PLAINTIFF'S EXHIBIT 3414
Comes v. Microsoft

Opus Development Postmortem

Peter Jackson
December 15, 1989

Table of Contents

I. Introduction and Summary

The Opus Development Postmortem was held Tuesday 12 December at the Bellevue Red Lion Inn. In attendance were:

David Bourne	Doug Klunder	David McKinnis	Mark Seaman
Chi-Chuen Chan	Tony Krueger	Krishna Mukherjee	Brandy Thorp
Brad Christian	Jurgen Leschner	Rosie Perera	Doug Timpe
Sylvia Hayashi	David Luebbert	Tom Saxton	Brad Verheiden
Peter Jackson	Chris Mason	Doug Scott	Bob Zawalich

MS-PCA 1295250
CONFIDENTIAL

After Word for Windows was released to manufacturing a questionnaire was distributed to all developers and some other parties to collect opinions and issues for discussion at the postmortem. The responses to these questions

were distributed to all of the attendees in advance and served as the agenda for the meeting[1]. This document is primarily based on those responses and on the discussions during the meeting.

The Opus project has been a long one. Many things went wrong and many things went right, but the final product that was produced is one that we can all be proud of. This document focuses on the things that went wrong—hopefully we can learn from these.

II. Project and Schedule History

The Opus project started in August 1984 as the Cashmere project. The project started out being the end-all Windows office, but eventually became a Windows word processor based on Mac Word. Prototyping work started in August 1985 and real development in November 1985 with seven developers. Code complete was declared in October 1988. The product went to manufacturing on November 30, 1989. A more complete outline of the project's history is included as Appendix I.

The Development Team

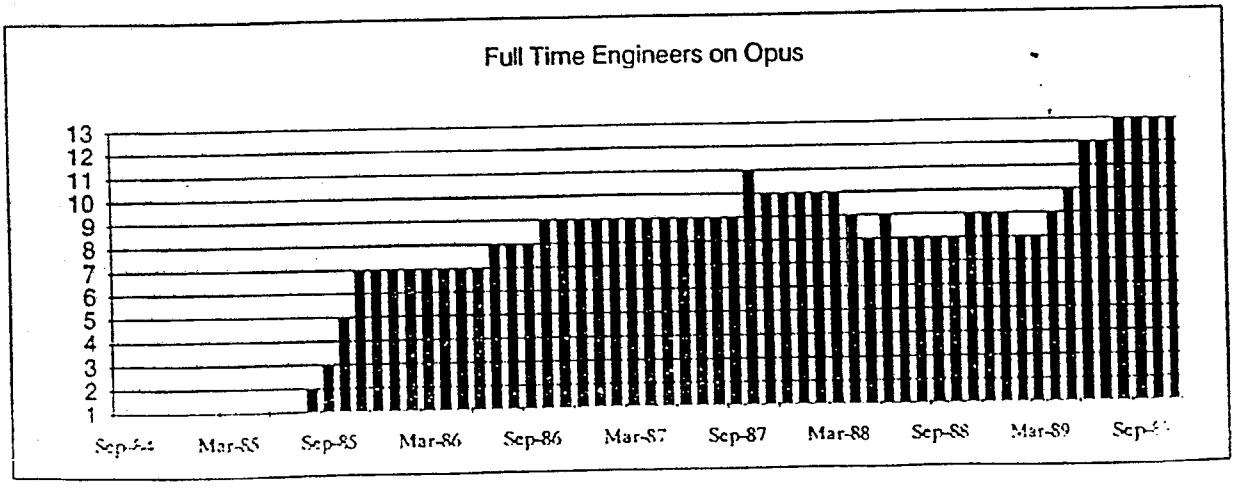
The following table lists the people assigned to the Opus project, the times they were assigned and the major areas they worked on. I have included here only people working on the Opus (Cashmere) core, I have not included people or periods spent working on Write or the Cashmere/email project (which was scrapped in 1986).

Name	Period Position	Areas
Bourne, David	09/85-11/89 SDE	printing, layout, styles, renumbering, print-preview, insert file, postscript support, compare versions, revision marking, importing spreadsheet formats, run clipboard/control panel, undo, document types
Brodie, Richard	08/84-07/86 Project Manager	initial research and design concepts, RTF, and much more
Chan, Chi-Chuen	08/86-11/89 SDE	multi window support, style name area, selection/cursor movement, footnotes and annotations, headers and footers, file find (document mgmt), pageview, PA coordinator
Christian, Brad	09/85-11/89 SDE	macros, key maps, menus, formulas, dialogs
Cockburn, Anthony	05/88-09/88 SDE	performance
Cox, Greg	09/86-05/88 SDE	native coding, speller, file conversions, dnatfile
Ezekiel, Alan	06/88-09/88 Intern	help & CBT hooks, error reporting
Geyser, Eric	07/87-08/87 SDE	show all
Hayashi, Sylvia	05/89-11/89 PA	testing, debugging aids, code searches, demo version
Hopstad, Mike	01/88-07/89 PA	printer bugs, testing, RAID administration
Jackson, Peter	06/86-04/88 SDE 04/88-02/89 Technical Lead 02/89-11/89 Project Ld./Tech. Ld.	bookmarks, glossaries, fields, save, open, file primitives, coversions, graphic filters, ruler, icon bars, auto numbering, print merge, goto, performance, DDE, memory management, PA coordinator
Klopfenstein, Herb	08/87-01/88 PA	testing, RAID administration, net administration
Krueger, Tony	06/88-09/88 Intern 07/89-11/89 SDE	macros, macro record
Lammers, Laurel	09/88-11/89 PA	sampler, macros, benchmarks, testing, techref reviews

[1] The original responses to this questioneer as well as the document distributed and some additional email discussions about the history of the project are on file for anyone who wishes to review them.

Loofbourrow, Bryan	11/85-07/86 SDE 07/86-04/88 Technical Lead 07/88-02/89 Project Lead 02/89-11/89 SDE	display & scrolling, speller, thesaurus, hyphenation, expressions, file system, "word technology", out-of-memory handling, fonts, draft view, clipboard display, performance
Luebbert, David	04/89-11/89 SDE	outlining, styles, scrolling, cursor movement, save
Martin, Ford	07/86-06/88 Project Lead	thesaurus, scheduling, specification
Mason, Chris	05/89-11/89 SDE/Dev. Mgr.	layout, postscript support
Matthews, Bob	01/85-11/85 Project Lead	
McKinnis, David	04/89-11/89 SDE	printing, printer drivers, PRDDRV, tables, revision marking, renumber
Mukherjee, Krishna	10/88-11/89 SDE	macros, glossaries, menu customization, formulas, field translation
Perera, Rosie	08/85-11/89 SDE	status line, search, replace, CBT, help, tables, window splits, view preferences, word deletion, index & TOC generation, cursor movement, debug menu, go back, PA coordinator
Porter, Dan	11/87-03/89 PA	network administration, testing, benchmarks
Rutenbeck, Jeff	10/87-05/89 PA (part time)	testing
Saxton, Tom	04/89-11/89 SDE	table display & caching, table native code, display
Scott, Doug	05/89-11/89 PA	macro tests, SDM verification, testing
Singer, Marc	05/85-12/85 Intern	dialogs and product specific controls
Thorp, Brandy	05/89-11/89 PA	librarian, disk images, macro tests
Timpe, Doug	01/88-11/89 PA	network administrator, macro tests
Verheiden, Brad	10/87-11/89 SDE	native code, performance, search, file preload, large table support
Wine, Bruce	06/87-08/87 Intern	PRDDRV
Yamane, Yoshito	07/85-07/88 SDE	dialog controls, hyphenation, expressions, outlining, insert field, summary info, customize, file dialogs, sort, date, time & number formats, autosave, kanji Write
Zawalich, Bob	08/85-11/89 SDE	dialogs, RTF, pictures, TIFF, dyadic opts, macro specification, ribbon, ruler, clipboard opts, sort, foreign file conversions, formatting commands

The following shows the allocation of full-time equivalent SDEs to the Opus project through time. An FTE is an attempt to measure the real strength of the team by counting SDEs who are working full time on development. Interns are not counted. New hires do not count during their first month. The number may also be adjusted according to the actual amount of development someone is doing.



Management of the development team, from within and from above has been one of the major failures of this project.

Bob Matthews, the original Project Lead and one of the only experienced developers on the project, was siphoned off very early to work on Windows. That left the development team working directly for Richard Brodie who was then the Word Processing Manager (also responsible for PC Word). While Richard did much research and contributed to the design of the product, he did not manage the team well and he never created a specification. When he resigned in July 1986, most of the work he had done during the preceding two years was lost. Richard's mismanagement set the tone for the next three years.

After Richard left the responsibilities were split between Ford Martin as Project Lead and Bryan Loofbourrow as Technical Lead. This would have worked out quite well had it not been for the excessive demands of the then Director of Applications Development, Jeff Harbers. Jeff continually hounded Ford for better schedules and more results. He treated the development schedule as a contract between the development team and himself and he really let us know when we did not live up to our end of the bargin. To make the situation worse, he questioned every estimate made on the schedule, resulting in a tighter schedule that could not be meet. (This is discussed further in the section on schedule analysis on page 9).

In early 1988 things were looking bad. Development was way behind schedule and Bryan was getting sick. During a development team meeting in early March Jeff made perhaps his biggest mistake when he got up and told us that the Opus team was the worst team in Applications Development. This, combined with the long project duration, the continual pressure of being behind schedule, and the upsets in leadership, contributed to destroy the team's spirit. This lack of spirit or team synergy is evident right up through the time when we finally did ship. The team became apathetic and burnt out.

In April 1988 Bryan went on a medical leave of absence and, feeling that he had no other recourse, Jeff made Peter Jackson, a junior member on the team, the Technical Lead. About this time Jeff considered disbanding the Opus team completely and starting over; in hindsight that might not have been a bad idea, though it probably should have been done when Richard resigned.

In June, possibly because he could no longer stand the continual pressure from his boss, Ford chose to take a leave of absence. Bryan, who was better but not well, came back and took over for Ford as the Project Lead. The next few months things were looking better and better. Opus made feature complete then code complete and it was looking like we were getting the bug list under control. Unfortunately this did not last. Within a couple of months it become obvious that Bryan's condition made it impossible for him to lead the team, he was not here nearly full time and he was not at his best when he was here.

Finally, in February 1989, Chris acted by making Peter both Technical Lead and acting Project Lead. Not only did this again disrupt the team, but it also gave one inexperienced person the responsibilities that should have been shared by at least two experienced ones. During the next ten months, Peter continually experimented with strategies to improve morale and to get the product done. Some of these worked well, others did not, but the amount of administrative overhead generated by these strategies and at the disruptions of going from one to the next probably did more damage than good.

