PO Box 6900
Incline Village, NV 89450-6900

September 11, 2000

Federal Trade Commission
Public Comment to
High-Tech Warranty Project – Comment, P994413

Reference: Notice in Federal Register, Vol. 65 No. 92, May 11, 2000, pp 30411-30413

Executive Summary

License-based software sales (which are still the best method) have encouraged software publishers to adopt extreme positions in the framing of these licenses, in order to avoid litigation and responsibility under warranty. Consequently, software prices are low. However, the extreme language of these licenses also causes problems for consumers when software doesn't function properly.

Software sales and warranty issues are also affected by piracy (i.e., software theft). Changes in warranties, as made in order to protect against piracy and in order to facilitate anti-theft regulations, should not interfere with the software publisher's right to be paid for its products.

This paper recommends that changes in software warranties and sales practices be strictly limited, in order to address the *real* issue, namely, how consumers can obtain better software, and how software publishers can provide it. Better disclosure---of the true state of software, of the right way to report bugs, and of how to obtain timely fixes---is the key. Communication should be better all around, with lawsuits the *last* resort instead of the first one.

This paper also discusses the expressive (as well as functional) aspects of software, and suggests approaches to the question of copyright and intellectual-property protection for software and software-based products.

Table Of Contents

INTRODUCTION 2
FORMAT OF CONTRIBUTION 4
RESPONSES TO GENERAL QUESTIONS 4
RESPONSES TO QUESTIONS ABOUT MASS-MARKET LICENSES 9
RESPONSES TO FUTURE TRENDS AN PUBLIC FORUM QUESTIONS 14
THE VIEWPOINT OF THE PUBLISHER---LITIGATION AVOIDANCE 15
WARRANTY AS A WEAPON AGAINST SOFTWARE PIRACY 16
ANTI-PIRACY MASS-MARKET SOFTWARE AND WARRANTY ISSUES 17
THE INTERNET, "PHONE HOME SOFTWARE," AND WARRANTY 19
RECOURSE NEEDS TO BE FAIR TO BOTH SIDES 21
CONSUMER ADVOCACY GROUPS NEED TO LEAD THE EDUCATION 21
COMPUTER PROGRAMMING IS ART, NOT SCIENCE 22
THE ISSUE OF CERTIFICATION OF PROGRAMMERS 23
CONCLUSIONS 24

INTRODUCTION

1. My name is Stephen Satchell. I am preparing this comment wearing a number of "hats": commercial software developer, Open Source software developer, author of documentation for software, professional reviewer of products for magazines and books, and a consumer of software.

2. In my capacity as a **commercial software developer**, I am the principal developer of a modem testing software package called "OTTO:3800", marketed and sold by Test Automation Software of Murietta, Ca. I also provide consulting services and contract programming to companies selling the results of my work to the open market. My experience began with mainframe computers in the 1970s, and with the APRAnet at the University of Illinois Center for Advanced Computation in Urbana, IL. When I moved into industry, I started working with minicomputer systems built into products in the publishing, then in the banking industries. During the 1980s and 1990s I developed for sale products for the Unix operating system., as well as product for sale that ran on IBM PC and Macintosh personal computers. In conjunction with my professional review practice (described later) I wrote, sold, and distributed benchmark and product testing packages.

3. In my capacity as an **Open Source software developer**, I have proposed patches to the Linux operating system, implemented certain extensions to functionality of Open Source programs, and have participated in security audits of programs.

4. In my capacity as an **author** of documentation for software, I have written user manuals for my own software and for the software developed by others, and also have a third-party book recently published: *Linux IP Stacks Commentary*, March 2000, Coriolis Press, ISBN 1-57610-470-2.

5. In my capacity of a **professional reviewer of products** for magazines and books, I have been writing reviews, product comparisons, and technology overviews for major publications since 1984. I am one of the founders of the *Internet Press Guild*, an organization created by myself and seven other journalists who were appalled at the inaccuracies in coverage of Internet issues to be found in such respected publications as *Time* magazine. (More information about the Guild may be found at our Web site: http://www.netpress.org.) In the course of the 16 years I have been writing, I was technical editor for *InfoWorld* magazine and also helped build two testing laboratories, the *InfoWorld Test Center* and *Ziff-Davis Labs* (now known as *Itest*).

6. My review that most influenced the industry was my critical product review published in 1985 of the IBM PC AT computer that appeared in *InfoWorld* and in *PC Products* magazines. In that review, I reported significant reliability problems with the 20 MB hard drive in that computer. Subsequent investigations with Peggy Watt, a senior editor at *InfoWorld*, led ultimately to providing IBM with enough clues to permit them to find and fix the problem.

7. In my capacity of a **software consumer**, I use commercial products on a daily basis in the course of my work. My average annual budget for software, including operating systems, is roughly $2,000 per year. This amount does not include the evaluation copies of software I received in my capacity as a professional reviewer or as part of my commercial software business. The software I use daily runs the range of applications: word processors, spreadsheets, book-keeping software, Internet communications programs, image readers /printers for fax service support, maintenance utilities, and the operating systems that provide the environment to run all these programs. My shop is a mixed one: you will find Windows computers, Macintosh computers, and Linux computers. Some computers are configured to run multiple operating systems. On occasion I have need to load other environments (BeOS, OS/2, FreeBSD, and others) and use programs that operate in those environments. Finally, my software suite includes small set of games that I play during those few times that I can devote time to that form of recreation.

