SOFTWARE CUSTOMER DISSATISFACTION

CEM KANER & DAVID L. PELS

MAY 27, 1997

Introductory Note: *Legislative drafting involves a complex balancing of risks and interests. When we wrote this paper, an effort was underway to rewrite the Uniform Commercial Code (United States) to govern all software related contracts. That revision was rejected by the American Law Institute, one of the two organizations that maintains the UCC ("maintains" means that ALI authors or approves all changes to the UCC before sending them to the legislature. The other maintaining organization is the National Conference of Commissioners on Uniform State Laws, NCCUSL.)*

After ALI rejected the UCC revision, NCCUSL renamed the proposed law as the Uniform Computer Information Transactions Act, which has now passed in Maryland and Virginia, but has (as of March, 2001) been blocked in every other state that has considered the bill.

We wrote this paper to provide background data for the drafting committee. Kaner has written critically about the proposed revisions previously. [1] We skip those discussions here and focus on the data that we've come up with while researching a book we're writing (called "Bad Software"). These data have value for software quality planning independent of any legislative work.

Much of what we have to say is controversial, and much of this data is hard to get, so we footnote our statements extensively. We hope the notes are more useful than distracting.

(NOTE: In 1999, Kaner was elected to the American Law Institute in recognition of his work on computer law, such as this paper.)

ABSTRACT

There is significant customer dissatisfaction with software and the software industry. Article 2B fundamentally favors software publishers (including the unscrupulous ones) over small customers. In this paper, we present some dissatisfaction data, and we note practices and attitudes that cause publishers to underestimate the problems. We don't discuss 2B directly in this paper, but our concern with 2B triggered the extensive effort that went into gathering the material that we discuss here. Our concern here is straightforward: We believe that a law that protects an industry from its legitimately unhappy customers is a time bomb that will explode into a backlash of excessive regulation and loss of market to foreign competition.

In the year since the last NCCUSL meeting, many proposed revisions to Article 2B have been considered and a few changes have been made, but the underlying philosophy and bias of the current draft is the same as last year's.

Here are our qualifications for writing to you: Kaner has worked with software for about 21 years, and has had several software development positions in Silicon Valley since 1983. He is the senior author of the best selling[2] book on software testing and a frequent speaker at conferences on software quality and customer satisfaction. He is also an attorney, and has attended the 2B meetings since February 1996. Pels has managed technical support for software or computer-related hardware at several Silicon Valley companies since 1986. We have been researching and writing a book on software customer satisfaction

since 1994.

Customer Dissatisfaction

For the first time, computers outsold television sets in 1994. By 1996, 35% of American households had computers.[3]

By the end of 1995, computers and software ranked #8 in the Top 10 list for complaints to the Better Business Bureau, outdoing such historically reputable businesses as used car dealers.[4]

There are a *lot* of customer complaints and the cost of handling complaints and requests for help is skyrocketing. Over the past seven years, the ratio of support to total employees in hardware and software companies has grown from 1 in 12 to 1 in 6.[5]

In 1996, there were 200 million calls for technical support.[6] At an average of about $23 per call,[7] the industry spent about $4.6 billion on these calls.[8]

- The industry left these callers on hold for about 3 billion minutes.[9]
- The software industry has been one of the worst for leaving callers on hold. A small study by Service Management International indicated that software companies leave callers on hold longer than any other industry studied, worse than government agencies, computer hardware companies, airlines, banks, utility companies, and others.[10]
- Along with long hold times, it often takes additional waiting to get to someone who can actually answer your question. According to a members-only database provided to Software Support Professionals Association members, the average response time to reach the right support technician is 30 minutes for PC/Shrink-wrap products.[11]
- Call hold times have declined recently, at least for companies that offer toll-free support lines. Last year these companies paid $500 million for toll charges while customers sat on hold.[12] They do that less now. Instead, they give you a busy signal when they're too busy to take your call. According to Ron Schreiber, the Chairman of Softbank Services, the world's largest software support outsourcing agency, **at peak times 85% of calls into tech support now get busy signals.**[13]
- Another problem faced by customers is extended finger-pointing in multi-vendor situations. If you buy a word processor made by one company, a printer made by a second, and an operating system made by a third, guess what happens if your word processor won't work with your printer. The word processor company blames the printer. The printer company blames the operating system. The operating system vendor blames the manufacturer of your video card, who blames your word processor. You have a merry time calling each company several times, until you either give up (a not uncommon result) or you finally get an answer.