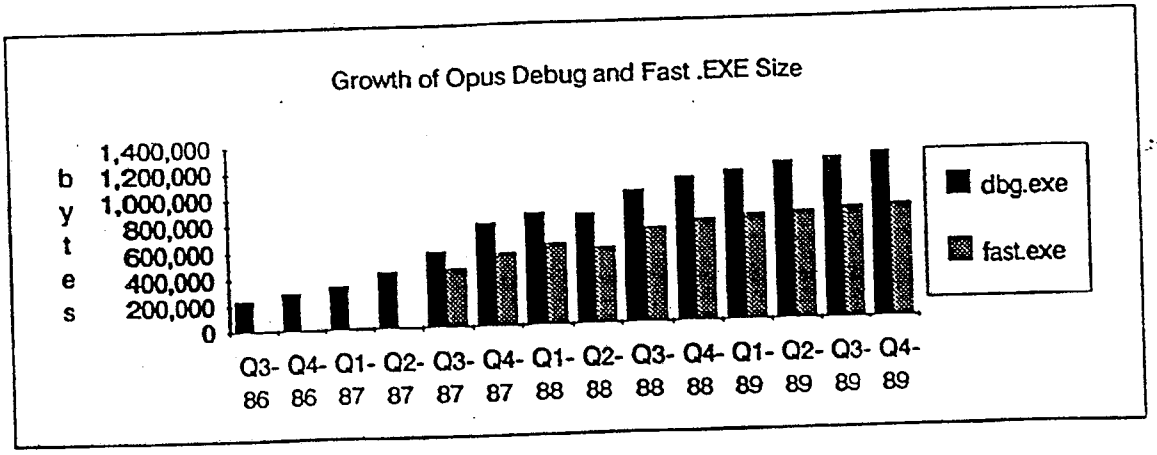
Not having a Technical Lead who could concentrate on technical issues definitely cost us. The size, speed and memory usage of Opus could have been made better than it was if the Technical Lead had not spent the last 18 months as Project Lead or covering for sick and burned out Project Leads.

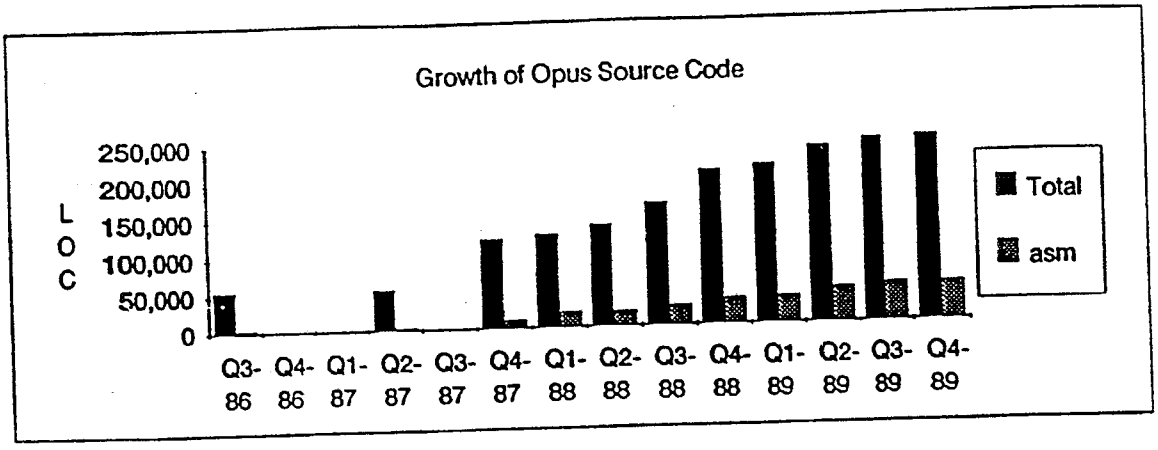
At its height, during the summer of 1989, the Opus team had fifteen developers, six programmer assistants and seven interns. Twenty eight people on a single team is *way* too many both because no one lead could possibly keep tabs on what each person was working on, but also because no team member had any idea what any of the other team members were doing. In the future teams should not be allowed to get this big. Smaller groups, possibly sub groups with leads and well defined tasks, would function much more efficiently.

The management of Programmer Assistants on this project has not been very good. PAs have not been given any consideration for career development and suffer from having too many people telling them what to do. PAs on the Opus team were organized under a PA Coordinator (first Peter, then Rosie and finally Chi-Chuen). The PA Coordinator was supposed to be responsible for assuring that tasks were well distributed among the PAs. This arrangement was subverted because the PAs would continually be getting requests from other people, including people in other groups.

Project Statistics

The following charts show the growth of the debug and non-debug Opus executables and of the number of lines of code in Opus[2].





Code complete was declared in October 1988. Subsequent to that more than twelve months of bug fixing and additional development work were done. The following table shows how much of the final executable was present at code complete.

Milestone	KLOCS	DBG EXE Size	FAST EXE Size
Code Complete (10/88)	208	1097	775
Shipped (11/89)	249	1260	853
% at Code Complete	84%	87%	91%

In comparison, Mac Word 4.0's Code Complete executable was 78% of the size of the shipping executable. PC Word 5.0 has been calculated to have 76% final code at Code Complete.

Of course, this does not measure how much of that 91% was rewritten. We are working on a method of using the SLM DIFF files to get a measure of how much really changed after code complete, but we do not have that information available yet.

[2] As measured by the utility CLOCS.EXE. This utility tends to count 50-60% of the lines in any given C source file. Thus a 1000 line file (as shown by WC.EXE or your favorite editor) would probably be counted as having 500-600 lines, depending on the density of the code.

In the summer of 1989, at a point where it seemed we might never converge, a program emphasizing quality of changes instead of quantity of changes was instituted. This program included code reviews and code ownership as well as a series of reminders and discussions to encourage people to think about and to be careful with the changes they made. This program was an attempt to instill some of the methods of zero-defects into a project that had gone a long time using an infinite-defects methodology and was too far in its development to consider starting from scratch.

It is really hard to draw any conclusions about the effectiveness of this program. It is true that the bug count dropped dramatically, regression rate decreased and we did finally ship four months later (we did not remeasure the injection rate to determine if it changed), but that probably would have happened anyway, if you believe that the product was ready to converge. The only real metrics we have are from the statistics gathered during the code reviews. The following table shows the areas reviewed[3], the time spent on the review (includes preparation, review and documentation), the amount of time making corrections and the number of "items" the review found (bugs and other items).

Reviewed Topic	Hours Review	Hours Correction	Bugs	Other
Tables: Formatting	67.3	26	14	55
Macro Execution	62.5	3	3	37
Print Preview (w/o Layout)	120.5	40	36	146
Table Primitives	111.25	30	26	203
Core Edit Routines	88	8	11	39
Table Display	87.5	18	10	85
Macro Tokenization/Detok.	68.5	25	12	61
Outlining	55	12	19	73
Total	660.55	162	131	699
Average	83	20	16	87

In addition to the 131 bugs and their solutions (or at least their causes) that these code reviews found, they provided a great educational benefit. The developers who participated in the reviews learned more about how Opus works and about better coding practices. The testers who attended gained a better appreciation for the complexities of the product and ideas on the kinds of things that can go wrong.

Most of the non-bug findings were performance related. If code reviews had been used all along on the Opus project, it is entirely possible that our final product's speed could have been significantly better.

The following table lists some statistics about the Opus project for comparison with other projects.

Size of shipped executable	852,576 bytes
Hand-native code as percent of executable	7 %
Total development time spent (including interns and PAs)	55 man years
Full-Time-Equivalent SDE time spent	38 man years
KLOCS of code in debug version (using CLOCS.EXE)	249 KLOCS
Thousands of lines of code (excludes tools & SDM)	347 K lines
KLOCS per FTE-year	7 KLOCS/FTE-yr
Fast bytes per FTE year	22 Kb/FTE-yr
Number of bugs reported	12,511 bugs
Fixable bugs reported	9377 bugs
Percent of reported bugs that are fixable	75 %
Total bugs postponed	1197 bugs
Percent of fixable bugs postponed	13 %

[3] Five other code reviews were held or started, but no data is available for them.

MS-PCA 1295255

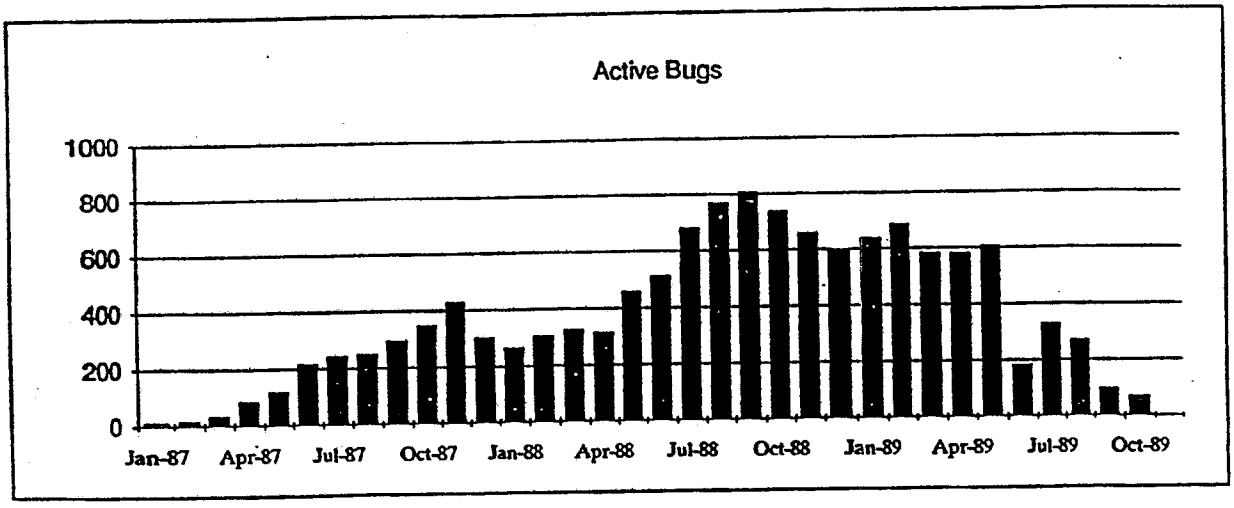
Fixable bugs per FTE-year	247 bugs/FTE-yr
Fixable bugs per KLOC	38 bugs/KLOC
Percent of FTE time expended after Code Complete	30 %
Percent of fixable bugs reported after Code Complete	62 %
Percent complete at Code Complete (by fast EXE size)	91 %
Percent complete at Code Complete (by KLOCS)	84 %
Percent of fixable bugs reported after ZBR	9 %
Total bugs fixed after ZBR	508 bugs

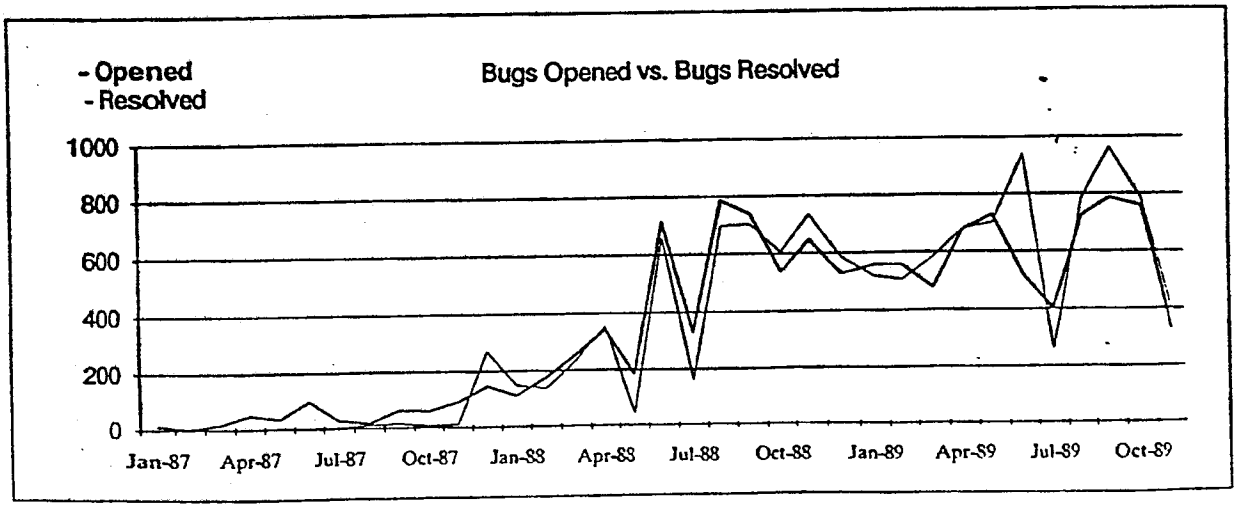
Raw data for many of these statistics can be found in Appendix II.