8. For a majority of my 30-year career I have been an active member of both the Association for Computer Machinery (ACM) and the Institute of Electrical and Electronic Engineers (IEEE). Although I am no longer active in either organization, I still read the professional publications of both organizations, as well as many of the special-interest publications published on system programming, programming languages, software engineering, and telecommunications. I had originally joined ACM to work to improve the computer engineering curriculums, one of which I found to be inadequate to the needs of the working practitioner. I made specific suggestions for improving the curriculum, some of which I understand were adopted into the model curriculum for software engineers.

9. I am not a lawyer, nor a paralegel. I have received no formal training in law. I speak as a layman.

FORMAT OF CONTRIBUTION

10. In the first portion of my written comment, I wish to address each of the sixteen (16) questions posed in the Notice as published in the *Federal Register*. Each of my responses will be keyed to one or more of the questions by number. In the second portion of my written comment, I wish to address issues I feel are important that were not posed as part of the questions published in the Notice.

RESPONSES TO GENERAL QUESTIONS

11. [*Q1*] Based on my own experience, experience reported by others, and my investigations into warranty practices, the actual level of warranty protection varies widely from publisher to publisher, and even within a publisher's line of titles warranty protection can be very different from title to title. In some cases, the warranty systems used by publishers are so complicated and cumbersome that they might as well not exist at all. Other publishers are more aggressive about warranty issues, particular issues concerning the inappropriate or astonishing operation of software (commonly referred to as "bugs", a term which came into use during the days of telegraphy and made popular by Rear Admiral Grace Hopper telling the story of the "first computer bug", a moth that was lodged in a relay of the Harvard University Mark II Aiken Relay Calculator).

12. [*Q2*] In order to answer the question "what expectations do consumers have" we need to identify the types of consumers, because each type of consumer has different expectations.

- *The home user, non-professional*: This class of user typically has the least knowledge of computers, and appears to expect that the commercial software they purchase from reputable publishers will work as advertised, and with the same attention to quality that consumers have come to expect from manufacturers of kitchen appliances, automobiles, and entertainment electronics such as televisions, stereos, and boom boxes. Depending on what the computer is used for, their experience falls short of their expectations. In many respects, this consumer has a healthy fear of "breaking" the computer by doing something inappropriate, and when faced with a lock-up, or the Windows Blue Screen of Death (the screen Windows shows when a crippling fault within the OS occurs), are most like to cry "What did I do? What did I do?"

- *The home user, advanced amateur*: This class of user has usually purchased at least two computers in sequence, has either attended classes or participated in user-group meetings, and has learned from others how to recognize and recover from common faults. The expectation of this level of user has been diminished by hearing stories from other users, and in many cases his/her experience meets, and sometimes exceeds, this diminished expectation. Rarely does the experience of this consumer fall below the

lowered bar of expectation. In many cases, the "stories around the campfire" s/he has heard has alerted this consumer that software does have faults, and in many cases how to "work around" the faults of popular operating systems and software packages. Because of the support group behind this class of user, the experience with the computer products is less unsatisfactory.

- *The home user, computer professional*: This class of user is similar to the advanced amateur in capabilities and expectations. The primary difference is the source of useful information: professional societies, the Internet newsgroups, professional and industry magazines, and the network of fellow professionals. There is cross-pollination between the computer professional and the advanced amateur because many computer professionals either speak at or organize user group meetings. In addition, many computer professionals have a stronger business relationship with the publishers and manufacturers, thus affording the computer professional user with additional avenues of information, sometimes of the "back door" variety that is more useful than the publisher's public access points.

- *The business user, clerical*: This class of user rarely makes the buying decisions, relying on either an Information Technology (IT) department (large firm), consultants (small firm) , or The Boss (tiny firm) to procure the hardware and software to be used in the job. When something happens, most clerical workers call an identified person for help instead of trying to fix the problem themselves.

- *The business user, Information Technology (IT)*: This class of user is responsible for procuring, installing, and maintaining computers on behalf of companies. There is a growing consultant class that provides the same services to smaller companies that can't or won't maintain an IT department. These are the people in the trenches of dealing with warranty and support issues, and usually don't have time to interface well with the publishers and manufacturers ---their job is to solve problems and get people's computers up and going again and minimize downtime (and the resultant waste of labor). There are many parallels with the home computer professional with regards to access to information, and inhabit the same interpersonal networking space as the home computer professional. These members of the consumer corps are bloodied, and have low to moderate expectations about reliability---usually due to experience learned in the school of hard knocks.

- The business user, Integrator/Developer/VAR/VAD: This class of user is a "supercharged" version of the IT person. Because of the much closer relationship between this class of consumer and the publishers and manufacturers (in many cases there is a contract enforcing terms, and a formal exchange of confidential information not available to the other consumers)

the ability of this class of consumer to press warranty and support issues is much better than any other class of consumer.

13. [*Q2*] In short, the more knowledge the consumer has, and the less power s/he has based on contracts or other leverage, the lower the expectation of reliability. Interestingly, it is my observation that the lower the expectation, the more the consumer works to have a "plan B" or a method of recovering from the disasters that do occur. Most major hard disk loss, for example, happens to people who *never* expected their hard drive to throw a recording head.

14. [*Q3a*] Most consumers resort to the computer dealer from which they purchased their computers, assuming they purchased the computer from a dealer. In many cases, the dealer assumes the burden of dealing with the broken software, and more importantly with the after-effects that can occur when software breaks. For hardware, the original dealer is the conduit for warranty service on that hardware ---and for consumers, using the dealer option can mean many delays and large shipping bills to obtain service from the original dealer, because the warranty statements don't allow other dealers to perform the work.

15. [*Q3a*] Contracts and service agreements are used by larger business customers to contain the cost and loss due to failures in software and hardware. Leasing companies offer an alternative to the savvy business consumer ---that option lets the consumer say "here it is back, give me a new one" with the least hassle. Contracts and confidential exchange make the Developer/VAR/VAD customer's task easier to obtain warranty service---in many cases, the business partner makes fixes without reference to warranty in order to sustain the business arrangement.