  Some companies have help desks. For example, a hospital might have a help desk, an in-house technical support group that supports the hospital staff in their efforts to use the word processor, printer, etc. These groups operate as super-sophisticated user representatives, and keep pushing at the manufacturers until they solve the problem. On average, it takes them 3 to 18 times as long to resolve a multi-vendor problem as a problem that can be pinned to a specific piece of software or hardware.[14]

  Consumers and small business customers don't have the luxury of a help desk to serve as their professional bloodhound. Often (we have no estimate, but have heard the story often enough) these customers' problems are never resolved. They throw one or more products away and start over.
- One reason that multi-vendor support is badly done is that the manufacturers' technical staff are often untrained and unfamiliar with the extent of issues in the products they are supporting.

According to the Software Publishers Association, the average support technician gets only 61.6 (mean; median is 40) hours of training before independently handling customer calls.[15]

- Another reason that multi-vendor support is so difficult is that software publishers don't collect the right data. We've seen no published statistics on this, but based on our direct experience with several companies and on discussions with peers in other companies, we can say with confidence that most software publishers test the compatibility of their products with a limited group of hardware and software, ignoring many products that they know customers will use in conjunction with their software. As a result, the publisher has no data on compatibility with these products, and the technical support technician has to work with the customer without supporting data.
- Customer satisfaction with software companies' technical support has dropped steadily for ten years. According to J.B. Wood of Prognostics Corporation, customer dissatisfaction with support has bottomed out in this eleventh year.[16] Schreiber's talk suggested that satisfaction is still dropping.

Software Is Sold With Known Defects

It is impossible to test a program completely.[17] Therefore it is probably impossible to find all the bugs in the program and it is certainly impossible to know whether you have found all the bugs or not. However, it is *definitely not impossible* to *prevent* most coding errors.[18] *Nor is it impossible to find the huge majority of bugs in a program*. Software test managers have told us of products in which all but five bugs reported by customers had actually been found first in the test lab.[19] One large software publisher tracked this performance over a two year period. The head of its software quality control organization told Kaner that customers reported only ten new bugs (not initially found in the test lab, before release of the product) in one product. They reported fewer than ten new bugs in all of the other products.

We (Kaner and Pels) have direct experience with some products. If you ignore complaints about incompatibilities with devices (specific models of printers, modems, etc.) that we intentionally chose not to test and that our companies would not fix, very few new bugs were reported in several of these products. The vast majority of problems that customers called about were known problems, problems that we knew about when we shipped the product.

Quality control managers and consultants sometimes measure the effectiveness of their software testers in percentage terms. Total all the bugs that were either found pre-sale (usually, by in-house testers) and all the bugs reported post-sale by customers. What percentage were found by the testers? In the mass-market software world, percentages like 95% seem to be common and percentages like 99.5% might not be out of line.[20]

This does ***not*** mean that software publishers are aware of every bug when they ship a product. It is dismayingly common to be surprised by one or two never-discovered serious bugs in the field. Note, though, that once you start getting reports of these bugs, you know about them. Some companies will continue to manufacture and sell product that contains serious defects for longer than others.

So why are there so many bugs in mass-market software?

There are so many bugs in mass-market software because it is normal practice in the software industry to leave some bugs unfixed in the software.[21] For example, it is widely reported that Windows 3.1 shipped with 5000 known bugs.

This sounds like a big number, but we want to stress that we are not Microsoft-bashing. Our considered opinion is that Microsoft's staff are careful in their risk analyses and that they strive to ensure that the bugs they ship will not hurt the customer.