Bug Statistics

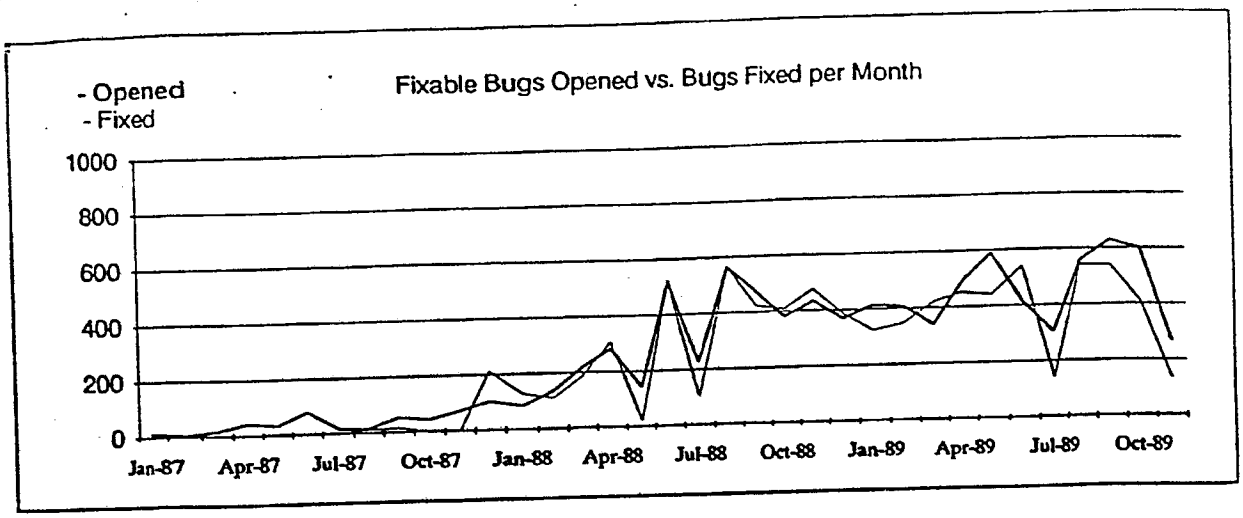
The Opus bug database was not started until January 1987, yet nearly thirteen thousand bugs were reported during the subsequent thirty-five months (about 370 bugs per month).

The following charts show the active bug list and the rate of bugs being reported and resolved (all bugs and fixable[4] bugs).



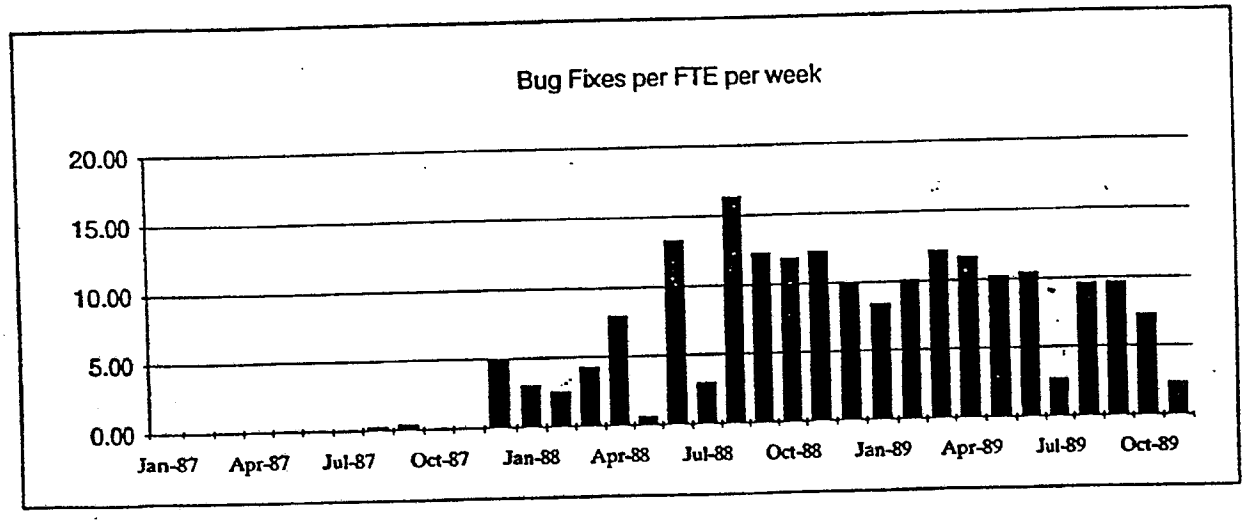


[4] A "fixable" bug is a bug which is eventually resolved FIXED or POSTPONED.



The interesting thing shown by these charts is that at no time did testing find significantly more bugs then development was fixing. Yet the small difference in the find rate and the fix rate caused the bug list to skyrocket, especially during the period from April 1988 to September 1988 (this was the period of the final code merge and completion of features). During the period before April 1988, relatively little was happening on the bug list, development was busy working on features (and introducing bugs) and testing had not yet geared up. During September 1988 a cross-over occurred and from there on, for the most part, development kept up with or exceeded the find rate. The turnover of bugs during this period was very high (about 600 bugs per month found and fixed). In June and July things tapered off as development turned it's attention to code reviews. Then in August, with the announcement of the Cancun incentive, things took off again. Finally both the find and the fix rates bottomed out in October and November as everyone decided it was time to ship.

The following graph shows the average number of bugs fixed[5] per FTE per week by month. These numbers are somewhat lower than the same figures for Mac Word 4.0, which ranged as high as 25 and seemed to average between ten and fifteen. I don't have any explanation for why Opus is so much lower.



The following table shows the distribution of bugs by area during the life of the project. This data cannot be considered too reliable because of the difficulty in assigning a bug to an area. This is difficult because the areas are assigned by the tester when the bug is opened and the area in which the bug seems to manifest itself may not be related to the underlying problem. Further more, the nature of areas makes it very difficult to pigeon hole bugs which are

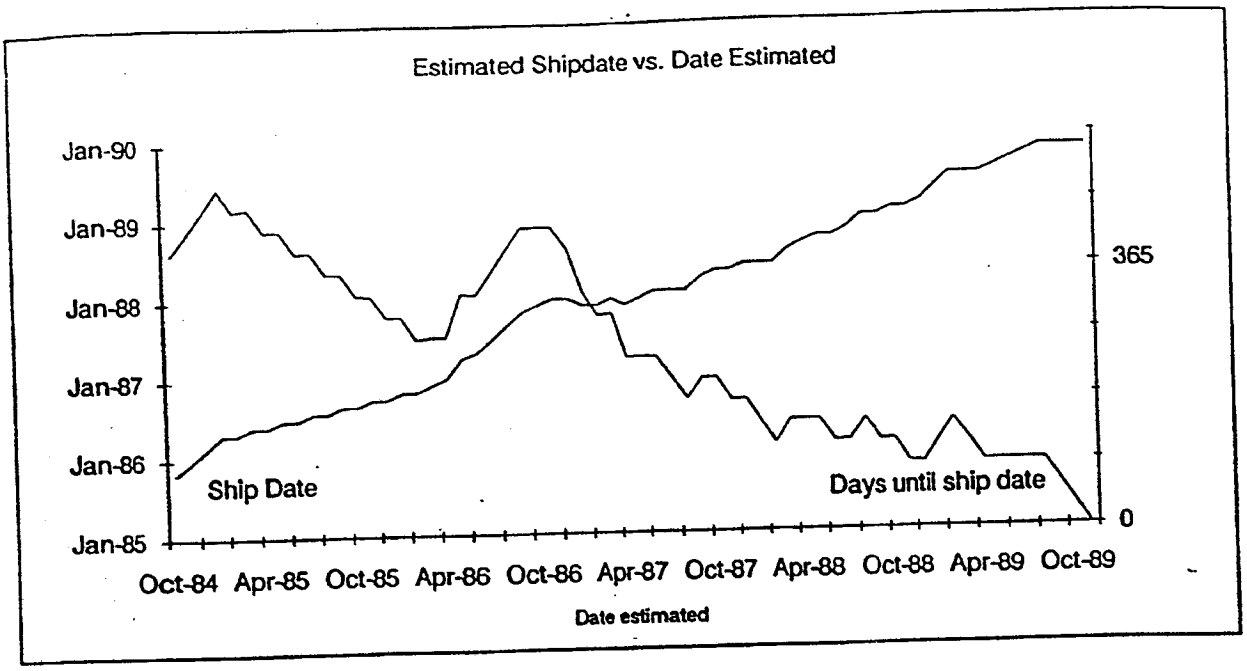
[5] Includes only bugs resolved FIXED.

caused by interactions between multiple features (such as tables and fields). A good percentage of our bugs, especially later in the project, are caused by such interactions.

Area	1	2	3	4	Total	% of Total
ANNOTATION	36	16	26	8	86	0.7
BOOKMARK	14	15	17	2	48	0.4
CONFIG	34	21	29	21	105	0.9
CONVERSION	145	589	174	22	930	7.5
DDE	53	25	27	7	112	0.9
DEBUG	27	9	5	5	46	0.4
EDITING	424	267	311	68	1070	8.7
FIELDS	99	119	123	37	378	3.1
FILE	374	226	208	35	843	6.8
FOOTNOTES	51	29	45	14	139	1.1
FORMATTING	150	270	197	39	656	5.3
GLOSSARIES	33	22	22	1	78	0.6
HEADER/FTR	52	48	33	7	140	1.1
HELP	87	89	63	20	259	2.1
INDEX/TOC	34	31	20	4	89	0.7
INIT-BOOT	51	19	24	6	100	0.8
INTERFACE	129	193	287	86	695	5.6
MACROS	584	778	467	117	1946	15.8
OUTLINING	15	13	34	14	76	0.6
PICTURES	36	56	48	13	153	1.2
PRINT	308	578	403	98	1387	11.3
SEARCH	88	77	62	11	238	1.9
SETUP	34	40	37	22	133	1.1
STYLES	79	108	82	12	281	2.3
TABLES	312	206	161	36	715	5.8
UTILITIES	212	274	245	37	768	6.2
VIEW	192	247	216	39	694	5.6
WINDOWING	43	54	51	14	162	1.3
TOTAL	3696	4419	3417	795	12327	

Scheduling Analysis

Opus was arguably one of the worst scheduled projects in Microsoft's history. The following chart shows the estimated ship date as a function of the date estimated. Notice that only during two periods out of the five years of the development cycle did we admit that we were more than a year from shipment. During the entire period from December 1987 until September 1989 we estimated that we were between three and six months of shipping.



