To: Mary Snapp, Jon DeVaan, Bob Muglia, Rich Tong, Dennis Tevlin, Grant George, Jon Reingold, Jim Allchin

From: Steven Sinofsky

Date: March 28, 1998

Re: Looming Crisis: Windows Software Quality

Introduction

Microsoft and the software industry are positioned to experience a crisis in public confidence of the highest order. I realize that this sounds dramatic and that I am hardly the first to make such a claim, which tends to make the rounds every few years. This memo looks at the current situation in our industry regarding the quality of software, both actual and perceived, and how a number of key factors are falling into place that make the time right for a major crisis to unfold.

We are in a difficult situation in that there are no magic answers. The reality of software today is that we barely understand how to build a marketable product, let alone build one that is free from *defects*, perceived or actual. Our ability to build systems far more complex than we really understand is very real, and somewhat troublesome. In this sense, this memo is written to enlist the help of those that can help to understand and hopefully direct the public perception of this problem, for we must humbly admit that our engineering discipline is far behind the desires of our customers and critics. We should also begin to understand how we might expand the corporate image to include quality, in addition to innovation and cutting edge technology.

As a note, this memo will use a number of words that have specific legal meaning such as defect, monopoly, liability, etc. There is no intent for these words to follow their legal definition, but rather the everyday definition is assumed.

The Alignment of Negative Factors

As we know, one of the key elements that can turn a single instance of a problem into a full blown crisis is the ability of the collective of industry press, pundits, and watchdogs to string together a series of seemingly unrelated events and find a common thread. With this common thread, even an average reporter or writer can formulate a thesis and necessary proof by citing these events. Recently, we have seen several bugs in Office 97 and there is genuine concern this can lead to the very costly *Office 97 is buggy* public perception. With the addition of problems from Macintosh Office 98 and perhaps the problems with the FrontPage Server Extensions, it is not hard to see how this can lead to a story on all of Microsoft software and the inherent *bugginess* of our products. We have seen the damage first hand with Word 6 for the Macintosh, a product that never really recovered from the initial negative perception and realities.

I believe this problem is far deeper than Office and has the likelihood to mushroom into a sweeping indictment of modern PC software. This indictment will be amply supported by real evidence from our products as well as fanned (perhaps for self-interested reasons) by our critics in our industry, government, and consumer advocates. The downside of this are risks to our business and asset of the Microsoft brand, and also opens up the opportunity for competition to leverage the failings of our products to provide alternatives. Although the later is a remote possibility due to the momentum of our products, it is not going to be hard to see how this can be used to great affect by our competitors even if it does not lead to profit.

At this moment in time there is alignment in four key areas and this alignment will provide the framework upon which anyone in a position of power, need, or desire could build a very strong case regarding the lack of quality of PC/Windows software. These areas include perception, circumstance, regulation, and evidence. By illustrating these I hope to show just how easily we are positioned to be the target of a great deal of negative energy. While not factors of causation, these are all symptoms of the looming crisis of software quality. It should be emphasized that the crisis is looming, and I feel it can be minimized and perhaps prevented.

Current Perception

First and foremost, the groundwork for a crisis in software quality has already been put in place. It is hard to imagine any customer of Microsoft's or any owner of a PC that does not think of their PC as flaky or buggy and places the blame on Microsoft. Part of this is definitely a perception issue tied to Microsoft, considering that Macintoshes are equally flaky by any measure yet are perceived to be less so (perhaps it is the smiley face when you system crashes and you lose all your work).

There are two aspects to the perception that software just does not work, or fails to meet expectations. First, there is the perception that software crashes and the industry collectively says, "well sometimes it just does that." This might remind one of an old Internet musing, *If Microsoft Built Cars*, that contains some gems such as:

- Occasionally your car would just die on the motorway for no reason, and you'd have to restart it. For some strange reason, you'd just accept this, restart and drive on.
- Occasionally, executing a maneuver would cause your car to stop and fail to restart and you'd have to re-install the engine. For some strange reason, you'd just accept this too.
- Macintosh would make a car that was powered by the sun, was twice as reliable, five times as fast, twice as easy to drive - but it would only run on five percent of the roads.