To avoid taking sides in the Great Divide, we'll also note this report of Apple's bugs:

> Apple shipped the first version of Hypercard with about 500 known bugs in it, yet the product was a smashing success. The Hypercard QA team chose the *right* bugs to ship with. . . I was working at Apple when Macintosh System 7.0 shipped and it went out with thousands of bugs.[22]

Apple, Microsoft, and several smaller companies might exercise care and wisdom in their risk analyses, but other companies' practices are much sloppier. As a result, they release products with inappropriate errors, that were known (if not necessarily fully appreciated) at the time of release, or that would have been discovered with minimal care.[23]

The practice of shipping with known bugs interacts with the recent trend toward charging mass-market customers for support. Both software and hardware companies see this (charging for support) as a revenue opportunity. Recent memos from Dataquest noted the benefit of the trend toward shrinking warranties for computers -- they discussed the new opportunities to sell service contracts to home users.[24] Some computer companies charge for support for problems with the software that they ship with their computers, from the first day that you have the machine.[25] Some companies charge customers for calls about actual bugs. (Most software companies will probably refund the price of the call to you if you are calling about something that is, in their opinion, a genuine defect in the program. But we are aware of software and hardware companies that will not. And even if a company will give you a refund, you might have to ask for it, and be persistent in your request.) Customers are upset about having to pay for software support, especially about having to pay to ask for the company's help when they are calling about a defect.[26] The ability to charge customers for support creates an incentive to leave some defects in the program in order to encourage customers to buy support contracts.

The World is Different in Different Segments

The testing effectiveness numbers we mentioned are mass-market numbers, collected from managers in quality-conscious companies. (Some other mass-market publishers' numbers might be smaller.)

More intriguing to us is information that Kaner has been receiving that the numbers are different when you deal with software developed on a custom basis, or for in-house use, or for a small number of customers. It looks as though testing effectiveness in such companies might commonly be as low as 50% to 70%.

This doesn't mean that large-system software is less reliable or otherwise of lower quality than mass-market software. For example, large-system vendors often use different planning and development methods that yield fewer problems in the software. They might also fix a higher percentage of the bugs they find. Therefore a product that goes into service might be more stable than a well-tested mass-market product.

Additionally, the service strategy in large-system situations is very, very different. In the mass-market case, you look to find problems that will generate phone calls. At a cost of $3 per telephone-support-minute, no one wants to receive 100,000 calls about a bug in their software. In contrast, in the large-system case, it can be (compared to the cost of extensive testing) cheap enough to fly a few technicians to a few customer sites, fixing the customer problems quickly, that this is more efficient to handle problems in the field rather than finding and fixing them before manufacturing begins.

Within the technical community, these differences in patterns of cost tradeoffs are surprisingly rarely discussed. Development and quality control strategies are often presented and argued as if they are equally applicable to all segments.

We are concerned about any standards-setting, regulation-setting, or legislative effort that treats software development as if it was homogeneous across market segments. The cost/risk tradeoffs are different across segments. Unless the differences in tradeoffs are carefully considered during drafting, the result is likely to be something that works only for the best analyzed situations. With specific reference to Article

2B, we think that from the initial ABA Model License forward, 2B has been dominated by mass market concepts.

We think that the 2B Drafting Committee should seriously consider restricting the scope of 2B to information-related products that are sold or licensed to mass-market customers. The upstream transactions and custom transactions and small-business-critical have different problems,which have not been carefully analyzed in our meetings. Sometimes, restraint is the better part of valor.

Software Publishers Underestimate the Extent of the Problem

Ron Schreiber (chairman of Softbank Services, the world's largest software support outsourcing firm) started a talk to the Association of Support Professionals[27] by asking how many members of the audience thought their departments were giving customers satisfactory technical support. About 90% of the room raised their hands. Schreiber then proceeded to quote several of the statistics we provided at the start of this paper (and others), surprising several members of the audience.

As a whole, the industry seems unaware of the extent of its customer support problems, and of the problems that lead to these huge volumes of tech support calls.