16. [*Q3a*] Consumers who purchase through outlets such as warehouse stores, discount outlets, or the used-equipment market usually find themselves out of luck. Some software licenses don't allow the original purchaser to resell the software to another party, so the new consumers find themselves having to pay huge costs for warranty support that was promised to the original consumer but doesn't transfer to the new consumer. Some hardware warranties require original proof of purchase from an authorized dealer before service is rendered on a warranty basis.

17. [*Q3b*] What remedies are supplied by state or federal law? For software, the answer appears to be "none." From what little I've been able to learn about product law as a developer a properly worded license can remove any remedy from the hands of the consumer. "Sold as-is" is so common in software licenses and warranty statements that if it's not there it's a surprise. Also, software licenses tend to impose additional conditions that are unlawful, working on the premise that some customers will be scared away from starting legal actions against the publisher regarding violations of warranty law because the license says "no" and the software vendor typically has more money than the consumer if it comes to a court fight.

18. [*Q3c*] Which leads to the next question: would consumers seek to invoke the remedies? The answer is an emphatic ***NO*** for all but larger corporations. The issues are simple:

    - Most publishers and manufacturer have more money and more lawyers than your typical consumer
    - The law and rules are so murky and bulky that understanding the rules themselves is a barrier to using the remedies
    - The lawyers that do understand the law are hesitant to accept contingency cases from individuals and from small businesses, at least until there are enough of them to be able to bring a class action lawsuit

19. [*Q4*] Because virtually every software package warranty states that the product is provided "AS IS" the question is moot. Among corporate buyers, warranties are a "check-off" item, and as long as there is something there, the purchasing agents will typically not use the warranty as a buying criterion. Advertising may have a single line about warranty, when warranty is mentioned at all. Published reviews usually are more concerned with technical support than they are with warranty, although obvious or crippling product defects get prominent play in the review articles. In many cases, consumers learn of the warranty only when they open the box ---"break the shrinkwrap" ---and get access to the manual that contains the statement of warranty. For those who shop on-line, by mail, or by auction, that means the sale transaction is long since completed.

20. [*Q5*] The current consumer protections seem to work against the consumer ever seeing the warranty and statement of support before the purchase. With a few exceptions, learning about known defects is very difficult or impossible. [*Q6*] Existing law and industry practices appear to do nothing to encourage publishers to be open about the condition of their software. Indeed, many business consider their "bug lists" to be Company Confidential information. I was effectively fired from Motorola for disclosing a bug list even when a contract with the customer specified that I do so in a timely manner. This occurred in 1998. [*Q6*] My understanding is that there is very little remedy available to consumers, other than contracts that the consumer may have entered into with the publisher, when software fails. The availability of comparable software as a remedy is remote for all but the most popular applications, and seamless interoperability of replacement software is virtually non-existent. In many cases, the pain level of changing packages is so high that many consumers limp along with what they have…or give up entirely.

21. [*Q7*] The only activity I am aware of that would have an impact on customers rights is the growing Open Source Software movement to provide alternatives for certain commercial packages, and the growth of software publishers moving certain software from the proprietary in-house development path to an open-source business model. Examples include Netscape releasing an open-source version of its browser into the marketplace; the Free Software Foundation's release of work-alike

software for the Unix operating system; the expected release to the open-source business model of the StarOffice office productivity package. Another initiative, started in 1982 by the late Andrew Fluegelman, is the "try before you buy" method of selling software, today referred to as "shareware" and "demoware". The consumer can download or otherwise obtain a trial version, then purchase the rights to full functionality if the software works out.

22. [*Q7a*] **The proposed UCITA is a step backward** in the effort to provide consumers the rights enjoyed for many other types of products. The growing reliance on overly restrictive terms in licenses, including the imposition of prior restraint on First Amendment activity, means that consumers have less protection in software rights than in any other product class today.

23. [*Q7a*] In particular, as an ethical professional product reviewer, I am concerned about the language that some software publishers are starting to place in their licenses **restricting discussion in public of defects and deficiencies** of the licensed product. This is tantamount to prior restraint on speech and on the press. This eliminates a very useful source of information about software products: feedback from other users. Many user group newsletters include member-contributed reviews of products; those publications cannot stand the threat, let alone the filing, of a lawsuit against them. Such clauses, if left to stand, would cause a severe chilling effect on grass-roots consumers discussing what they have purchased.

24. {*Q7b*] I believe that the role of the federal government should be minimal with regard to protecting consumers who purchase software, hardware, or information services. I would strongly recommend that the States take up the responsibility, just as the States have taken up the gauntlet with regards to the UCC. I have the following specific recommendations, which should be imposed on the State level but could be imposed at the federal level:

- Require that licenses be accurate and legal in all respects, by saying that, irrespective of any separation clause that may be present in the license statement, that the entire license is void if any portion of the license has a defect or that the license does not confirm to the requirements of the State in which the software is sold.

- Provide a default software license in the event that a software license is invalidated as described above, or no license is provided.

- Request that the Software and Information Industry Association (SIIA) prepare and publish a model software license that has been reviewed by the FTC, the industry, and the public and that the model license be made available to all interested parties. The model software license would be used as the default license, as modified by each State as part of each State's adoption of the UCITA.

- Request that the Software and Information Industry Association (SIIA) publish, advertise, and information the public with the same zeal as they pushed the "Don't Copy That Floppy" campaign, a "Software/Information Consumer Bill of Rights" that describes the law in language consumers can understand, describes the remedies available to consumers when defective software is encountered, and provides a list of agencies that consumers can turn to for resolution of their problems.