A second problem with the current way computers are perceived (and are in reality) is that they are just too hard to use. Of course you will get no disagreement from me about how hard computers are to use, but relative to what? One thing that has caught our design and engineering abilities off guard is the incredibly large and diverse customer base for our products. This base is ever increasingly unable to and uninterested in becoming *computer users*, but rather is more interested in earning their paycheck and moving on to important things in their lives. A view of this reality also must take into account the ever more complex software that Microsoft is capable of building, both because of the increasing sophistication and experience of our developers, our desire to solve more complex problems, and because of the sheer number of people we have on each product.

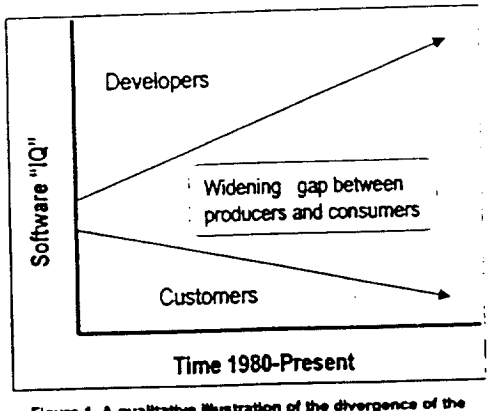

Figure 1. A qualitative illustration of the divergence of the developer and customer.

To our customers, it appears as though we take pride in this complexity and offer to show it off at every step. It is if we are trying to prove that we know a lot about computers and are intent upon showing off. Not unlike the telephone engineers who thought it was cool back in the 1970's to build a clear plastic phone to show off all the wonders of modern age telephony. On the other hand, it is natural to assume that in a specialty-based economy producers always know more than consumers (most of us know little about cars, farming, medicine, or television production). It seems we are at a unique point where there are just enough people who know just enough about computer software to think that it should be trivial to both produce and consume, regardless of the amount of functionality.

Another well-known Internet musing that comes to mind is *What If People Bought Cars Like They Buy Computers*, with the thesis that GM does not have a help line for people that do not know how to drive. Although this one tries to point out that sometimes expectations are bit high, there is some truth to the emotions this brings forth:

HelpLine: General Motors HelpLine, how can I help you?

Customer: I got in my car and closed the door and nothing happened!

HelpLine: Did you put the key in the ignition slot and turn it?

Customer: What's an ignition?

HelpLine: It's a starter motor that draws current from your battery and turns over the engine.

Customer: Ignition? Motor? Battery? Engine? How come I have to know all these technical terms just to use my car?

Another important aspect of the perception of software quality is one that is specific to Microsoft—the amount of profits the company makes. Many times when we talk with customers about bugs they perceive in our products, it goes without saying that they feel we should leap tall buildings to fix the problem and help them to deploy it. The negative perception that people get from being put on hold when they call technical support is often accompanied by the feeling of "they have so much money, why don't they hire more people." This is also an area where we are at a deficit relative to IBM in that IBM seems always to deploy a large number of people at a customer's location in order to help with a problem. Even if the problem is not fixed, the perception is very positive when a team of people fly in to help out. This all boils down to the view that Microsoft has so much money there must be a way to avoid problems. One wishes we could just explain to a individual customer that they paid $100 for a package which might only entitle them to a certain level of expectations, especially when I consider the poor service I feel I get with my car that costs 250 times as much. Even corporate customers who spend a large amount with us at one time are still spending only $200-300 per desktop.

Little more can be said about the current negative perceptions of Windows software. Jim Allchin in his well-circulated memo spoke of the complexity issue in terms of how we present our software functionality to customers. I fear this is just the tip of the iceberg in terms of the negative perceptions of our products.

Today's Circumstances

The environment surrounding Microsoft at this time is at an all time low, as we are all unfortunately aware. Two factors that contribute to a negative perception of software quality are the perception of Microsoft's **monopoly** position and the perceived **arrogance** of Microsoft. Our entire industry is facing a **labor shortage** that is starting to receive widespread coverage. Finally, there is clear reason to believe that PC/Windows software has reached a level of importance in society that we should expect **consumer watchdog organizations** to zero in on any problems, real or perceived.