Kaner frequently teaches courses on software quality control and frequently talks with senior quality control staff in several companies. These staff have a strong, ongoing interest in failure data from the field, but they say that they rarely get much data.

Surprisingly few companies have systematic procedures for bringing information about bugs that were found in the field back to Product Development. Individual bugs that are serious might only be reported back as a matter of individual discretion. "Top 10" call lists are often compiled subjectively (and not necessarily accurately) and then reported to some companies' management on a regular or irregular basis. But in the mass market, in most companies, data on call rates per bug are either not kept or not reported to Product Development.

Risk analyses are common in pre-release discussions of known bugs, but they are complicated by a serious lack of data.

- Most companies don't know how much it costs them, per call, to field technical support calls.[28]
- Across industries, fewer than one customer in 20 will complain to senior management about a major problem in the quality of a product, and fewer than one in 50 will complain about smaller problems.[29] Therefore most managers are unaware of the extent of the dissatisfaction with their products.
- Customer dissatisfaction with quality significantly reduces a company's sales,[30] but several (in our experience, most) companies ignore the dissatisfaction-associated revenue risks because they don't know how to estimate their magnitude.[31]
- For an example of an influential analysis that downplays the revenue and dissatisfaction risk due to poor customer support, read Jeffrey Tartar's "Commentary: Is There a Payoff for Service Quality?"[32] His conclusion: "***Despite lots of wishful thinking to the contrary, spending money to upgrade a company's service reputation remains a lousy investment***" because it is (argued to be) unlikely to increase the company's sales.

Poor Data Collection During Development Leads to Lower Quality Products

Systematic failure to collect, use or communicate data is not just a problem on the customer service side. It is typical of software product development, leading to shipment of lower quality products.

In his well respected book, *Patterns of Software Systems Failure and Success*, Capers Jones summarizes his research:

> The number one root cause of cancellations, schedule slippages, and cost overruns is the chronic failure of the software industry to collect accurate historical data from ongoing and completed projects. This failure means that the vast majority of major software systems are begun without anyone having a solid notion of how much time will be required.
>
> Software is perhaps the only technical industry where neither clients, managers, nor technical staff have any accurate quantitative data available to them from similar projects when beginning major construction activities. Lack of measurements means that every major software project tends to be treated as a new and novel mystery, even though similar systems may have been built hundreds of times by hundreds of companies.[33]

When you have no data to estimate the probable time required for a project, you have to guess. In many companies, these guesses err significantly on the short side. The consequence of these underestimates was well put by then-Director of Testing at Microsoft, Roger Sherman, "Bad schedules are responsible for most quality problems."[34]

We agree with Sherman's and Jones' observations. In our experience, overly ambitious schedules are common, and they lead to high risk end-of-project behavior. The software publisher has announced that the product will be ready at a certain time. The customers, the trade press, and the shareholders anxiously await the release of the product, but it is not yet in good shape. The result, as the announced ship date approaches, then arrives, and then recedes into history, is a series of battles inside the company between people who want to start selling the product NOW and people who want to delay shipment even further. This pressure all too often results in the release of a substandard product. Eventually, this product might drown the publisher in complaints and support costs, but many publishers seem to find it difficult to learn lessons about product development from customer complaints.

Concluding Notes on Article 2B

Article 2B relaxes contract law to favor software companies, adopting a material breach standard for defects. The rationale for the material breach standard is that it is *so hard* for the software industry to make bug-free products.

We agree that it is difficult to create bug-free software, but we can't let the extent of the excuse-making that we see in the draft and in the meetings go by without challenge. Most software failures are there because we don't spend the time needed to analyze the product's requirements effectively, to design it carefully, to prototype it thoughtfully, to code it painstakingly, and to test it thoroughly. We set ourselves up for these failures because we keep ourselves clueless about the historical costs of projects. Many defects occur in programs because we don't track our patterns of failure well enough to recognize that we make the same mistakes over and over again. Many of these defects stay in programs because we don't track their costs to our companies or to our customers.