During the early phase of the project (until mid 1986) there was no formal schedule, instead just a list of tasks that had to be completed[6]. The "Excel Model" scheduling was started about August of 1986. Many of the Opus status reports between mid 86 and early 88 are simply diffs of these schedules with justification for the changes. Several of these schedules are attached as Appendix III (8/25/86, 3/16/87 and 3/21/88). After April 1988, development switched to the "Block Model" schedule. One of these is also included in the appendix.

The methods of scheduling used were fatally flawed. A schedule should be considered a tool used to predict a ship date, it should not be considered a contract by development. Because there was so much pressure to meet the schedule, development got into a mode which Chris Mason refers to as "infinite defects." Developers get credit every time they can check a feature off, so they are more inclined to mark off their current feature and go on even tough it really is not done. There was a prevailing attitude of "the testers will find it" when thinking about potential bugs in code being developed. In many cases they did find it, and that is what caused our stabilization phase to grow from the expected three months (which is a pretty random number anyway) to thirteen months. Because every task was cut to the bare minimum, performance work that should have been done was neglected until the very end of the project, reducing what we could do in a reasonable amount of time.

The situation becomes worse since no one wants to see the schedule that would be accurate. Every estimate made by a developer was challenged, first by their manager then by his manager then by Program Management. This caused the initial estimates to always be far short of what would be realistic.

An interesting case in point was the great schedule review held by Jeff Harbers in August 1987. The purpose of the review was to generate a schedule which we could believe in. To do this we spent several weeks investigating all facets of the project looking for additional tasks that would have to be done. In the end we all sat down and explained (justified) our estimates to Jeff. The net effect was a negligible change in the schedule. The long term effect was a team that got burned out because they were busting their butts to meet a schedule that was too ambitious anyway and introducing a whole bunch of bugs in the process which made the stabilization phase longer which then caused everyone to be even more burned out!

The estimates used for our schedules were further compromised because each task was considered independently and consideration was not given for the ways in which features may interact. The most striking example of this is the interaction of Tables and Fields, discussed further in the section on Technical Issues (page 12). Tables were developed by the Mac Word team and were supposed to be nearly free for Opus. That proved to be completely wrong.

[6] The ship dates during this period are based on various sources such as notes from staff meetings, after late 1986 period they are based on ADL reports.

Our schedules also did not realistically allow for the inevitable changes in the specification, both small changes made as a feature is developed and major changes such as the addition of a new feature. These changes are inevitable and become all the more so as the project drags on.

Summary

The biggest question everyone will ask about Opus is why did it take so long? In the section above why we did so poorly at estimating how long development would take is discussed, but obviously this project took a lot longer than it should have. The following points summarize the main reasons as we see them.

Lack of an early, clear direction and specification. Early on the product was to be the end-all windows office. Later it was toned down to a word processor. Changes in direction like this caused us to waste a lot of effort in the wrong directions. The ideas for Cashmere that Richard had never got written down. It was not until mid 1987 that there was anything approaching a spec. Even then many areas of the product were wide open. The macro language was not well speced until mid 1988. I'm not suggesting that a project's spec be frozen early on, since that would severely limit our flexibility to respond to new information and market changes, but from the time development work begins the major features should be down.

Inexperienced team, lack of leadership. Very few members of the early development team had much experience. With a project of this magnitude and apparent importance, one would expect it to be better staffed. As discussed in the section on the development team above, the constant changes and other problems with the leadership on the Opus project cost us a lot in efficiency and morale.

Infinite defects. The principles of infinite defects were instilled in the project from the beginning. We started out by building a prototype and on top of that prototype we tried to build a product. This started us out on a very unstable foundation. If you are going to build a prototype (which is written quickly as throw away code) then throw it away before you start working on your real product. We then proceeded to add features as quickly as we could (because of our scheduling methods) which contributed more bugs than it did stability.

Redesigned, reimplemented, re-ported code, over and over. There are very few areas of Opus that were implemented only once. Almost all features were written then rewritten or re-ported from Mac Word. The continual Mac Word code merges (there were at least five full scale ones[7]) caused us more delays than any other factor. When the merges were complete and we were actually sharing code (through the WORDTECH SLM project) the pain was more spread out, but was still there. The Opus project would have been much better off if it had not done those merges (beyond the first one) and if it had never tried to share code. Another reason for the continual redesign work were the constantly changing platforms (especially Windows 1, 2 then 3). This is in part a result of our being so late but it did have it's own impact on our schedule. Having to rewrite features (such as the sort tables) to satisfy International also cost us—they should have been written right the first time.

SDM. The Standard Dialog Manager is another example of a reimplementation, but it was a problem even beyond that. We had a perfectly good dialog manager in Opus which, we believed, could have fulfilled all our needs. But in the summer of 1987 Jeff Harbers dictated that we take a new dialog manager to be provided by the tools group. The installation, bug fixing and performance work required on SDM was probably the second biggest cause of delays for this project. Even today we are not convinced that SDM meets our needs or that it is possible for an shared library on the scale of SDM to ever be successful.

Too much schedule pressure. The idea that a schedule is God leads to infinite defects, as explained above. Also the principle that a schedule must be ambitious so that the development team will work hard is severely misguided. By working the development team so hard for so long, we burned out the team and lost more in the long run. The total losses from this mismanagement are not even known yet since this will affect the productivity of the team members for years to come, as well as affecting who they choose to work for in the future.

Dependencies on other teams. When we are dependent on another team for some component, any changes in their schedule or the quality of their product similarly affects us. Opus was dependent on too many teams for too many components. The tools group supplied us with SDM and the interpreter. In both cases we could have done much better

[7] November 1985, January 1987, October 1987, April 1988, June 1988.

developing them ourselves. We were dependent on Mac Word for tables, APOs, pageview and other technology. When they slipped, so did we.

Ancient tools. Discussed below, but our tools are out dated and not up to the task of developing an application like Opus.

III. Tools

The tools we have today are way out of date with the platforms we are working on and the products we hope to build. Many times during the Opus project we out grew our tools and we had to work around them or try to get them upgraded. Ilink was mostly inoperable for about a year. We exceeded the defined symbol space and had to get many tools patched or had to strip symbols out of our map files. Special versions of the debugger had to be developed to allow us to even run, since there was not enough room in memory to fit our application, the debugger and our symbols. On a pretty regular basis one or more machines would not be able to compile one or more modules because of insufficient memory, we would have to try to increase the available memory or strip down some header files. Moving to OS/2 for builds will help that situation.

After much debate, all of our developers finally got second machines. Everyone who has them feels that having two machines helps their productivity. It is unfortunate it took so long and so much begging to get them.

Our debugger and the problems of mixing PCODE and native code are a big handicap. Most developers debug using a version of Opus which does not have CS Native. That combined with the ability to dynamically turn off the hand native code (replacing it with PCODE) made debugging much easier since you could stay in the PCODE debugger almost all the time. Several people have complained about the failure of the PCODE debugger to show proper stack traces when there is mixed PCODE and native on the stack. Future tools should make using mixed PCODE and native easier.

We need a tool that will generate directly useable native structure declarations from C header files. The CS compiler has an option that tries to do this, but what it generates for bit fields is not useable. We would have avoided several hard to track down bugs if our .INC files had been generated automatically with such a tool.

The compiler should be more stringent (or have the option to be) about implicit casts and other type problems. A utility like LINT or a compiler option of that sort would have saved us some pain, especially during the many Mac Word ports.

There have also been many requests for a true source level debugger or a debugger with the capabilities and/or interface of Codeview.

IV. Technical Issues

With a project of this size, there are going to be things done well and things done poorly. The general consensus seems to be that more was done poorly than well on Opus.

Successes

Selectable Native, Pcode, Verify versions. One of the biggest time savers were the *Use C Versions* dialogs which allowed the user, at runtime, to select whether to use the C (PCODE) version of a routine, the hand-native version of the routine or to run both and have the results compared. This made it almost trivial for any tester or developer to narrow down bugs as to whether they exist in just the hand-native code or if they also exist in the original C version. (Most routines which were hand coded in Opus were maintained in their original C version also; both versions were linked into the debug EXE).

Forcing allocation failures. This idea was stolen from Excel and worked out extremely well. The user could choose to have *n* allocation processed normally then *m* subsequent allocations forced to fail. Windows and memory allocations were separated out (not clear that they needed to be) and controlled independently. A set of dumps to the user's COM port would tell them how many allocations of each type had actually been processed. To make debugging

bugs reported using AllocFail easier, a DebugBreak() could be optionally forced (based on a user's debug preference) every time an allocation was forced to fail. By using AllocFail the testing team has gotten Opus' out-of-memory recovery code into pretty shape (better than Mac Word).

Memory/swap area usage. Later I will say that Opus uses too much memory, but here I want to say that the ethods that Opus uses to allocate memory are very good. We take the best advantage of EMM that we possibly can, but we are a "good app." in that we leave some memory behind for other applications. We play a lot of games with Windows' swap fence, but this has succeeded in making our performance reasonable in a very unreasonable environment. We make intelligent choices about how much memory to allocate for swapping, for the Opus file cache, for other structures and how much to leave free.

Failures

Effective loss of BryanL. Having Bryan sick or not in the office most of the time through the last two years of the project really hurt us. Bryan understands the code we are working on and the reasons for the designs better than anyone else. Nothing could have been done about this, but it is important to acknowledge how this affected the quality of this product.

Infinite defects. This has been discussed elsewhere. It is difficult to stress all the ways our development and scheduling methods cost us because of sloppy and inefficient workmanship. In the future every line of code should under go a code review at least once and every new major data structure should under go a design review.

Code sharing. Merging code with Mac Word and later sharing some code with Mac Word was a clear disaster for Opus. Only about 20% of our code was shared (slightly higher for Mac Word since they were a smaller project) and even that code was full of #ifdefs. The code that was shared was not really the "core" of either product, merely a collection of modules that at one point had seemed shareable. The assumptions and feature sets in the two products were different enough that any change they made was likely to break us, and any change we made was likely to break them. At least once a week, our builds would be completely hosed (or theirs would) because some change they (or we) made would not even compile or link in the other project. Lack of communication between the teams that are sharing was one of the major problems. A number of bugs that we tracked down in Opus while we were sharing were found to have already been fixed in Mac Word, sometimes the fix was in shared code but had been placed under #ifdef MAC. If the code that was shared had a well defined interface (like a library) then the process might have worked better. *It is strongly recommend that we not try to share ill defined project subsets in the future.*

Unnecessary deviations from Mac Word. Made worse because we were trying to share code, the minor differences between Opus and Mac Word (headers implemented differently, CRLF pairs, rulers working differently, etc.) caused a lot of pain. Code that worked fine for them would not work in Opus. Further these differences are going to cause problems in the future since we will have to resolve them in the user interface for Pyramid.

Interactions of features. The biggest source of bugs was not any one feature of Opus but the interactions of two or more features. Tables and fields, discussed below, is a prime example. Not enough thought went into the designs of these features to assure that they would work well with each other. Design reviews in the future will attempt to address this.