The wrong-headed perception that Microsoft exerts monopoly leverage in the software industry is the first step in creating the software quality crisis. Once consumers feel that a company has a lock on their dollars, the expectations of quality go way up and the sensitivities to poor service and quality are heightened. Microsoft has entered into the realm of suppliers that owe more to society than just providing an acceptable level of products and services; we must exceed customer expectations. Steve Wildstrom's recent Business Week article clearly set this tone and there is no reason for us to think this will not catch on and become mainstream zeitgeist.

Once there is an assumption by a customer that he or she is locked into a company, the risk of appearing arrogant dramatically increases. The old *Laugh-In* line, *we're the phone company, we don't have to care*, comes to mind. When customers feel that their supplier is out of touch or cannot be bothered to address shortcoming in their products, then it is the first step down the road of breaking the implicit trust between customer and supplier. The next step is a general feeling that the supplier does not need to bother to care because the customer has no choice. It goes without saying that the general perception among many influentials, and certainly among our dissatisfied customers, is that Microsoft is arrogant.

Recently a number of stories have appeared in the popular press touting the labor shortage in the software industry. One need not look farther than our own problems at hiring enough people to satisfy our needs, or the help wanted section from any newspaper to see the number of openings in the information technology profession. One recent example is that SAFECO has begun to hire people with no software experience, but a demonstrated aptitude based on a written exam and interviews, and then runs them through a three-six month course to enable them to become members of their technical staff. It does not take much of a leap to assume some companies are employing people to build software that might not be qualified. Once the conventional wisdom says that unqualified people are writing software, it will become very difficult to claim that a product is free of unqualified contributors.

Consumer watchdog organizations, notably Ralph Nader's but also consumer reporters on TV and in print, are likely to start taking an interest in the quality of software. Steve Wildstrom has definitely begun to view himself as a consumer guardian, and most of the local technology columns in newspapers tend to be

focused on questions and answers about broken machines. To date, we have been fortunate in that review-based watchdog organizations have mostly focused on hardware because that more easily fits within their traditional testing methodologies (Consumer Union for example). It seems, however, that any business that reaches several billion dollars and simultaneously reaches into the average American home is likely to attract the attention of these watchdog groups. The current government investigations probably served as a wake-up call to these groups. In a pure economic sense, it seems that this is a growth opportunity for a watchdog or any organization looking to increase their influence and stature is likely to look at software as an easy target.

State of Regulation

We are all aware of the current government scrutiny of Microsoft. It has become clear that this is an opportunity for government officials to use this state of affairs to bolster their own careers (putting aside any genuine concern for society of course). Software lacks two key regulatory elements that often help to slow an ensuing crisis, or at the very least enable a crisis to avoid targeting a specific supplier of products. There are **no government agencies** responsible to the people for the quality of software and there are no **professional societies** to oversee either the discipline of software creation or the certification of the members of the discipline.

It goes without saying no one wants the government involved in the production of software. Unfortunately, once a product becomes critical to society and problems with that product become pervasive and there is evidence or perceptions that the suppliers are not always acting in the interest of consumers, government involvement becomes inevitable. It is hard to believe that calls for government oversight of software products will not increase in the near future, especially from consumer watchdogs and perhaps even our own competitors. The fact that there is no government agency responsible for software or involved in software certification can lead to the general perception that no one is watching out for consumers. Given the size of our industry, the perception of quality of our products, and the political spoils to be had, it seems obvious that there will be a ground swell of efforts to create such a body. This seems to be an especially likely outcome of government investigations into our industry since it will enable government officials to appear pro-consumer without looking like they are targeting a specific supplier. In terms of creating a crisis in software quality, it seems that pointing out that planes are complicated so there is the FAA or NHTSB crash tests many cars to ensure that manufacturers are not cutting corners makes for reasonable evidence that software requires government intervention. How far are we from *the NSEB (National Software Effectiveness Board) reports that Excel has recalculation errors and is investigating the ramifications*?