- Require that software and information licenses warrant that the package operates as advertised, specify a time period (not to exceed 60 days) when fixes for reported problems shall be released, provide a procedure as simple as the procedure for purchasing the product to report a problem, and provide a paper document or Web page URL accessible before the sale at the point of sale describing all known problems (including cosmetic problems).

- Prohibit "AS IS" clauses for software that (a) is published without source, (b) by a for-profit company, (c) which is licensing the product and not just providing the software on the medium on which the software is provide at a price that is related to the actual cost of the medium.

- For software bundled with hardware or other software, the statement of warranty or license shall clearly indicate who provides warranty service on which software.

- If a software publisher wishes to take a title off the market, that publisher shall be required to release all source associated with the product into the public domain, or actively support the user base for a period of at least seven years after the product has been taken off the market for sale.

- In the event of bankruptcy, the publisher shall be required to sell the product to an active maintainer, or release all source associated with the product into the public domain.

25. [*Q7c*] I am not aware of any international developments prompting harmonization of warranty and service requirements.

RESPONSES TO QUESTIONS ABOUT MASS-MARKET LICENSES

26. [*Q8*] The impact of characterizing a mass-market software transaction as a license instead of as a sale of goods is emphasizing the true nature of ownership of software. In the sale of an item covered by copyright, the sale itself is [usually] the transfer of the right to use the copyrighted product for personal purposes in exchange for a fee. If the consumer wishes additional rights, for example the right to resell additional copies of the product, the consumer would be required to comply with the requirements of the owner of the intellectual property (pay a royalty per copy, pay a one-time fee for unlimited copying, give credit to the author ---the re-

quirements can take any form). In short, the normal and usual "sale" of software is the granting of a right to use.

27. [*Q8a*] Therefore, characterizing the transaction as a licensing arrangement instead of a sale makes clear that only certain rights are being transferred. This has been a long-standing practice for 30 years, and despite the abuse of license practiced by some software publishers, it is in my opinion the right way to categorize the transaction.

28. [*Q8b*] One of problems with software is that, **unlike virtually every other form of copyrightable material, there is a *functional* aspect to the copyrighted work** that doesn't apply when looking at the "warranty" for books, magazines, audio recordings, visual recordings, performances, the vast majority of artwork, musical scores, and the other forms of "expression" covered by Title 17. This tension in the law is clear by the conflicting considerations of software as "speech" as exemplified by the encryption suits[1,2,3] versus the DeCSS suit[4]---in some cases software is "speech", in others the functional aspects take precedence. Is it functional, or is it expressive?

29. [*Q8b*] The legal ramifications are that software is the *only* type of "expression" in which the expression itself can be "broken" in any way. A poem can be bad, but not broken. A motion picture can be poorly edited and have content that is criminal (such as child porn) but the expression is and of itself just that, expression. An artist can create a sculpture, and the nose broken off can constitute damage that requires fixing, but even the broken nose could be an element of the expression and not a "defect". (Copies of an art piece could be defective in manufacture, however.)

30. [*Q8c*] This defect in the copyright treatment of software leaves the consumer in the lurch. The actual expression, perhaps with a patch, is protected by copyright and further protected by statements in the license agreement that might prohibit "reverse engineering" even when that reverse engineering is for the purpose of applying a "patch" to fix a problem. When a competitor desires to come up with a replacement for the defective software, he is faced with two problems: first, he has to develop the entire package *from the ground up, including initial design*; and he also has to be sure that in doing so he does everything right. Sorry, but Murphy's Law says that won't happen.

31. [*Q8c*] In addition, the license is a tort between the artist and the consumer. In mass-market software, the consumer is powerless to negotiate any element of the contract for sale, which is what the license agreement is. That means that the li-

[1] Daniel J. Bernstein v. United States Department of Justice, et al., No. 97-166686 (9th Cir., May 6, 1999)
[2] Peter D. Junger v United States Department of Commerce, No. 1:96-CV-1723 (6th Circuit)
[3] Press release of July 8, 1998 by Mr. Junger: "There is thus a clear split between the two courts: Judge Patel holding that computer software is protected by the First Amendment and Judge Gwin holding that it isn't."
[4] MPAA members v 2600 et al, No. 00 Civ. 0277 (LAK) (SDNY, August 17, 2000)

cense can remove all rights to resell the software under any circumstances; indeed, it has been shown that a license agreement can prevent a consumer from moving a piece of software from an old computer system to a new one, or to install a package on a new hard drive when the old hard drive ceases to function properly. In most cases, the consumer is forced to ignore the contract, or to spend even more money on replacement software.

32. [*Q8d*] Software has an *expressive* quality and a *functional* quality. Current law does not recognize that fact consistently, and the Courts are hard-pressed to tell the difference. The two faces of software need to be identified, segregated, and handled properly.

- **Software as expression:** Congress should clarify that software in human-readable form ("source code") is expressive for the purposes of copyright and for the purposes of First Amendment protection of free speech. It's no different from a chemist "talking" with structured molecular formulas, a mathematician "talking" in obtuse symbols, or a lawyer talking in inscrutable Latin and in words defined not in Websters but Black's Law Dictionary.

- **Software as function:** Congress should clarify that software in machine-readable form ("object code" or "executable code") is tantamount to the paper roll used in a player piano, the wooden bars used in a Jacquard box for a weaving loom, or a piece of paper tape used to direct the cutter in a milling machine. The instructions are just that: instructions. Those instructions guide actions, and are liable to fault just as much as the fluid computer in an automatic transmission, or the electronics of a biomonitor in a hospital. Executable object code should be handled exactly the same way that a brake pad is, or a toaster. It is a tangible item, even though extremely easy to copy.