Perfection might be an unreasonable goal, but the software industry can do a much better job of delivering higher quality products at a reasonable cost. Article 2B's liability rules will have an impact on the risk analyses done during product development. That, in turn, will have an impact on the ultimate quality level of products on the market.

Article 2B makes it too easy for companies to ship products with significant, known defects. It provides no meaningful recourse to the customers of these products. We think that this is a grievous error. What value is the preservation of "freedom of contract" (for the drafter of an adhesion contract) if the net effect will be long-term damage to a leading industry? How many times do we have to lose, or see crippled, key American industries before we learn that there is great social value in the protection of customer satisfaction with American manufacturers?

As you appraise 2B's treatment of the law of software quality, we hope you'll ask this question: "Over the long term, will Article 2B help the industry prosper, or will it help the industry commit suicide?"

---

[1] For example, see Kaner, "Uniform Commercial Code Article 2b: A New Law of Software Quality

and other sofware quality specialists, including Doug Hoffman, Brian Lawrence, and Watts Humphrey

[2] In its marketing materials, Kaner's publisher, Thomson Press, says that Testing Computer Software is now the best selling book in the area.

[3] "Personal Computers Now in 34% of U.S. Households", Software Publishers Association, Washington, D.C., April 22, 1996. (http://www.spa.org/research/releases/press1.htm).

[4] Annual Inquiry and Complaint Summary 1995, Better Business Bureau, Arlington, VA, 1996.

[5] Johnson, Bob & Gately, Amy, "SystemSoft to Develop Products to Automatically Identify and Resolve PC Users' Most Common Problems", Dataquest, March 12, 1996. (http://dqi2.dataquest.com/register/stories/swsv-na/swsv-na-da-9606-pv-0001.html. Access this through www.dataquest.com.)

[6] Dataquest predicted this number early in 1996. See Johnson, Bob & Gately, Amy, *op cit.* (Note 4) The same estimate has been provided by several industry experts in 1997 conference talks, such as Bultema, Patrick & Oxton, Greg, "Emerging Standards for the Support Industry", *Software Services Conference East*, Nashville, TN, March 11, 1997; Schreiber, Ron, "How the Internet Changes (Almost) Everything", *Internet Support Forum*, San Jose, March 26, 1997.

[7] "Benchmark Report: Technical Support Cost Ratios", 10 *Soft*Letter* #10, p.1, December 21, 1993 (reported median value=$23.33); Murtagh, Steven, J., *An Analysis of Cost-of-Calls in the Customer Support Industry*, Help Desk Institute, 1994 (reported median = $15.50, mean = $20.22); *Help Desk and Customer Support Practices Report: October 1996 Survey Results*, Help Desk Institute, 1997 (estimated average, p. xii = $25; median is in the $20-29 range, p. 25). The *1994 Customer Care Survey of Service & Support Practices in the Software Industry* gave an $18 median, but the 1993 survey estimate was $25, Customer Care, Inc., Tarrytown, NY, 1994, p. IV-45. The Software Support Professionals Association estimate is $25 per call for PC/Shrinkwrap products and much more for UNIX and Mainframe products. SSPA (members only) Support Center document rtr0011.

[8] Not all of these calls are for defects. Bill Rose, head of the Software Support Professionals Association (SSPA), estimates that "actual code defects make up only 10-15% of customer calls." (Rose notes that this total doesn't include calls triggered by errors in the product documentation, which he says is "the single most critical area for software vendors to improve.") Rose, Bill, *Managing Software Support*, SSPA, San Diego, 1990.

According to Goodman, John, Arlene Malech & Colin Adamson, "Don't Fix the Product, Fix the Customer", The Quality Review, Fall, 1988, p. 6, "New research indicates that two complaints out of three are caused by problems for which the customer is at least partly to blame, and which won't respond to traditional quality improvement methods." This research involved several non-software industries. James Bach, a leading quality consultant in the mass-market software industry, offers about the same estimate (38% of calls for software support are due to bugs) for software (personal communication).