The other missing element of regulation is our own industry's lack of self-regulation. This takes on two forms. First, we do not have the traditional organizations that other engineering disciplines have. In fact, our own industry publications (ACM, IEEE) are regularly filled with articles pointing out this deficiency. For example, the Society of Automotive Engineers, the American Society of Civil Engineers, American Society of Mechanical Engineers, or many other affiliations that view themselves as the source of best practices and speak out when the discipline's integrity is challenged. It is not unusual to see ASCE representatives on TV after a bridge collapses, or to have SAE members testify in automobile related trials. Software Engineering has two organizations, neither of which has any critical mass relative to the number of practitioners and arguably little credibility and certainly even less in industrial settings. The Association of Computing Machinery and the IEEE Computer Society essentially represent the academic arm of our field and predominantly serve the Ph.D. research community. In fact, most practicing *programmers* tend to dismiss out of hand these organizations when it comes to understanding the issues faced in developing commercial products such as Office and Windows. Nevertheless, when it comes time to discuss the state of our industry we are left with corporate representatives who cannot avoid appearing self-serving. Our industry does not even have any form of self-policing such as Underwriters Labs or even the Good Housekeeping Seal of Approval, let alone an organization as invasive as the FAA, FDA, NHTSB, or OSHA. The *Year 2000* issues certainly bring this to light, as the expert sources for the numerous magazine articles have almost always been representatives from consulting firms directly profiting from this foreseeable event.

A second form of discipline related regulation that is missing is some form of professional certification. Pilots, surveyors, civil engineers, doctors, accountants, bus drivers, and just about every other discipline (especially lawyers) have some form of entrance examination and/or recognized apprenticeship, either often government regulated or mandated, required before one can practice their profession. Many might view these as ways to increase the barrier to entry and thus somewhat self-serving, but nevertheless these serve as checks and balances in light of issues of quality and consistency of products and services. If you are unsure of your doctor's qualification you can call the AMA (or even check their web site) and even within sub-specialties there are organizations that make qualifications available and provide for some form of measured admittance (for example American Academy of Ophthalmology). Our industry lacks any such

entrance exam or self-policing activity, and even worse clearly sets the public perception that we shun these activities. There is hardly anyone that is not familiar with the notion that great programmers are unkempt, college dropouts who work long hours surviving on pizza and Mountain Dew. Boy, if one is frustrated with a computer crashing it sure seems obvious why, especially when I compare that with the image of a fine doctor.

As an aside, the efforts over the past 5-7 years or so to elevate ISO-9000 certification to packaged software show customers' desire to have some form of regulation or certification. Once again our own efforts in this area might show that we have taken less than a pro-consumer view of these attempts. We have not worked nearly as much to gain certification as to avoid being certified until it absolutely mattered. Most of us would agree that the ISO standards for software completely miss the boat and offer little to prevent the creation of totally non-functional software, but the lack of the standard creates a vacuum to fill.

Evidence At Hand

There is already ample evidence in the popular and trade press regarding the poor level of software quality. In fact there are several web sites and publications that are entirely devoted to uncovering, documenting, and publicizing how software misbehaves. These sources often serve as the starting point for stories regarding software quality. We saw the use of these sources quite a bit during the recent, and not yet over, events surrounding software security and the Internet.

In general, there is a wealth of information available to anyone wishing to show that software is defective. We've even seen the mainstream business press use anecdotal data to support claims and opinions regarding software quality—The New York Times, Business Week, and Wall Street Journal all regularly run stories based on a single reader (or often writer's) experience.

Three areas come to mind as providing evidence to one wishing to demonstrate the poor quality of software: Year 2000 issues, software rushed to market, and sources for known bugs.

The **Year 2000** phenomenon provides the clearest entry (perhaps lead) into a story about software quality. The examples are easy for people to understand (wrong bank balance, credit card denial, missed plane trip, etc.) and the explanation is also easy to articulate. There is also a reality that Microsoft's efforts in this area have been late and our competitors have mostly done a better job at being proactive and making customer commitments.