Over specialization of developers. The developers on this project were too compartmentalized. We each had a set of features we really knew (generally because we wrote it or ported it) or that we at least knew better than anyone else. This might have been an unavoidable outgrowth of the size of this team and the duration of the project, but it certainly contributed to the problems with feature interactions and the duration of our stabilization phase.

Memory consumption. Opus is a hog. Low memory messages are going to be one of the things users run into the most and they are not going to understand[8]. We have too much that needs to grow that we keep in the near heap (like list box entries) and EMM has us too constrained by it's 16K limit. Getting away from Real mode will help this situation some, but work needs to be done in the future to understand where and why we use memory and to determine if there are ways that usage can be reduced.

MS-PCA 1295262

CONFIDENTIAL

[8] For more information on how Opus does use memory, you may want to refer to the Opus Memory Management document of 19 September 1989.

Speed. Opus, despite all of our efforts, is still much slower than we would like. Our code reviews showed that we can get minor speed improvements out of just about any section of the code, we just need to review more of it to realize those improvements. We wrote a lot of hand native code, which helps our speed, but algorithmic improvements are much better leveraged.

Tables, fields, tables & fields. Tables work by overloading paragraph properties. A bit indicates whether a paragraph is within a table. Another bit indicates the trailer paragraph (where the row properties are held). End of cell is determined by a special character as end of paragraph.

Problems: Cell markers and trailer properties must agree or you die. This changes some very basic assumptions at low levels in Word (you can't just delete anything). It's slow to determine the boundaries of a table and its cells: you have to find paragraph bounds forward and backward and you have to fetch the end-of-paragraph character looking for the cell marker. You can overflow the row width which is stored as an integer in the trailer. All these properties are tacked onto the end of the paragraph properties structure, making it bigger (doesn't affect file size, but it does increase the largest possible PAP, which forced Mac Word to change the file format).

Fields usually consist of two text sections (codes and results) punctuated by three separators. A field may consist of just one text section and the outer separators. Each separator has a PLC entry.

Opus had many problems because of the stream nature of fields (they will jump and allow the visible text stream to appear to start at a latter point) and the paragraph nature of tables. If a field wanted to skip into a table or out of a table from within, the display code would choke.

Selection and cursor keys. Pageup then pagedown: you're not guaranteed to be where you started. Cursoring through a document gives unpredictable results. Some operations can only be performed on some selection types (e.g. you can't operate on discontiguous text). Arbitrary operations performed by Word in response to user commands can result in illegal selection states. Highlighting selections in the macro pane has innumerable bugs. Mac and Win disagree on the best model for ensuring that the selection is visible.

Page view display model. Currently all text in page view is drawn on the same layer, even if it overlaps. Where overlaps do occur, the drawing order can be random, resulting in garbage. We try to avoid overlaps in obvious cases by restricting the rectangles of the header/footer and near APOs, resulting in clipped text. This is overly restrictive.

CRLF. In Opus paragraph marks are represented by two characters (carriage return-line feed) except for section marks which are only one character (chSect) and possibly other format files (Unix files with only line feeds and Mac files with only carriage returns). In Mac code all paragraphs end with a single character. Our model caused many problems and complicated code on top of the complications arising from being different from Mac Word.

Macros. The macro language became functional very late in the project. This caused problems for development since the necessary hooks had to be retrofitted into every command (a very error prone process). It caused problems for testing since they could not use the macro language to automate tests until the last year or so. The interpreter we have now was originally written by the tools group, but was turned over to us to maintain because no one else wanted to use it. Both tokenization and the interpreter could benefit from an overhaul.

Outlining. Outlining and renumbering require an entry in the PLCPAD for each paragraph. For outlining at least, it's clear that this is not necessary: you only need entries for the paragraphs that are being displayed or possibly just the transitions between levels. For numbering, the answer is not so clear. This use of memory restricts the size of a document that you can go into outline view with or have autonumbers in (about 800 paragraphs).

Format & display. These need to be made reentrant. Very late in this project (one of the last two bugs fixed) we discovered that we could crash because of our failure to be reentrant. Apparently Write had the same problem (so why didn't we know about it?).

Screen vs. printer units. On the Mac a screen unit equals one printer unit. This makes layout, display and printing all nicely interchangeable. Under windows they are not the same unit and you have to determine at run time what each unit is (the printer unit may even change during a session). This difference caused many bugs and difficulties in sharing code.

MS-PCA 1295263

CONFIDENTIAL

V. Interaction With Other Groups

In the following sections I try to discuss both the ups and down of our interactions with these groups and ways in which it seems these groups could improve. This is not meant to be a bashing or an attempt to focus blame for Opus' problems on others, we all have some problems and have room for improvement.

Program Management

Relations with Program Management have generally been very good. Adrian was always available and willing to discuss features and implementations. His method of polling when he wanted something was effective.

The Opus specification really never was. We should have had a real spec sooner and it should have been more complete and better maintained. The addition of features and the modification of existing features is inevitable considering our market and the need to do usability testing. However, when adding things or making changes it is important that we consider whether the change is worth the delay it will cause and we need to be realistic about what that delay is going to be.

The attitude that "program managers do no work" is a bit dangerous. There were often problems arising from program management dictating tasks, especially to the PAs in development. If Program Management has a lot of non-development tasks that need to get done (benchmarks, disk images, etc.), perhaps they should hire their own people to do them. Development has hired PAs to do development related tasks.

In order to keep the entire product team better informed, it was suggested that it would be great if Program Management would keep and distribute weekly minutes from the leads meetings, instead of relying on the individual leads to pass that information on to their people.

It was also suggested that developers should spend more time using other Microsoft products (with which we need to be consistent) and with our competitors products. To keep this from being a time sink, it was suggested that more product presentations (like the one done for WordPerfect 5.0 several years ago) be done and even video taped. A possibility would be to have different people (both developers and Program Managers) become experts on different products then present those products to a group. This is something that Program Management should coordinate.

Finally, it is very important that Program Management continue to talk to the individual developers about features. It is important to remember who is going to be writing the code; they are the ones who really own this product. Through the course of the Opus project, I think Program Management did very well in this regard.

Testing

Overall the testing team did an excellent job testing Opus, sometime under very trying circumstances. The developers who came over from Mac Word felt that the environment that the Opus testers had to work in was much harsher than it was on Mac Word. There seemed to be more attention on politics and rivalry then on working together to create an awesome product.

Communication could have been better between the teams, especially earlier on in the project. Forums such as the code reviews and testing strategy meetings really helped improve communication and get people talking about areas of the product. These forums should be continued in future projects. It is felt that it would help if development supplied testing with a list of developers and their areas of specialization (especially on a project like Opus where everyone was very specialized). It would also be good if testing provided a similar list to development.

Development needs to take more responsibility for the testing of the product. A prevailing attitude by development was that somehow the testers would learn all about the features and all the necessary test cases but there was no mechanism in place to do get this information to them. To help with communication and to assure that all areas of the product are being covered, it was suggested that developers be required to supply some information every time they check something in. That information could be TRD entries, test plans or completed macro tests for the test suite that would exercise the area changed. The main point is to force the developer to think about how the feature needs to be tested and what the test cases are.

Another suggestion to improve tester-developer relations would be to team testers and developers up 1-1 or 2-2 to work on features (possibly with one or more PAs also working on the teams). This would enhance communication, team spirit and help test important features.

Testing of the peripheral parts of the product was lacking. Setup was not really tested until the last minute and hen by only one tester who was not available at some crucial times. The sampler was mostly ignored by testing despite pleas to have it tested. Despite it's being in a TRD, the DEMO version did not get any attention until after we shipped (which was really too late if anything had to be changed). Testing should think more about these items and schedule time for them.

There was also some concern about the testing effort at the end of the project. It seemed like everyone (testing and development both) had decided it was time to ship the product. This didn't appear to be a conscious decision or an attempt to ship by a date, it just seemed that everyone felt it was ready (and perhaps it was).

International

Overall relations between international and development went well, though there are some things left to be desired.

Ideally an international engineer would be put on a project like Opus from the beginning. They should have a hand in the spec, which should identify exactly what is going to have to be localized. It was very unfortunate that by the time Jurgen started looking at Opus, everything had basically been done. It was also unfortunate that very few people on Opus development team had any feeling for what international would need. In the many areas that Jurgen found deficient from his point of view (both for the Z version and for localized versions), we either had to stop and redo something (as discussed under technical failures) or we had to say "NO," making the product less international-friendly. Considering the amount of sales we can expect from international sources, this is a crying shame.

Having Jurgen working in the same building with us has been a great help. This has both made us more aware that international is there, so we are more likely to do things in ways that make their life easier, and it is more convenient to just drop in with questions about how they would like things done or other issues.

The size of the international build kit for Opus (our entire source tree and tools) and the amount of time required to build a localized version (3-4 hours) is a major problem for international. This makes it impractical to have any localization done off site (i.e., Ireland) and the marginal expense of adding an additional language is far too high. Development, international and the tools group need to work together to make this more reasonable for future products.

Our dealings with other facets of international were not entirely rewarding. Sometimes those working on the sampler and on the international disk images relied too heavily on our resources (especially Laurel and Brandy) and seemed to make no attempt to solve problems for themselves. This has caused some hard feelings. International should look at the technical knowledge of their people, especially as our products become more and more complex, and should take the time to try to solve problems themselves before falling back on us.

User Education

Print Based

The biggest problem with user-ed was our schedule. They were continually trying to meet a schedule which turned out to be way to early. The user's reference was sent to the printer over a year ago, naturally the product changed enough during that year that it is now way out of date.

Having Russ working directly with the development group was great. It seems likely that he knows this product better than anyone in user-ed has ever known a new product before it shipped. Once Russ left to go to Preis, the relations between development and user-ed changed a lot. Most of the interaction now became between program management and user-ed, and there is some feeling that program management could have done a better job at keeping them abreast of changes.

Development should schedule more time to conduct technical reviews of documentation. A pretty good review was done on an early draft of the user's reference by all of development, but individual developers did not have any subsequent chance to review it nor did they have the opportunity to review any of the other documentation.

The technical reference was a big disappointment. It took many, many revisions before anything resembling an accurate document was produced, and that was after Brad Christian and Adrian practically wrote it themselves. User-ed definitely needs to have writers who are more technical.

On-Line

In general, communication between the on-line user-ed groups and Opus development went very well. There were times when user-ed seemed to lean too heavily on development. They would assume problems they were encountering were our problems without fully investigating them (and often they proved to be problems on their side) and they would often ask for changes in Opus without really thinking about their cost and benefit. Along the same line, many members of the CBT group were not good about using the prescribed communication path (funnelling everything through David Innes) and would instead go directly to Rosie. This was very disruptive of Rosie's work, especially since David could have resolved a good portion of the issues without ever involving development.

It would have helped a lot to have understood the hooks required for CBT much earlier in Opus' development and for those hooks not to have had to change so much. Changes of this type were more complicated since they would often require changes by four different groups: Opus development, CBT authoring, DOT development and SDM. Coordinating these changes and making them work didn't go too smoothly.

A RAID database was set up to track CBT and Help problems, but they were not really used until the end of the project. Making better use of these standards would have kept problem reports from getting lost and simplified communication between the groups.