We (Pels & Kaner) studied customer calls and letters in detail for one product and in less detail for some others. It is not appropriate for us to name the products or to elaborate on the circumstances, but we were in a position to know, in depth, the engineering design tradeoffs involved with these products. Our conclusion was that changes that could reasonably have been made by the product development department (i.e. bug fixes, documentation improvements, and minor design improvements to make the product more understandable) could have reduced the number of incoming calls by 50%.

[9] Based on an estimate of 15 minutes as the mean hold time per call. Ron Schreiber (*op cit.*, Note 5) estimated that callers were left on hold for 66,000,000 hours (3.96 billion minutes, which would correspond to a mean hold time of 19.8 minutes per call). Estimates of call hold times vary widely. For example, survey results from the SPA show a median of 2 minutes and a mean of 12.2 minutes. *1995 Technical Support Survey Report*, Software Publishers Association, Washington D.C., 1995, p. 17. The Customer Care Institute reported a median time of 4 minutes, but no means (*1994 Customer Care Survey*, *op cit.*, Note 6.) The Software Professionals Association

estimates average hold time at 15 minutes for PC/Shrinkwrap products and less for UNIX and mainframe products. SSPA Support Center document rtr0018. We think that these studies carry a sampling bias—the companies that service their customers least well are probably also least likely to answer these lengthy questionnaires. The SPA and the Customer Care Institute listed the companies that responded to their surveys and we noted the absence of several companies that we regard as troubled. Industry consultants frequently estimate 15-20 minutes at meetings.

[10] Brown, Dave, *Optimizing Support Center Staffing*, Service Management International, 1996, p. 43.

[11] SSPA Support Center document rtr0022.

[12] Schreiber, Ron, *op cit.* (Note 5).

[13] *Id.*

[14] Schreiber, Ron, *op cit.* (Note 5) estimated 18 times as long. Bultema, Patrick & Oxton, Greg, *op cit.*, (Note 5), estimate 3.3 – 17 times as long. Oxton, Greg, "Multivendor Support Challenges", *Software Services Conference East*, Nashville, TN, March 10, 1997, estimates that 36% of problems are multi-vendor but 75% of help desk costs are due to multi-vendor problems. (Remember that many of the problems that hit help desks have been seen before, so they have a quick solution even if they are complex.)

[15] *1995 Technical Support Survey Report*, Software Publishers Association, Washington D.C., 1995, p. 17.

[16] Wood, J.B., "Linking Customer Loyalty to the Bottom Line", *Software Support Professionals Association Executive Forum*, San Diego, CA, October 4, 1996. The SSPA also says that we are at a 10-year low in satisfaction with technical support. See Rose, Bill, "Email: Software Support's Secret Weapon" at http://www.sspa-online.com/publications/email.html.

[17] Kaner, Cem, Jack Falk & Hung Nguyen, *Testing Computer Software*, 2nd Ed., International Thomson Computer Press, 1993, p. 17 ff.

[18] Using the Watts Humphrey's PSP approach to long-term bug prevention, some companies have dramatically reduced their bug creation rates (at least in terms of coding errors, though rates of usability problems and other external design issues might be unchanged.) Ferguson, Pat, Watts S. Humphrey, Soheil Khajenoori, Susan Macke, & Annette Matuya, "Results of Applying the Personal Software Process," IEEE Computer, May, 1997, p. 24-31.

[19] There might still be bugs in the product that customers have not yet reported, including very serious defects that will occur only under rare circumstances. However, the list of defects reported by customers provides a useful estimate of the population of significant bugs remaining in the product at time of shipment.

[20] These estimates are all based on verbal reports from testing/quality control managers who take pride in their groups' work, and from one highly reputable independent test lab that told Kaner it keeps such statistics as a matter of contractual responsibility. These numbers are not widely published or easy to obtain and we do not know how representative they are.

We suspect that not all mass-market software publishers obtain results like these. In these fast-to-market times, several companies have cut back on their quality control budgets. They probably find and fix fewer problems.

