides the fact that Word itself would be the best content editor, and that Word would have the highest fidelity display of that content.)

We could perhaps invest in a model similar to that of Folio Views, where there is a full version of Views used to create the content. Then, individual users can use Views Lite to view the content and make edits to their own private "shadow" file (this is essentially the same thing as a "changes" file that I've described before in relation to replication.) Perhaps this is a separate product that just happens to be very similar to Word proper?

Standard Format for Information Distribution

This is an overriding theme in several of the sections discussed in this document. By providing a standard format, Word will also become the standard authoring tool, as well as the standard browser and content delivery platform. Word, by typing to workflow, and document production systems, will become a standard tool for the distribution of information--in effect, becoming a modern day, GUI replacement for the old standard ASCII file. This would include higher end, online information sources like e-books and e-libraries.

We should also consider how Word might fit into our plans for an online information service. Could Word serve as the standard information browsing front end, and allow for a textizing engine that provides for users without Word?

Tight Integration with a Search Engine

The search engine in Cairo is one of the focal points of the product as a step towards information at your fingertips. We need to think about ways that we can let Word integrate better with either the Cairo search engine, or third party engines, to further support Word's position as the leading front end for browsing online information.

This should include a standard UI (same as Cairo, or more value added? How do we differentiate Word?) for searching and navigating the information hierarchy, and a smoothly integrated interface for viewing the information.

Hyper-Integration with Excel and Office

It's not clear what this means in terms of features, but the idea is to use (private?) interfaces between Word and Excel (and the rest of the Office?) to provide a tremendous level of integration. This might be describing the Integrated Office, or it might just be extensions to the apps in the current Office. Some things to think about are:

- Use Excel engine to provide a great Table object in Word.
- Take more advantage of system support for messaging. (Distributed processing? App-to-app communication?)
- Implement commands which take advantage of features in other applications.
- Implement features as objects or shared components so other apps can use them.
- Complete support for Object Basic
- OLE 3.0 (whatever we need it to be)
- Platform (or Microsoft) standard tabular and textual interchange formats
- Component architecture taken to it's logical extreme (explosion in variety of shared components.)
- Consistency with object oriented shell and support for OFS. (Cairo)
- Native drawing tools shared

FL AG 0021495
CONFIDENTIAL

HyperIntegration and an extreme component architecture could have a significant impact on how we market our software as a company. This will need a lot more thought as the time approaches.

MS 0127355
CONFIDENTIAL