The biggest problem with the on-line products was the lack of testing support. CBT needed a lot more testing than it ever received and a lot of what it did receive was done by one of our developers. This is a big area that needs a lot of testing. The code involved might be small and well tested (which, in fact it, was not) but the authoring also needs a lot of testing. User-ed should either arrange with testing to have a lot more testing done for future products or they should provide the man-power themselves to do it.

Windows Development

Our interactions with the Windows development team were very mixed.

Direct interactions with developers (especially David Weise and Bob Gunderson) were fantastic. David was very responsive to our pleas for help when we were running into bugs that seemed to be Windows related that we could not track down. He spent some late nights and long hours tracking back in forth between our code and his. The result of this attention was fixed bugs in Windows and in Opus, and not a lot of finger pointing and saying "it's not my bug."

Our experience trying to report bugs against Windows 3 has been quite the opposite. Our developers try to determine if a bug is in our code or in Windows code before they ever consider assigning a bug to Windows (though we are wrong sometimes). The Windows group was not very cooperative at all about those bugs that we sent them. A very high percentage came back WONT FIX saying that it "obviously" was not their problem, but with no explanation about what was going on or any advice about what we could do. Another large portion of the bugs came back NOT REPRO, but in no case did they contact anyone who had reported or investigated the bug to see how to reproduce it. Our development team has a rule: you can never resolve a bug NOT REPRO without getting the consent of the person who reported the bug. It is, after all, often the case that some condition needs to be satisfied to reproduce the bug that may not be listed in the actual report. After seeing how they treated our bugs, I am worried about the quality of the product they will produce.

We had a lot of problems with printer drivers with Opus. We designed out printing around the HPPCL driver and the features it supports. We were surprised to discover, late in the project, that many drivers do not support many things that the HPPCL supports or, worse, supports them in different ways. When we reported bugs against these inconsistencies we were usually told that they were by design or that they were not interested in fixing them. I think it is

important that a platform like Windows try very hard to make all of the drivers present a uniform appearance to the applications (isn't that supposed to be one of the great advantages of Windows?).

Development Support

Overall development support has tried to be very responsive to our needs, though it has seemed like we often had to escalate the problems before we could get that response. This is probably attributable to the many clients the different groups have to serve.

One general comment about DS: the quality of the product often leaves much to be desired. Frequently what we would get from them would be very buggy and it would take several passes before we would get anything usable. DS should look hard at the principles of zero-defects and try to determine how they can be applied to their products.

Scott Randell was always a great help. He is an excellent resource for simple information and to assist in debugging hard problems (like the AT&T bugs that he tracked down for us).

The SDM team was pretty good about responding to our needs, though it did take quite a while to eek out the performance we needed. We were also very cautious about accepting any releases since, for most of the time we used SDM, every new release would break several things. It was not until the last several months that we started getting reliable releases. It also seems like it was a poor idea that SDM, which is supposed to be shared by all of our applications, was originally written by new hires.

DOT development seemed way under staffed but despite that they were very responsive.

Help development, by contrast, was not at all responsive. We found that we had to do a lot of their debugging for them and even when we pointed out their bugs they didn't fix them. We finally had to go through Jeff Raikes to get them to fix a number of problems that were holding us up.

Product Marketing

The development team did not deal much with Marketing. Maybe that is itself a problem. There is an impression that Marketing does not really understand Opus very well (it's capabilities and how to use them) and may misrepresent it. Development as a whole also didn't get to hear too much about the great progress that was being made on corporate accounts, advertising, etc.

Product Support

Having Michel Girard working with us went very well. Michel learned Opus quickly and now is quite an expert. He also shared his concerns with us, helping us make a better product.

The PSS Bug Party that was held was a great success. The developers went away from that feeling good about Product Support and about Opus. It also helped forge relations between the development team and the people who will actually be supporting the product (what a novel idea). We hope that there will be more opportunities in the future for development to interact directly with Product Support.

There has been some concern about how effective Product Support has been in getting feedback to us. There seemed to be very few beta350 bugs that actually made it onto the Opus bugs list. There have also been reports that our customers might not be too happy with some of the support that they have been receiving. The specific example of our lack of presence on the forums on Compuserve was mentioned.

Appendix I
A Brief History of the Opus/Cashmere Project

Month	Current Events	Facts
AUG 84	Memo written describing a demonstration version of "WinWord" to be developed to demo at COMDEX in November 84. Pat Tharp and Chi-Chuen Chan start work on Write. Dan Lipkie is Write project lead.	
SEP 84	Bill Gates asks Russ Borland to join the Cashmere team. Cashmere is supposed to ship within a year. Cashmere is organized as a special business unit under Richard Brodie. Jonathan Prusky is working on initial concepts.	shipdate: sep 85 SDEs: 0 FTEs: 0 (Write: 2)
OCT 84		
NOV 84	Russ moves his office to adjoin the rest of the team. Russ spends time working on documentation strategy and reviewing design concepts developed by Richard and Jonathan. Bryan Loofbourrow joins Opus team, soon moves on to Write (as "training" for Cashmere).	shipdate: sep 85 SDEs: 0 FTEs: 0 (Write: 3)
DEC 84	Many meetings held with Bill and Charles Simonyi (both of whom were in the same hallway as development) to discuss strategies and specifics (on going through next several months).	
JAN 85	Bob Matthews joins as Project Lead, heads up Write effort. Development work proceeds on Write. No development being done yet for Cashmere.	shipdate: may 86 SDEs: 0 FTEs: 0 (Write: 4)
FEB 85	Design meetings held discuss many features: ruler, ribbon, short menus, tab stops, hanging indentation, interface to mail and flat file manager (last two eventually scrapped) (ongoing).	
MAR 85		
APR 85		
MAY 85	Marc Singer starts work on special Windows controls. Bob Zawalich joins the Write team.	SDEs: 0 FTEs: 0 Interns: 1 (Write: 5)
JUN 85	Bill Aloof joins the Cashmere team to work on email.	SDEs: 0 FTEs: 0 Interns: 1 (email: 1) (Write: 5)
JUL 85	Yoshito Yamane joins the Cashmere team Prototyping work begins based on a forked set of Write sources.	SDEs: 1 FTEs: 0 Interns: 1 (email: 1) (Write: 5)
AUG 85	Rosie Perera joins the Cashmere team. Bob Zawalich comes from Write to work on Cashmere.	SDEs: 3 FTEs: 2 Interns: 1 (email: 1) (Write: 5)

MS-PCA 1295268

CONFIDENTIAL

SEP 85	Prototype A demoed to Bill (includes ribbon, print preview and split windows). David Bourne and Brad Christian join the Cashmere team. Bob Zawalich makes a checkin that reduced cashmere.exe from 243K to 222K! Upgraded make process to use CMerge version 3.00.16. Have code under #ifdef MOCKUP (for prototype), #ifdef SAND (for Mac only - Sand was the old codename for the Macintosh), #ifdef MEMO (Memo was the code name for Windows Write).	shipdate: sep 86 SDEs: 5 FTEs: 3 Interns: 1 (email: 1) (Write: 5)
OCT 85	Work proceeds toward Prototype B. Compare Versions written — basically never changed.	SDEs: 5 FTEs: 5 Interns: 1 (email: 1) (Write: 5)
NOV 85	Windows 1.0 ships! "Real" development work begins. Bryan rejoins Cashmere team. Chi-Chuen and Pat join the Cashmere project after Write ships to work on email. Bob Matthews leaves to work on Windows; Richard Brodie assumes management of Cashmere. SLM 1.0 is used for the first time. Bryan starts the first MacWord merge.	SDEs: 7 FTEs: 7 Interns: 1 (email: 3)
DEC 85	Marc returns to school. Bryan ports MacWord 1.05 internals to Cashmere. Converted to the CS compiler.	SDEs: 7 FTEs: 7 (email: 3)
JAN 86	Bryan checks in conversion from Write data structures to MacWord data structures.	
FEB 86	Project not fully recovered from first merge until late February.	
MAR 86	Bryan reports that it takes 2 hours and 38 minutes to do a mass make (but remember that was on an AT!!). Moved from having a one-directory development environment to having subdirectories. CSHARE (which later became wordtech) was introduced. Added passwords to our mickey shares. \\mickey\slm's password was mori (Yoshi's mother's maiden name) and for \\mickey\private it was failte (Bob's contribution, it means "welcome" in Gaelic).	
APR 86	Pat Tharp moves to Systems. Bob works on customizing the "standard" dialog manager. Kanji Write is proposed — a two week task. PC Word 3 ships. Greg Slyngstad joins the Cashmere effort.	shipdate: jan 87 SDEs: 7 FTEs: 7 (email: 2)
MAY 86	Bill Aloof leaves. Jonathan Prusky asks development to start thinking of a real name for Cashmere. He said that it may end up being called Windows Word unless we think of something more significant. Bryan eliminates the MOCKUP flag once and for all. Ilink is used for the first time; Bryan reports "breath taking performance increases." CMAKE.EXE is written to perform builds. Brad Christian works on Formulas (one thing that was never *completely* rewritten).	shipdate: apr 87 SDEs: 7 FTEs: 7 (email: 1)
JUN 86	Peter Jackson joins Cashmere. Code review held on (first) implementation of tables. Richard starts work on RFT (later RTF).	SDEs: 8 FTEs: 7 (email: 1)

MS-PCA 1295269

CONFIDENTIAL

JUL 86	Richard Brodie resigns, Ford Martin joins the team and becomes Project Lead, Bryan becomes Technical Lead. Code name changed to Opus. Opus is reorganized as a normal application instead of its own business unit. No spec exists yet, only an "overview" which describes general features & interface. Text files are used to implement a bug list. Bookmarks are implemented.	SDEs: 9 FTEs: 8
AUG 86	Email project is dissolved, Chi-Chuen joins Opus effort. Brad Christian is writing macros and running them with his version of the interpreter. "That's ok, it can be a macro" is heard for the first time. All references to Cashmere are changed to Opus except project name.	
SEP 86	Version 0100 (A) released. Greg Cox joins Opus to work on hand native coding. Bryan starts work on the "unified field theory." CSCONST feature is implemented in the CSL compiler.	SDEs: 10 FTEs: 8
OCT 86	Opus shown with "come back next year" sign at Company Meeting. Berke, our first '386, is provided; people are asked if it would be useful to have more.	SDEs: 10 FTEs: 9
NOV 86	Recently functional features: styles, glossaries, ruler, mac-in-the-box Considering cutting features in order to ship in nov 87 Yoshi finishes Kanji Write after many more than two weeks.	shipdate: jan 88 SDEs: 10 FTEs: 9
DEC 86	Version 0200 (B) released. Switched over to Windows 1.5 (overlapping windows). Filenames and line numbers appear in asserts for the first time.	
JAN 87	Version 0300 (B1) released. Shipdate pulled in by eliminating buffer tasks. Testing effort begins. Decision made to allow editing in Preview using Charles' Mac Word technology. Core code for fields functional. MacWord 3.00 ships! Bryan sets aside his work on tables and starts another MacWord merge.	shipdate: dec 87 SDEs: 10 FTEs: 9
FEB 87	Phil Fawcett reports the first bug in the Opus database.	
MAR 87	FormatLine hand-native coded (first native code for performance) in 4.5 *long* days. OPEL spec completed. External conversion hooks defined. Opus dialogs changed to be consistent with Excel. Drop-down listboxes implemented. Background pagination working and "looks impressive." Macro record/playback "nearly working." Yoshi coins "crushes." Search & Replace functional. Opus prints a one line document.	shipdate: jan 88 SDEs: 10 FTEs: 9