33. [*Q8d*] My strong recommendation is that software sold in binary-only form be treated exactly the same way as any other item sold in a grocery store, hardware store, stereo store, or other outlet of tangible items. Indeed, we already treat embedded software in exactly this way. There is software in a microwave oven, yet the microwave oven has no special treatment or a "software license". There is software in the emissions control computer in your automobile. There is software in your digital radio. Yet I have yet to see any "software license" or other nonsense.

34. [*Q8e*] DVD might as well be sold under license instead of a "product", as the DeCSS case demonstrates. What would be better is for a license to the *content* be granted, and the container for the product be sold separately. So, for example, if I want to purchase the right to view without limit the movie *The Matrix*, I purchase the right to do so, then with that license I can get the VHS tape, or the DVD, or the next decade's gee-whiz movie storage format---but I purchase the right to the content *once*.

35. [*Q8e*]The same thing would hold true for musical albums: I've purchased several copies of Pink Floyd's *Dark Side of the Moon* in several different media, and paid not only for the physical media but for the license to hear the music multiple times. How did I purchase it? The original vinyl, the "Master Works" vinyl, the cassette tape, the reel-to-reel tape, and the CD. Should I pay yet again for a license to the content I've already purchased several times when the next great audio format (such as DVD-audio) is rolled out?

36. [*Q9*] How do software licenses create express warranties? Most of them say that the *only* warranty is a 90-day guarantee that the medium used to carry the content will be free of significant defect. With the growth of Internet sales, even that guarantee has gone by the wayside.

37. [*Q10, Q11*] As for implied warranties, every single software license I've seen for the past five years gives a list of all the warranties that are *not* provided. I'm led to assume that anything that is disclaimed is somehow a warranty implied otherwise. Perhaps that is a naïve assumption, as lawyers tend to be cautious creatures and would include things that *might* come up as litigation points. Each year, as I purchase software, the list of things not covered keeps growing.

38. [*Q12*] The problem with "shrinkwrap" and "clickwrap" licenses is that they are rarely available at the time of sale. Indeed, the last seven software packages I have purchased did not have the license and warranty text available at time of sale. Some of those purchases were made in stores; others were made over the Internet. Some of the license terms were real shockers, too, and would have affected my choice of software to buy. Frankly, I suspect that software publishers make the terms as hard as possible to find so that purchase decisions are *not* affected by license or warranty terms.

39. [*Q12a*] I don't use shrinkwrap licenses, so I've never had to test one in court. I don't violate licenses, so I've never been taken to court for my practices.

40. [*Q12b*] Terms in a shrinkwrap license: no reverse engineering; no resale; as is; usability is solely the customer's problem; break the shrinkwrap and you agree (even when the "agreement" is bound in the manual contained in the shrinkwrap); prohibition of public statements denigrating the software; must run on a single computer; software may not be transferred to another computer under any circumstances; use at your sole risk; software subject to unilateral recall by the publisher; if any clause is found to be illegal, everything else stands.

41. [*Q12b*] Here is an interesting new one: "You may use the Software only on a stand-alone basis, such that the Software and the functions it provides are accessible only to persons who are physically present at the location of the computer on which the Software is loaded." Another new killer: "You acknowledge that the Software contains trade secrets and other proprietary information…"

42. [*Q12b*] And there is this interesting new thing I just saw, the "rat out" clause: "In any event, you will notify <company> of any information derived from reverse engineering or such other activities, and the results thereof will constitute the confidential information of <company> that may be used only in connection with the Software."

43. [*Q12b*] At least one company has attempted to extend its license for software to cover *hardware* sold (or in this particular case, given away) with the software. Installing the software binds you to the concept that the hardware is "on loan" and subject to recall by the publisher without recourse by the consumer. No such language appears on the device, the instructions, or the packaging.

44. [*Q12c*] The license terms that are beneficial to consumers are the ones that force publishers to produce quality products, to fix bugs in a reasonable time, and to disclose information useful to the consumer in making choices. Publishers don't like to do this, because it increases their risks and liability, which means they have to charge more in order to cover the potential legal fees. By disclaiming all responsibility, even the responsibility of making a product that is accurately described in their advertising (otherwise, why disclaim all fitness and merchantability?) the publisher is free to push the product without fear of significant litigation.

45. [*Q12c*] The benefit to most consumers, I would believe, is that the lack of litigation costs means that the price is lowered for the products, and the profit margins kept up so that the company stays in business…until some other company buys them up for a specific technology and dumps the products it doesn't want, or dumps products that compete with the buyer-company's cash cows. As for legal recourse, what can you say to a judge? "Hey, the company said that it sells the software as is, I have to determine whether it meets my needs, and they disclaim all warranties. I want to sue them." That judge would laugh the case right out of court, because the case has *no* merit. Exit consumer, poorer by the purchase price of the piece of worthless software and the court costs.

46. [*Q12d*] *To what extent are the terms of shrinkwrap/clickwrap license currently available to interested consumers prior to sale?* You're joking, aren't you? The computer stores do not have open packages of all the software so you can gaze at the verbiage that is almost incomprehensible anyway. In many cases, the shrinkwrap licenses are built into the installers, and there is *no* printed or viewable version of the "agreement"---this is especially true for downloaded software, where the Web page that describes the product and takes your credit card number doesn't include a link to the agreement. Indeed, the last time I tried to view a software agreement I found myself in a "Catch 22" situation: in order to view the license I would have to break the shrinkwrap, and in order for the clerk to let me break the shrinkwrap I had to pay for the software.