Our industry is well known for two things regarding releasing software: **missed release date commitments** and **software rushed to market.** It does not take much to assume that if a product is late in coming to market, then the supplier might cut corners to get it *out the door.* It also does not take much to conclude that if something is rushed to market it runs the risk of not having adequate testing. Unfortunately nearly all of our industry's products suffer from these perceptions. Most of our customers already have hard and fast rules that insist on waiting for the first maintenance release or service pack before deploying a major package. Windows 95 was truly unique in this regard, as there was no need to rush out a service pack. Nevertheless, many early stories on Windows 95 quality were laced with expressions such as *plug and pray*. This concern, however, did not stop the average corporate customer from proceeding with caution and waiting quite some time to begin serious evaluation and deployment.

The recent advent of so-called ***Internet time*** has also contributed, in reality, to a decline in software quality. Companies are more willing to make early test releases available to the general public for download—a process that used to require an NDA and signing up for the exclusive club of beta testers. It is interesting to note, for example, that Netscape Navigator required 4 incremental releases within a short period of time for both Communicator 4 and Navigator 3. Many of our corporate customers have explicit policies prohibiting the use of downloaded software or beta software, yet find themselves unable to prevent the *mayhem* they see taking place on the desktop. If people can try the software at home, then they want it at work as well, which is exactly what the marketing people are hoping for. Of course this is closely tied to rushing products out the door, but it is unique in that the business environment and customers are demanding companies release software in this manner before it is ready as it has become a key marketing and communication tool.

There are also numerous sources for lists of known bugs regarding software. In many ways the mere existence of these lists, whether from the supplier or a third party, merely admit to the fact that software is riddled with defects. The product README, the knowledge base, PSS, internal customer bug lists, watchdog web sites, newsletters, etc. all provide ample evidence that software has lots of problems. To compound matters all of these sources are readily available to consumers, regardless of their ability to rationally act on the information. As consumers we probably all believe in full disclosure—we've probably all read the hundreds of possible side effects of something like a common allergy pill—yet we must consider

that there is a huge stream of information continuously pointing out that software products are defective and/or behave in unintended ways.

Smoking Guns

Given these negative factors that are perhaps aligning to make a case for universally poor software quality, the next step one would take to support such a case would be to gather evidence. The use of customer anecdotes, personal experience, or combing newsgroups is certainly one way, but nothing beats going straight to the engineers that created the product to find out what is wrong with it. Unfortunately perhaps software developers overachieve at **documenting the flaws of software products**.

We all marveled at the internal memos from tobacco companies warning of the danger of smoking or from car companies warning of injurious crash scenarios, yet our own industry is filled with an equal set of *smoking guns* when it comes to software quality. That is that the unexpected or undesirable behavior of software products is readily documented throughout the very process of creating, selling, and supporting it. Some of this information is widely available and some is part of the development and support process. Regardless, if one wishes to make a case for low quality or defective software there is a volume of supporting material that is quite scary. The following just lists some examples of what someone might call a **smoking gun aimed at poor quality software:**

Knowledge Base and ReadMe: These two sources of information are the front line customers see of our products. The ReadMe is often a document listing known compatibility and usage problems, along with pointers to the additional sources of information online. Microsoft calls this source the Knowledge Base, but every software company has a list of known issues usually described in short articles that are easy to search and collect. As an example, searching the Microsoft Knowledge Base for the word *bug* in articles related to Microsoft Office turns up only about 40 articles. If you happen to know that we use the phrase *confirmed problem* you will be able to scroll through hundreds of articles on Office. Most of these will never cause one to lose data or cause the application to crash, but they all document unexpected behavior.

Angry customers and press guardians: These constituencies are great sources for information on problems with software. Many of them are in very influential places and it is easy for them to have an impact on a product far beyond what one might consider reasonable. If Walt Mossberg says a product is *unstable* or contains *bugs*, it is a sure bet that a product is on a path to develop that perception.