MS-PCA 1295270

CONFIDENTIAL

APR 87	Version 0400 (C) released. All tables tasks moved to MacWord tasklist. Headers/Footers working. Looking for features to cut to give high confidence in Jan 88 ship date. Date slipped because more time than estimated was spent fixing bugs for Release C. Non-debug make implemented. Glossaries can now be saved. Annotations feature speced. Spec meeting with Bill, need to consider Draw functionality. New Windows with dramatic performance improvements unveiled. RTF functional, being extended for Opus specific features. Contest to name Opus proposed. Paul McKee provides the Opus icon. Double underlining hooked up for the first time.	shipdate: dec 87 SDEs: 10 FTEs: 9
MAY 87	OPEL edit/debug environment almost complete. Changes made to take advantage of ExtTextOut. Search/Replace formatting completed. Version 0500 (D) released. Bug fixing for testing release took longer than anticipated.	shipdate: jan 88 SDEs: 10 FTEs: 9 114 bugs total
JUN 87	Bruce Wine starts on PRDDRV. 300 entries so far for Opus naming contest. Revised spec distributed. Reviewing spec and schedule for remaining open issues in preparation for schedule review. Merging styles and fonts for cut/paste implemented. Field expression (=) implementation complete.	shipdate: feb 88 SDEs: 10 FTEs: 9 Interns: 1
JUL 87	"New SDM" agreed to. Fields format switches implemented. New OPEL spec. Pictures supported in clipboard and file format. Footnote insertion and editing implemented. MacWord 3.01 ships! Stephen Arrants starts on the Technical Reference. Eric Geyser joins Opus team.	shipdate: none SDEs: 11 FTEs: 9 Interns: 1
AUG 87	Version 0600 (E) released. Schedule Review conducted with Jeff Harbers; resulted in negligible change in end date. Adrian Wyard joins Program Management on Opus. Herb Klopfenstein joins Opus team. Eric Geyser leaves Opus to work with ADC. Keyboard Accelerators shown on menus. Opus naming choices narrowed down to "Maxxam" vs. variant of Word. "final spec" distributed.	shipdate: feb 88 SDEs: 11 FTEs: 9 PAs: 1
SEP 87	Draw functionality dropped. Annotations implemented. DDE server/client functional. Iconbars put in split panes. Thesaurus work started. Indexing and outlining features implemented/ported. Usability testing done.	shipdate: apr 88 SDEs: 10 FTEs: 9 PAs: 1

MS-PCA 1295271

CONFIDENTIAL

OCT 87	Versions 0700 (F) and 0800 (F1) released. PC Word 4 ships! Spec Addendum distributed Jeff Ruttenbeck joins Opus team. Brad Verheiden is "loaned" from Word Technology. Rick Saada assists in porting Document Management from PC Word. New EL interpreter installed. Bryan begins MacWord Code merge. Ford works on Write bugs. Formulas are converted into fields. InsertFile and InsertField are implemented. First draft of User's Reference reviewed. Opus not even mentioned at Company Meeting.	shipdate: may 88 SDEs: 11 FTEs: 11 PAs: 2
NOV 87	Version 0900 (I) released. WinExcel ships! Windows 2.0 ships! Dan Porter joins Opus team. Schedule padded to reflect past trends (with no net effect). Discussions begin with MasterSoft regarding conversions. PC Word 4 and Pageview are demoed at COMDEX. TOC generation and repeat implemented. New SDM installation begins. David Bourne coins "poc voon" (made popular by Bob). MacWord code merge continues.	shipdate: may 88 SDEs: 11 FTEs: 10 PAs: 3
DEC 87	Opus Visual Freeze version released. MacWord code merge and New SDM installation continue. Slippage due to code merge being reestimated and more time added for PageView and Tables. Program Management working on Macro Language, Tables and PageView specs.	shipdate: jun 88 SDEs: 11 FTEs: 10 PAs: 3
JAN 88	Versions 1000 (J) and1100 (K) released. Herb leaves for active duty in the Air Force. Performance campaign is conducted. Attempt made to fix all known bugs in order to stabilize. Doug Timpe and Mike Hopstad join the Opus team. InfoWord gets hold of a confidential overview from one of the corporate emphasis participants, writes a front page article. Marketing goes silent to prevent repeats.	shipdate: jun 88 SDEs: 11 FTEs: 10 PAs: 2
FEB 88	Version 1200 (L) released. MacWord has APOs, Tables and PageView working "to various degrees." John Parkey joins marketing effort. Specing the details of the macro language continues. Conversion of dialogs to new SDM continues. Double clicking becomes a big issue with Bill; many hot spots added.	shipdate: jun 88 SDEs: 11 FTEs: 10 PAs: 4
MAR 88	Version 1300 (M) released. schedule slip because of MacWord slip, additional time added to final merge estimate and time spent on performance work. Sample templates and files to include with the package are specified. Spec addendum 3 released. All dialogs converted to new SDM (not yet fully functional); old SDM removed. Preliminary spec for SETUP completed. Jeff Harbers tells development, "you are the worst team in applications development."	shipdate: aug 88 SDEs: 11 FTEs: 10 PAs: 4

MS-PCA 1295272

CONFIDENTIAL

APR 88	Versions 1400 (N) and 1500 released. Bryan takes a medical leave of absence (CFS); Peter becomes Technical Lead. Opus presented to Air Force for Desktop III. Greg Slyngstad married. PAs lose their "t-" prefixes and become real people. Final code merge conducted; checked in state hosed for two weeks. Peter starts buying dinners.	shipdate: sep 88 SDEs: 10 FTEs: 9 PAs: 4
MAY 88	Print Preview and Thesaurus demoed to FAA. Spec addendum 4 released; includes TIFF and color and user interface changes. Anthony Cockburn joins Opus to work on performance. Final set of featues now working "to some degree." Greg Cox leaves Opus to form conversions group. George Hu starts work on SETUP. Macro Language spec revised.	shipdate: oct 88 SDEs: 10 FTEs: 8 PAs: 4
JUN 88	Version 1600 released. Another Visual Freeze version released. MacWord merges our changes back; "true" sharing of wordtech begins. Ford goes on a leave of absence; Bryan returns from medical leave and becomes Project Lead. Tony Krueger and Alan Ezekiel join the Opus team for the summer. Almost all features in place; development attention turns to bug list. WinWord demoed to OEMs; Jeff Raikes describes it as "Word processing technology." Publishing character support gets speced. Trevor Zawalich born. Jurgen Leschner starts looking at Opus for localization.	shipdate: oct 88 SDEs: 10 FTEs: 9 PAs: 4 Interns: 2
JUL 88	Version 1700 released. Slip occurs due to release prep time, new tasks, instability after the merge and the time taken on the macro language. Jeff Sanderson takes WinWord on a Press Tour. Final review of User's Reference completed. Macro spec updated. Yoshi leaves to return to school. Bryan implements "rational scroll bars" so that we can be consistent with Excel. Out-Of-Memory handling is reformed. Majority SPRM is implemented. TIFF support added. Mergeformat changed to merge table and picture formatting. caCharBlock concept implemented.	shipdate: nov 88 SDEs: 9 FTEs: 8 PAs: 4 Interns: 2
AUG 88	Version 1800 released. Feature Complete! Apps Reorg takes place. Interfaces to draw app and to file converters implemented. First discussion held on how to translate fields through RTF for INTL, no good ideas. Word SIG Summit held, everyone enjoys a day on the lake. Program Review held with Bill.	shipdate: jan 89 SDEs: 9 FTEs: 8 PAs: 4 Interns: 2

MS-PCA 1295273

CONFIDENTIAL

SEP 88	Version 1900 released. Anthony leaves Opus to work on Windows Works. Laurel Brown joins the Opus team. Macro language code reviewed. TIFF grey scale support implemented (was not included in Feature Complete). Scaling and cropping of TIFF images added. REVIEW comments assigned to individuals. Changes made to move more structures from NEAR to FAR/EMM memory. User's Reference goes to the printer. "Final SDM" incorporated (new list-box and dropdown look). Tony and Alan return to school. Bifurcated Replace/ReplaceCps implemented. Macro test suite set up. Added AllocFail test abilities.	shipdate: jan 89 SDEs: 8 FTEs: 8 PAs: 5
OCT 88	Versions 2000, 2100 and 2200 released. Code Complete! Opus demoed at Company meeting (election demo). Krishna Mukherjee joins Opus team. International sensitive sort and string compares implemented. Failure to do revision marking in tables discovered.	shipdate: feb 89 SDEs: 9 FTEs: 8 PAs: 5
NOV 88	Versions 2300 and 2400 (Beta I) released. Opus demoed in a back room at COMDEX. Serious performance work starts.	shipdate: feb 89 SDEs: 9 FTEs: 9 PAs: 5
DEC 88	Version 2500 released. Slip due to high bug find rate relative to bug fix rate. Hooks for graphic filters implemented. Development exceeds net bug fix goal (50/week) by factor of two for two weeks. Attention turns to SDM speed.	shipdate: mar 89 SDEs: 9 FTEs: 9 PAs: 5
JAN 89	Versions 2700 and 2800 released (2600 never existed). MacWord decouples wordtech sources from Opus. Performance Review meeting held with BillG. Adrian switches his month and year resulting in an ADL date of Sep 5. OBU moves to building 5.	shipdate: may 89 SDEs: 9 FTEs: 9 PAs: 5
FEB 89	Version 2900 released. Again, slip due to low net fix rate; new date called "atypically unaggressive." Testers loaned to MacWord. Peter assumes Project Lead responsibilities. Latest SDM implements more windows controls itself; performance gains disappointing.	shipdate: jul 89 SDEs: 9 FTEs: 8 PAs: 5
MAR 89	Versions 3000, 3100 and 3200 released. Dan Porter leaves to work for Summation. Dennis Andersen joins Opus team; will be working with Mike on printer driver problems. First complete benchmark document produced. Krishna gets 4x performance improvement in Macro Detokenization. Stephen Maguire assumes responsibility for SDM performance. Charles investigates "RCODE" at Bill's request. Automatic installation of document converters implemented. Performance work declared complete.	shipdate: jul 89 SDEs: 10 FTEs: 8 PAs: 4
APR 89	Version 3300 released. MacWord 4.0 ships! PC Word 5.0 ships! Bug Campaign I reduces active bug count below 150 with 600 bugs resolved in 3 weeks. David Luebbert, DavidMcKinnis and Tom Saxton start working on Opus part time.	shipdate: jul 89 SDEs: 13 FTEs: 9 PAs: 4