47. [*Q12e*] With regard to alternative dispute resolution, I have never had a case that could be put to arbitration or other alternative dispute resolution methods. As a developer, I have never had a customer have a complaint that has gone that far.

48. [*Q12f*] I believe that firms are unwilling to compete based on license terms (particularly shrinkwrap or clickwrap) because such competition would expose those companies to increased litigation potential. All full disclosure would do is decrease sales across the board---because people would recognize the software situation for what it is: a crapshoot.

49. [*Q13*] I believe that software licenses for mass-market products honor the Magnuson-Moss Warranty Act mostly in the breach; specialty software and vertical-market software mostly honor the terms of the Magnuson-Moss Warranty Act. The problem is that very few people have the $30,000 required "just lying around" to even begin to sue companies for violations of 15 USC 2301 et seq.

50. [*Q13a*] There is a big problem with bundling everything that is "software" together into a big lump. Software sold in the retail channel, "retail channel" being defined as shrink-wrap software sold in computer stores, department stores, discount houses, via the Internet, via mail-order houses, when there is no negotiation of terms of the license (including price) or when a single amount of money is to be paid, then it should be treated as a consumer product. If the license is for a lease in which payments are made month-to-month, or the terms of the agreement are negotiated, then the software should not be treated as a consumer product.

51. [*Q13b,c*] As for classifying software as "tangible personal property" and classifying the transaction as a "sale", I believe I have provided sufficient reason to say "no" to these two questions. Just as the words in a book are not tangible personal property, neither should software be.

52. [*Q13d*] To answer the question "should software licenses be treated as 'warranties' subject to the Act" we again need to differentiate between retail-channel software and other software. To the former, the answer is "yes" because the software license confers specific rights to the buyer. To the latter, the answer is "no."

RESPONSES TO FUTURE TRENDS AN PUBLIC FORUM QUESTIONS

53. [*Q14*] **I object strongly to the proposed amendment to UCC Article 2** on the grounds that the terms of sale form an integral part of the contract of sale between seller and buyer, and that postponing disclosure of terms is a form of deception that needs to be removed from the marketplace, not added to it. No sane business will let a consumer introduce post-sale terms into the transaction; why should the seller allowed to be different in this matter?

54. [*Q15*] The summary as published in the Notice is a good outline of the focus and scope of the Commission's initial forum. My only request would be that the scope be broadened to include non-consumer software so that the differences and similarities can be examined as part of the investigation.

55. [Q16] I believe that all interested parties need to be present to express their views, their concerns, and most importantly their agendas at the public forum. Specif i-

cally, I would encourage invitation to and participation by developers, publishers, consumers, consumer groups, industry groups, members of the Executive Branch, members of the Legislative Branch, and judges.

56. In order to have the largest possible breadth of input, I suggest that the forum be hosted in part on the Internet, where people who are unable or unwilling to travel to Washington DC and pay the cost of hotel, food, transportation, and so forth would still be able to participate. This would reduce loss of income/revenue of interested parties located in the Western and MidWestern portion of the United States, particularly participants from California, Nevada, Oregon, and Washington, states which are arguably the nexus of mass-market software development in the United States.

THE VIEWPOINT OF THE PUBLISHER---LITIGATION AVOIDANCE

57. The questions appear to look at software warranty and license from the viewpoint of the consumer, and that's part of the story. The other part of the story, however, is software warranty and license from the viewpoint of the publisher.

58. First and foremost, the first priority of the publisher is to earn and keep money. The way that most publishers do this is to cause to be produced a product that will be purchased in some manner by consumers. Anything that gets in the way of collecting money from consumers and keeping that money has to be dealt with. For software publishers, the most common irritating drain on income is the cost of running a support desk. Publishers use a number of methods of reducing the financial drain of support, in particular relegating support issues to methods that don't involve significant labor (e.g. Web site support). Indeed, Microsoft Corporation instituted a huge program, known as the Window Logo Program, to reduce technical support calls for third-party devices used with its Windows operating system. Peripheral makers who wish to use the logo design their products to well-published guidelines, submit their product to Microsoft for compliance testing, and certify that the company has adequate help-desk support available for the customers.

59. Interestingly, advertising is considered less of an evil, even though in monetary terms the spending may be higher. The difference between advertising and support is that the former can be subject to a planned budget, while support is an open-ended spending issue that depends on consumer demand. It's interesting to note that a good product, well-designed and well-documented, generates less technical support even when sold in large volumes than a poorly designed, poorly documented product sold in tiny volumes. The expense follows the call volume, not the sales volume.

60. For traditional tangible-product companies, litigation appears as an expense item fairly high on the list. For some products, the litigation exposure is relatively low for large-sales items; for others, litigation exposure is a significant problem. Certain fields tend to have higher litigation expenses (litigation can also include regulatory expense, such as drug companies with the FDA) and so have to adjust

latory expense, such as drug companies with the FDA) and so have to adjust pri c-ing to cover the expected litigation load. Software companies, by using very r e-strictive licensing terms, tend to hold down exposure by tell the customer "buyer beware."

61. The effect of this ability to turn away litigation before it happens has contributed to the price decline of software. If the exposure to litigation is increased, the publis h-ers will have to respond by increasing the reserve in their budgets for litigation e x-penses, **which would in turn lead directly to an increase in prices for software titles.** This is the "unintended consequence" of knocking down some of the "abuses" in software licensing today.