[21] Kaner, Cem, Jack Falk & Hung Nguyen, *op cit.*, (Note 15), p. 13, 73, 95-96, 109-111, 113-114, etc. To the best of our knowledge, Kaner's first edition (*Testing Computer Software*, TAB Professional & Reference Books, 1988), was the first book on software quality control that acknowledged that it was a normal practice in the industry to not fix every bug. The book gave testers practical advice on how to research and describe bugs persuasively, to minimize the chance that significant problems would be left unfixed.

[22] Bach, James, The Challenge of "Good Enough" Software, Software Test Labs, 1996. (http://www.stlabs.com/bach/good.htm).

[23] A few companies do remarkably little quality control, and therefore their products contain few *known* bugs.

[24] For example, read Stephen Clancy's "Service Opportunities in the Home PC User Market", *Dataquest*, February 5, 1996, (http://dqi3.dataquest.com/register/abstracts/syss-na/syss-na-dp-9601-ab-0001.html. Access this through www.dataquest.com.) Also, Christopher, Elena L., "Desktop Strategies for Success: Maximizing Service Opportunity within Industry Trends", Dataquest, February 3, 1997 (http://dqi3.dataquest.com/register/abstracts/sysd-na/sysd-na-dp-9601-ab-0001.html, see www.dataquest.com) and see the press release at this site that summarizing

this, titled 1996 Begins Era of "Warranty Take-Back". It says “Desktop PC vendors began a "warranty take-back" trend in 1996, according to Dataquest. [P] This is one of several emerging trends in desktop service and support that vendors may find affecting relationships with end-users. But, upon further examination, each seemingly negative trend can hold a positive service opportunity, according to Dataquest.”

[25] Bertolucci, Jeff, “PC Reliability and Service: Good-bye to Good Support,” PC World, December 1996, p. 143-152.

[26] Ed Foster, personal communication, 5/14/97, based on much correspondence and study as the author of InfoWorld’s Gripe Line column; Bertolucci, Jeff, *op cit.* (Note 23).

[27] Schreiber, Ron, *op cit.* (Note 5).

[28] In a major study of call costs, over 80% of the respondents indicated that they did not know their average cost per call, and and almost half of the remaining respondents didn’t provide their average cost. Murtagh, op cit. Note 6. In 1996, 60.5% of respondents said that they didn’t know their average cost per service request. *Help Desk and Customer Support Practices Report: October 1996 Survey Results*, Help Desk Institute, Colorado Springs, CO, 1997, p. 25.

[29] Goodman, John A., & Grimm, Cynthia, J., “A Quantified Case for Improving Quality Now!”, *Journal for Quality and Participation*, March, 1990, p. 50. Goodman, John A. & Robinson, Larry M., “Strategies for Improving the Satisfaction of Business Customers”, *Business*, April-June 1992, p. 40, *and see their references*.

[30] *Id. and also see* Goodman, J.A., “Utilizing Customer Support Feedback to Measurably Improve the Value of Support”, *Software Services Conference East*, Nashville, TN, March 10, 1997; Wood, J.B., *op cit.* (Note 15).

[31] This reflects our experience and the experience of colleagues. Also, during his talk (“Utilizing Customer Support Feedback to Measurably Improve the Value of Support”, *Software Services Conference East*, Nashville, TN, March 10, 1997), Goodman stated that this was a common problem across industries.

[32] Initially published in *Soft*Letter*, which Tartar publishes. is a widely read and highly influential newsletter on mass-market software industry practices. This article is available on the web at http://www.asponline.com/rlib9.htm.

[33] Jones, Capers, *Patterns of Software Systems Failure and Success*, International Thomson Computer Press, London, 1996. Another widely published and respected author on software measurement points out that what measurements have been taken are largely unusable. Hetzel, Bill, “The Sorry State of Software Practice Measurement and Evaluation”, in Fenton, Norman, Robin Whitty & Yoshinori Iizuka, *Software Quality Assurance and Measurement: A Worldwide Perspective*, International Thomson Computer Press, London, 1995.

[34] From a presentation at Software Testing Analysis Review 96, Orlando, Florida.