MAY 89	Versions 3400, 3500 and 3600 released. Most of MacWord team join the Opus effort. Dennis removed from our roster. Jeff Rutenbeck leaves to take a job in Colorado. Chris Mason, Sylvia Hayashi and Doug Scott come over from MacWord. DavidLu, DavidMck and TomSax go to full time. Brandy Thorp joins the Opus team. Simon and Chip start as summer interns. ADL report shows -0/+13 week range on the ship date. Decision made to hand-code some of the table display routines.	shipdate: aug 89 SDEs: 13 FTEs: 10 PAs: 6 interns: 2
JUN 89	Version 3700 released. All automated macro tests pass for the first time. Amid continuing slips, development shifts focus from quantity to quality. Laura and Danny start as interns. Kodak and American Airlines distribute 3300 to their users. SDM 2.21 installed, realizes a 10-25% improvement. All native coding to be done is completed. REVIEW count down to 28.	shipdate: sep 89 SDEs: 13 FTEs: 12 PAs: 6 interns: 4
JUL 89	Versions 3800, 3900 and 4000 released. Jack, Clay and John join as interns. Concept of code ownership implemented. WinWord 1.1 plans discussed. Mike Hopstad leaves. Tony Krueger joins the Opus team, again. Attempts made to install the RCODE compiler.	shipdate: oct 89 SDEs: 14 FTEs: 12 PAs: 5 interns: 7
AUG 89	Versions 4100 and 4200 released. Cancun incentive announced. Code reviews wind down, focus shifts back to the bug list. Bug Campaign II reduces the active bug count below 100. RCODE canned. Macros run under Windows 3.0 for the first time. Display speed slows down for two releases then picks back up; never explained. Testing concentrates on automation. Laurel Brown marries and becomes Laurel Lammers.	shipdate: nov 89 SDEs: 14 FTEs: 13 PAs: 5
SEP 89	Versions 4300, 4400, 4500, 4600, 4700, 4800 and 4900 (ZBR) released. Novell net problems brought up. Development conducts bug find campaign; finds 168 bugs, half of those reported.	shipdate: nov 89 SDEs: 14 FTEs: 13 PAs: 5
OCT 89	Versions 5000, 5100, 5200 and 5300 released. Final swaptuning done. Testing does a regression on entire bug list. WinWord announced.	shipdate: nov 89 SDEs: 14 FTEs: 13 PAs: 5
NOV 89	Versions 5400 (RC1), 5500, 5600, 5700, 5800 and 5900 released. WinWord publicly demoed at COMDEX. Disks released to manufacturing 30 November.	shipdate: nov 89 SDEs: 14 FTEs: 13 PAs: 5

MS-PCA 1295275

CONFIDENTIAL

Appendix II
Project Statistics

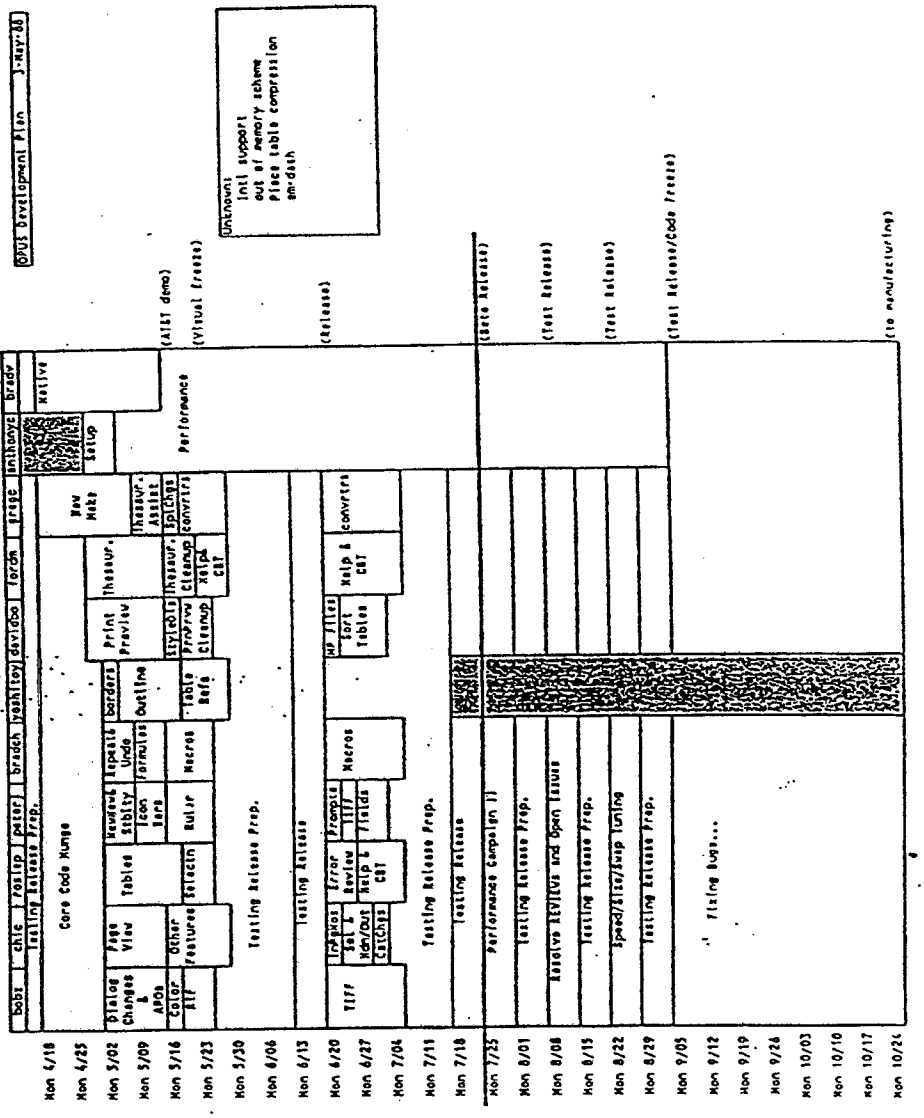
Date	Ship Date	Days to Shp dte	SDEs	PAs	Int.	FTEs	KLOC	ASM KLOC	DBG KB	FAST KB	Bugs Opnd	Fxabl Opnd	A.Bug Count	T.Bug Count	Bugs Rslvd	Rslvd Fixed	Fixed/ FTE
Sep-84	Sep-85	365	0	0	0	0											
Oct-84	Nov-85	396	0	0	0	0											
Nov-84	Jan-86	426	0	0	0	0											
Dec-84	Mar-86	455	0	0	0	0											
Jan-85	May-86	485	0	0	0	0											
Feb-85	May-86	454	0	0	0	0											
Mar-85	Jun-86	457	0	0	0	0											
Apr-85	Jun-86	426	0	0	0	0											
May-85	Jul-86	426	0	0	1	0											
Jun-85	Jul-86	395	0	0	1	0											
Jul-85	Aug-86	396	1	0	1	0											
Aug-85	Aug-86	365	3	0	1	2											
Sep-85	Sep-86	365	5	0	1	3											
Oct-85	Sep-86	335	5	0	1	5											
Nov-85	Oct-86	334	7	0	1	7											
Dec-85	Oct-86	304	7	0	0	7											
Jan-86	Nov-86	304	7	0	0	7											
Feb-86	Nov-86	273	7	0	0	7											
Mar-86	Dec-86	275	7	0	0	7											
Apr-86	Jan-87	275	7	0	0	7											
May-86	Apr-87	335	7	0	0	7											
Jun-86	May-87	334	8	0	0	7											
Jul-86	Jul-87	365	9	0	0	8											
Aug-86	Sep-87	396	9	0	0	8											
Sep-86	Nov-87	426	10	0	0	8	56	3	241								
Oct-86	Dec-87	426	10	0	0	9											
Nov-86	Jan-88	426	10	0	0	9											
Dec-86	Jan-88	396	10	0	0	9			292								
Jan-87	Dec-87	334	10	0	0	9			339		14	14	14	14	2	0	0.00
Feb-87	Dec-87	303	10	0	0	9					4	4	18	18	0	0	0.00
Mar-87	Jan-88	306	10	0	0	9					18	18	36	36	0	0	0.00
Apr-87	Dec-87	244	10	0	0	9	56	3	437		50	42	86	86	0	0	0.00
May-87	Jan-88	245	10	0	0	9					37	34	120	123	3	2	0.05
Jun-87	Feb-88	245	10	0	1	9					99	81	218	222	1	1	0.03
Jul-87	Feb-88	215	11	0	1	9					29	21	244	251	3	3	0.08
Aug-87	Feb-88	184	10	1	0	9			580	447	18	16	250	269	12	12	0.31
Sep-87	Apr-88	213	10	1	0	9	121	11	789	553	64	54	297	333	17	16	0.41
Oct-87	May-88	213	11	2	0	11					60	44	349	393	8	3	0.06
Nov-87	May-88	182	11	3	0	10					92	73	429	485	12	3	0.07
Dec-87	Jun-88	183	11	3	0	10					146	103	304	631	269	213	4.95
Jan-88	Jun-88	152	11	2	0	10					112	87	266	743	150	131	3.05
Feb-88	Jun-88	121	11	4	0	10	126	20	854	615	176	136	307	919	135	110	2.56
Mar-88	Aug-88	153	11	4	0	10					255	219	327	1174	235	184	4.28

Apr-88	Sep-88	153	10	4	0	9	138	20	837	577	346	285	317	1520	357	308	7.96
May-88	Oct-88	153	10	4	0	8					185	145	460	1705	42	22	0.64
Jun-88	Oct-88	122	10	4	2	9					713	521	513	2418	659	516	13.33
Jul-88	Nov-88	123	9	4	2	8					330	228	682	2748	161	103	2.99
Aug-88	Jan-89	153	9	4	2	8		28	1002	713	787	566	772	3535	697	564	16.40
Sep-88	Jan-89	122	8	5	0	8	166	27	1003	714	739	475	809	4274	702	423	12.30
Oct-88	Feb-89	123	9	5	0	8					536	380	741	4810	604	408	11.86
Nov-88	Feb-89	92	9	5	0	9	208	36	1097	775	648	436	658	5458	734	478	12.35
Dec-88	Mar-89	90	9	5	0	9					529	369	601	5987	584	388	10.03
Jan-89	May-89	120	9	5	0	9					558	416	639	6545	520	325	8.40
Feb-89	Jul-89	150	9	5	0	8	215	37	1139	799	558	408	689	7103	507	348	10.12
Mar-89	Jul-89	122	10	4	0	8					478	342	584	7581	583	420	12.21
Apr-89	Jul-89	91	13	4	0	9					678	490	583	8259	680	452	11.68
May-89	Aug-89	92	13	6	2	10	238	47	1197	817	730	592	610	8989	703	441	10.26
Jun-89	Sep-89	92	13	6	4	12					518	415	186	9507	941	540	10.47
Jul-89	Oct-89	92	14	5	7	12					404	306	330	9911	260	141	2.73
Aug-89	Nov-89	92	14	5	3	13	247	50	1228	844	724	558	272	10635	782	540	9.66
Sep-89	Nov-89	61	14	5	1	13					787	631	99	11422	964	542	9.70
Oct-89	Nov-89	31	14	5	1	13					759	600	70	12181	785	410	7.33
Nov-89	Nov-89	0	14	5	1	13	249	52	1260	853	330	268	0	12511	399	133	2.38

	SDE	PA	Int.	FTE
Man Months:	506	115	34	455
Man Years:	42.2	9.6	2.8	38
% post c.c.:	31%	57%	56%	30%

Pre c.c.:	4810	3546
% of total:	38%	38%
Post c.c.:	7701	5831
% of total:	62%	62%
Total:	12511	9377
% of bugs fixable:		75%

	total	post c.c.
Bugs fixed:	8180	5158
Postponed:	1197	673
% pstpnd:	13%	12%



MS-PCA 1295278

CONFIDENTIAL