62. Very few companies like to be in the business of being sued. In his book *A Feast For Lawyers* (1989, ISBN 0-87131-589-0), Sol Stein characterizes the entry into bankruptcy as a company leaving the business of what it was doing and enterin g the business of bankruptcy. As a software developer I have seen the fear in the eyes of publishers of just this sort of business mid-life crisis: a blurring of just what the publisher is in the business of doing. This is particularly true of the small businessman, who is less likely to survive a series of lawsuits than a large co m-pany.

63. Finally, there is a chilling effect if there is a higher risk of lawsuits for particular types of titles. So-called "man-rated" software---software in which a bug can k ill someone---is very, very expensive not only because of the care that has to go into catching and eliminating every bug, but also the liability reserve that must be i n-cluded in the budget to cover potential lawsuits in the event something isn't pro p-erly caught. You find such software in plant automation control, in avionics, in people-mover machine control (such as for elevators), and in medical instrument a-tion and control software. Now, I know of no mass-market software that comes even close to being "man-rated" in its function, but some people would believe that errors in software that affect their net worth should be held to similar standards. If those consumers were able to litigate that expectation, the effect on software prices would be horrible to contemplate.

WARRANTY AS A WEAPON AGAINST SOFTWARE PIRACY

64. One large issue facing software publishers is both casual and organized efforts to make unauthorized duplicates of software. These acts are referred to collectively as *software piracy*.

65. While dif ferent companies have different definitions for types of piracy, Micr o-soft's description of the four classes of piracy appears to be the most accepted on by the industry (http://www.microsoft.com/piracy/basics/default.asp). *End user pi-racy* is the casual copying of software, including disk swapping. *OEM piracy* is the sale of software with a system without a license from the software publisher to do so. *Counterfeiting* is the duplication of software, at varying levels of quality, for

sale. *Mischanelling* is the inappropriate sale of software destined for a specific market, such as selling to anyone an academic version of a product. Not included in this list, but more and more frowned upon by software vendors, is rental or loan of software.

66. Theft of software is unique in its effects among copyrighted works. When someone make a wholesale copy of a book, a duplicate of a phonograph record, a knock-off cast of a piece of scuplture, or a bootleg performance of a play, the loss is restricted to non-payment for the rights to do the act, and a very remote possibility of defamation of the character of the artist, composer, or playright because of inadequate attention to quality by the infringer. The act of copyright infringment in traditional works of copyright very rarely involve the copyright holder having to pay money out because of the infringement.

67. Not so software: the necessity for technical support---help for the *functional* aspect of the copyrighted work---costs money to provide, and when a person obtains an illegal copy of the software and calls the publisher for assistance in making the ill-gotten package to work, **this takes money out of the pocket of the publisher**. In this unique respect, software copyright infringement represents a larger loss than any other form of copyright infringement to date.

68. Software publishers also realize that the need for support is one way for them to control piracy. If you, the consumer, can't prove you are a legitimate owner of the right to use software, the software publisher refuses to support you, and the value of that pirate copy diminishes as a consequence. For example, if I were to receive a telephone call from a person claiming to be a customer of my modem testing software, and I don't have that customer on my list of valid customers, I would immediately inform my publisher of the piracy, and the publisher would [I assume] take appropriate legal steps to fix the problem.

69. For niche-market software such as mine, this brute-force technique works very well. There is a contract between my publisher and my user, and I'm made aware of the business relationship. While this blocks inter-company piracy, it does nothing for too many installations of software within the same company.

ANTI-PIRACY MASS-MARKET SOFTWARE AND WARRANTY ISSUES

70. For mass-market software, there is a problem: how do you validate users? One way is to have a token of some kind that a legitimate user must have. The token can be a hardware device, it can be software in the form of an *enabling key* (known in the Unix world as *serial-number-password*) which is included in the package, or it can be an *activation code* distributed to a user at the time s/he registers the software with the vendor. Finally, the use of digital certificates is growing in favor.

71. One common hardware device is the *dongle*, which the software interrogates to verify from time to time to be sure the user is authorized to run the software. The reason the key device is interrogated multiple times is to protect against hot-swapping

a key between multiple computer stations. These devices normally plug between the system unit and the keyboard, or between the system unit and a printer; other versions exist as well. Dongles are used on high-ticket software. Indeed, one view of software sales when a dongle is involved is that the software is "free", but the $2,500 pays for the dongle, the key to unlock the software.

72. Another method that works with some computers, and some versions of other computers, is to use a serial number source embedded in the computer to "mark" software as it's installed. For example, a software publisher could interrogate the serial number of certain versions of the Intel Pentium line of microprocessor chips and read out the unique serial number from the chip. During the execution of the software, the software could read out the serial number from time to time to be sure the software hasn't been moved from machine on which the software was originally installed. No serial number in the CPU chip? There are other devices that have unique numbers, such as the 48-bit Ethernet hardware address on network interface cards.

73. Activation codes are used extensively with "try before you buy" software. These codes are a key to unlock the full functionality of a package. Before the code is entered, some functions of a software package may be disabled, or the software is fully functional for a trial period. When the user decides the package will fulfill his needs, he pays a fee to the software publisher, and the software publisher provides a special key sequence to be given to the program. Once the program's registration software sees a valid code, it conditions the rest of the program to function normally.

74. Software certificates are files of code (typically 300-1000 bytes) that include considerably more information that the short activation codes used with shareware. These certificates can encode time limits, the name and company of the user so that unlocked versions of the software are branded to the user, and additional information.

75. Certificates are especially useful for software that is part of a service. For example, a warranty-claim preparation software package creates claim forms that meet certain requirements of the Computer Technology Industry Association (CompTIA). One of those requirements is that each warranty claim form have a unique claim number, and that the claim number be machine-readable in the form of a 3-of-9 bar code. Another requirement is that the codes used in a claim must be current. A software certificate would contain information that verifies that the software is using the correct version of software, the correct version of the code database, the range of numbers assigned to that particular user, and the date on which the certificate for the service expires.