Books such as *Annoyances* or *Idiots*: One particular guardian is Woody Leonard who publishes Woody's Office Watch which is a compilation of tips, tricks, and of course bugs regarding Microsoft Office. He has also co-authored a series of books for O'Reily Word 97 Annoyances and Excel 97 Annoyances. These books have a wealth of how-to information, but also contain a plethora of descriptions of what Woody thinks are misbehaving products. There are examples of such books for many software products, so this is not limited to Office. But in terms of a source of information on how products are quirky or buggy, these are a ready-made pot of gold (in fact we even bought copies for all of our testers).

Beta Tests: Many reporters are now actively involved in the beta tests of our products. The widespread use of insecure downloadable releases has made it easy for people to obtain the software. Our own desire to obtain more outside testing of our products before we ship has relaxed the standards we use for accepting someone into the beta. Once someone is on the beta, they have access to the private beta newsgroups. These discussions are often a good source of information on the problems that a product has and more importantly allow people to calibrate the quality level of a product over time. It is easy to look at the bugs reported during a beta test to see if they were fixed, and in fact many of the most raging debates on these beta forums have to do with updates of the software containing previously reported bugs.

Internal Bug Databases: The true smoking gun possessed by any software company is the internal bug database. Some might remember several years back the leak to the trade press of the number of bugs that were active in the Word 6 database prior to shipping (thousands). This certainly contributed to the industry perception that Word is buggy, and there were 4 subsequent maintenance releases as well which did not help. Our current bug databases for Windows and Office contain enormous numbers of issues, tens of thousands. If one were to look at the details of many of these reports there is likely to be a raging debate between members of the team over the severity of the issue, the need to fix or postpone fixing the issue, and the ramifications of not fixing the issue. Any product development in any field will have debates such as those, but my guess is that few of them are so easily searched, categorized, and reportable and also contain such candid and easily misinterpreted language. Needless to say our industry will suffer greatly if such databases of software products make it out to the general public.

Software Quality and Our Competitors

In terms of the strength of the Windows platform and Windows applications, it is worth thinking about how our competitors have been or could be using the software quality issue against us or in their favor. The biggest issue is that Microsoft is the predominant supplier of off-the-shelf software that runs on desktop PCs, whereas our competitors provide either more industry-specific software or server software. The implication is that it is quite easy for our primary competitors to distance themselves from the low quality PC experience, and they can do so with some credibility because in a sense most people in the opinion-setting world do not experience our competitors' software directly.

In particular, Sun and Oracle can talk about how their software is used to run businesses and perform mission critical functions. They know how to keep things running and can say so with incredible examples and bravado. Of course they do not talk about the large staff of engineers and the high cost of running such software, which would be prohibitive for the average customer of a word processor. In fact it is this software where Y2K issues will be most acute. Because these software systems are viewed through the lens of a terminal or even more often through a phone talking to someone in front of terminal, the average person is somehow insulated from the failure of these systems. The UAL flight reservation software is easily as complex as Word in the number of commands, yet the average Word customer receives no formal training compared to many weeks of training for a reservation assistant. Nevertheless, it is easy to see how Sun and Oracle could use the issue of poor Windows/PC quality to their advantage. They have been doing so to some degree, fortunately not as it directly relates to quality, but more to forward their own agendas of server-based computing.

IBM is an especially blessed competitor on this front. Outside the software industry the IBM name has traditionally stood for service and computers that run the world (I haven't seen this year's AAU, but IBM usually does pretty well). Because of the large number of service employees and the direct benefits to IBM of performing service, there is almost a motivation to show that software is complex and requires service. IBM's response to Year 2000 shows how sophisticated they can treat the software quality issue—the IBM Y2K web site has been up and running for quite some time and their commitment is strong (Microsoft's commitment will be very similar, but our site is not yet live). In addition, IBM brings with it the benefits of running *mission critical software* that Sun and Oracle can claim.

Netscape is probably in a position similar to Microsoft's and the rest of the Windows application software industry (Lotus, Borland, Adobe, Corel, etc.) in that it is through the use of their applications that most consumers will experience the negatives of software quality.

What Can We Be Doing?

This memo has tried to show that at this time it appears the environment is right for the build-up of a public crisis regarding Windows software quality. This does not mean such a crisis is going to happen, and chances are we will avert this one as well. We need to be realistic, however, that our software is complex and it often behaves in ways that are unexpected and is even broken. There are some opportunities for Microsoft to provide leadership, while at the same time helping to level-set consumer's expectations as we continue to innovate our products, given that there are not any fundamental changes in our engineering practices on the horizon. Of course some of these ideas are a little nutty.

Start talking now: First and foremost, we should come up with a communication plan and a set of engineering commitments that discuss software quality and how Microsoft views this issue. Perhaps we can pioneer some sort of assurance program, money-back-guarantee, or even a statement that goes beyond our warranty that removes any liability. We might consider having a standard slide that we can use at many high level briefings and executive speeches that discusses software quality and our commitment to meeting the needs of customers.

Define the concept of software bug: One stumbling block we have when talking with customers is defining what it means for software to have a bug. Our products do so many different things, and often do things that people did not know were possible, and thus the expected behavior is often ill defined at best. When we talk to the press we say that a "bug is just when the product behaves differently than someone expects." Unfortunately this does not help us to serve customers very well, because implicit in this is the customer's desire to see the bug fixed. Perhaps as part of our communication plans we should define parameters that mean something is a bug, missing functionality, strange functionality, or just a wishlist item. It might even make sense to define a set of criteria similar to our own RAID tracking criteria in order to help customers understand where on a quality spectrum a report might fall.

Microsoft Secrets: This is a book that has fallen a little out of date but does a very thorough job of describing our development process and how we go about building software. It does not directly discuss the quality issue, but it does serve to show a level of professionalism associated with Microsoft. We onginally chose to cooperate with the authors with an explicit goal of using the book as a tool to help customers to understand our process. Perhaps we should work on an updated work or at least revisit the use of this more directly.

Document and quantify our testing process: Our Directors of Testing and Development have in past years made great progress in helping customers to understand the processes we use to engineer, test, and release our software. I believe for the customers that know to ask for this information and get a chance to talk to the right people this has been a positive experience. It might be worth our time to investigate a more proactive discussion of these methods. From personal experience, I have done a standard presentation on our development process for dozens of customers and audiences, and I think they find it very worthwhile and it leads to a higher degree of empathy. We might consider making a development process discussion and presentation part of the standard EBC schedule, or at least make such a talk available to any customers that are interested. We could also choose to write our own version of a *Microsoft Secrets* that describes the quality and testing processes that go into our software.

Beta Tests: We should revisit our beta test process and look at it more closely in terms of how the quality of the product is impacted. Most test managers will tell you that a very small percentage of the bugs that are fixed in the product are found by outside testers. Yet we continue to release massive amounts of software that does not yet work very well. On the other hand, when used effectively beta processes can be a huge builder of customer satisfaction and buy in. For example, most of the early Windows 95 beta testers took great pride in saying "she's ready to ship" on the beta forums, and it is not likely any of those people will call the product buggy since they felt so involved in the process of releasing the product. On the other hand, we had very poor luck in getting people to install betas of Office 97 and in the end the product suffered from a lack of this buy-in from customers. A key part of communicating our view of quality will be the terminology and associated expectations of pre-release software.

Bug Lists: If ever there was a double-edged sword then it is bug lists. Many customers ask for detailed bug lists for our products that include all known issues. We attempt to do this in the knowledge base to some degree, but things are not in a list form and you have to know the problem for which you are searching. A key aspect of this issue is defining what it means for something to be on a *bug list* and what it means in terms of our response to the issue. For example, do we commit to fixing everything on the list or is the list merely documentation? It goes without saying that a list of several thousand problems with a product is quite a negative and hard to see how to turn it into an asset.

Help form a professional society for industrial programmers: Our discipline lacks a professional society, and as a leader in the field perhaps Microsoft should help put in place a society that can over time evolve standards for programmers or at least be a source of best practices. We have been a key member of the BSA, so it is only natural that we might help more with the engineering elements of our products as well. The Microsoft certification program might be a good starting place in terms of programs and policies, but such a group needs to be more removed from the sales process and less focused on a single product. One aspect of this we could be more proactive about is working with universities on the skills required to be an industrial programmer. Today universities, especially those we recruit from heavily, are much more focused on sending students to graduate school. The ACM curricula recommendations are not well-tuned to what most programmers will be hired to do.

Help drive the creation of the equivalent of UL-Listed for Windows applications: We have gone to great lengths to create a Designed for Windows 95 logo program. This logo, however, is focused on the exploitation of key features of Windows 95 and does not imply any standard of quality or functionality beyond leveraging key technologies. It might be worth investigating the creation of some sort of seal of approval. There are numerous testing labs as well as consumer advocacy groups that might be good organizations to help devise such a system. Obviously this has the potential to be an arbitrarily complex process (as our own logo has proven) but there is a certain base level of quality we might be able to bring the industry up to. We are all familiar with downloading a piece of software that plain does not work, or crashes very easily, so perhaps this is something we can help to avoid.

Microsoft fix-it squad: Perhaps we might think of having a new arm of MTS, or reassigning some portion of MTS, to deal directly with customers in major metropolitan areas. We might choose to have a staff of people in each city that we could dispatch on location to certain classes of customers (retail only, or small business) to help them with problems when the phone-based support fails to solve the problem. Obviously this is a bottomless pit and might backfire, but the idea is to think of something big to do that will cause customers to pause and say we're serious. We have begun offering email-based support but anecdotally I think this is

probably not working as well—the turnaround time is slow, and the ability to diagnose and resolve the problem is pretty limited.

Improved service: This is a challenging area because we have long held out adding something as simple as a toll-free number when we had very competitive reasons to do so. Yet we improved our service reputation in measurable ways despite this *obvious omission*. We must, however, find a way to be viewed as a company that stands behind its products. We are running the risk of being too focused on our corporate customers, when it is the individual customer, or reporter, that is often the most helpless. We should look hard at some more novel approaches to support that show we are making support better, though I admit to having few suggestions.

Make sure the subject is about all software, not just Windows: All software suffers from bugs, we're sorry to say. It is probably important to consider how we might have any conversations about software quality be inclusive of all of the industry and profession and not focus entirely on Windows and Windows applications. I would confess to having no idea how to do this, since it seems that most any discussion of software quality issues will focus on the software that most people see, which is Microsoft's. In any event, perhaps part of the goals of our communication strategy would be to enlist other companies outside our traditional ISV community. We might even expand this to include web site interactivity or VARs building turnkey systems, both of which also demonstrate very uneven quality.

I wish I had more suggestions or suggestions that I thought would help to directly fend off the issues at hand. More importantly I hope we can make progress at building better software sooner, though I am confident that we remain among the world's best. Microsoft would benefit enormously if we could be mentioned in the same breath as quality, much the same way people look to us as a source of innovation and success.

Selected Bibliography

As stated at the start of this memo, this is hardly the first time that someone has jumped up and down claiming that there is a crisis in the software industry. Hopefully, reality will set in and we will avert such a crisis. The following are books that some might find interesting as they relate to the problem at hand. The hyperlinks are to Amazon.com.

Interface Culture : How New Technology Transforms the Way We Create and Communicate by Steven Johnson. This book describes how day-to-day experiences are altered by the technology that defines those experiences. In other words, one could read this as how we are victims of technology.

Why Things Bite Back : Technology and the Revenge of Unintended Consequences by Edward Tenner. Tenner describes the basic laws of unintended consequences and illustrates these with current technology examples.

Digital Woes : Why We Should Not Depend on Software by Lauren Ruth Wiener. This is a relatively early book on the flaws of software and the state of unreliability of modern software.

Fatal Defect : Chasing Killer Computer Bugs by Ivars Peterson. This book takes a look at some of the more well-known bugs and takes the reader through the detective trail on how the bug was found and remedied.

Microsoft Secrets : How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People by Michael A. Cusumano, Richard W. Selby. This book might help us to understand how the industry perceives our development and engineering process, when it is viewed in a good light.