76. There is another form of piracy prevention that will be discussed in the next section on "Phone Home Software."

77. For many of these control measures, the failure of a component of a computer, or of the entire computer itself, does not prevent the user from moving the software to a new computer or replacement peripheral. Unfortunately, some of the measures that use unique characteristics of the computer (such as a network interface card address) fail drastically when the component being used to "brand" the software to the machine fails and is replaced, or is upgraded to a higher-performing product (such as when a 10-megabit network card is replaced with a 100-megabit network card). The hassle in trying to get all the software working again can take days, because the software publishers are unable or unwilling to assist the consumer. In some cases, the consumer has no choice but to purchase another copy of the software.

78. When the hardware key devices fail, the software is useless without it. In my experience, the software publisher who utilizes such devices to protect software work "regular business hours" and require a complex procedure be followed to exchange the hardware key device. Heaven forbid you are using dongle-protected software over the Christmas holidays and your dongle dies---you are out of business until the New Year---if you survive that long.

THE INTERNET, "PHONE HOME SOFTWARE," AND WARRANTY

79. There are software packages used as parts of on-line services that make extensive use of the Internet as part of their function. The most identifiable example of this sort of software is the AOL software package. This package acts as a "front end" for users of AOL, reducing the amount of information sent via the communications link between the front end and AOL servers. The range of Internet-using front-end software packages doesn't stop with portal access software like AOL's. A number of electronic mail clients also serve the same capacity, exchanging mail using well-known protocols (Post Office Protocol [POP] and Simple Mail Transfer Protocol[SMTP]) with mail servers. Other more-specific software packages provide a graphic interface to stock ticker feeds. The list goes on.

80. Indeed, this linkage also permits service vendors to more closely control the consumer's use of a given service. For example, the electronic mail package may fetch mail from a mail account only after it has properly identified the user. This prevents an unauthorized third party from collecting the user's mail. Similarly, a vendor of stock ticker information can ensure that the user asking for real-time stock quotes has paid the appropriate license fee to one or more stock exchanges.

81. Software packages in general are starting to use the Internet in other ways. For example, many anti-virus software packages include Internet portals to make the update of virus signature databases easy, quick, and secure. Software package registration functions interact with the software publisher's database to eliminate errors caused by the keyboarding of data from classic paper registration cards. Finally, on instruction of the user the software can "phone home" and find out if there are any updates.

82. This last function is a real boon to software warranty, because it can relieve the software publisher of the twin problems of notification and delivery of updates. By using the "phone home" method, the user can indicate his/her desire to learn of updates, and can specify his/her preference regarding installing those updates now or later.

83. As well as serving the publisher-to-consumer communications needs, many software packages now include the ability to submit problem reports via the Internet. One advantage of this system is that the software in the consumer's program can include significant setup and debug information into the problem report, giving the support engineer information necessary to track down the bug without having to quiz the consumer endlessly regarding how the software is set up, the version of the operating system in use, and what else may be installed in the computer that would affect the operation of the software.

84. This is all well and fine when the software publisher and developer make it clear what the software is doing, and obtains informed permission from the user to do so. The dark side to this, though, is that some software publishers are *not* disclosing what the software is doing "behind the back" of the consumer. For example, the Microsoft Windows 98 SE operating system has a function, installed without notice, that will cause the system to query the Microsoft Web site *every five minutes* for update information…and do it without asking for permission from the user of the computer. Further, because there is no user moderation or ability to monitor the communication, the content of the communications can be suspect.

85. In some instances, software violates the policy of the network on which the consumer's computer is connected. These surreptitious "phone home" incidences violate company security regulations, regulations that require *any* communication with an outside agency to be monitored by the employee to ensure that no company confidential information is transmitted beyond the boundaries of the company's firewall. It can also consume bandwidth not authorized by company policies as well, which companies consider to be a theft of service. (And you thought that software publishers tried to avoid *all* litigation, didn't you?)

86. Lack of disclosure can cost an employee his or her job when the company security team detects the transmission and finds that the employee didn't know anything about it.

87. While the issues described in the paragraph lie more in the realm of privacy concerns, they do impact warranty and service in that the software publisher needs to be aware of the need to ask permission, to disclose completely the nature of the transmission, and to limit the nature of the transmissions such that company policies are not violated.

RECOURSE NEEDS TO BE FAIR TO BOTH SIDES

88. The fear of litigation on the part of publishers has caused them to avoid telli ng the customer anything about how to fix problems---it's better to lose the customer than to open the floodgates of legal action. This stems from the attitude of people that when things go wrong, the strong go to court. This is wrong.

89. Or is it? It depends on the situation. If a consumer has a problem with a product, can the consumer contact a company representative that can help? If the first fix doesn't work, how does the company escalate the problem? In a company I worked for as a "back-room engineer" I found myself going out on airplanes to customer sites when the situation went beyond the ability of the field engineer and the site analyst to handle. The management made it clear: they didn't let problems fester, but sent development people out to find the problem when the field people couldn't solve the problem. (This tendency to send development people out on field calls gave us developers motivation to design and install more diagnostic tools into the product.)

90. Currently, there are a number of companies who tell its consumers that service is not a major concern to the business. "Like it or lump it" the publishers say. It's those companies that cause consumers to get a little hot under the collar, and to jump to legal remedies a little early wi th companies that would just as soon fix the problem.

91. My strong recommendation is that companies be encouraged to publish their "bug lists" and the fix dates for them. Ascend Corporation (now a part of 3Com) and Lucent Technologies both did this with thei r remote access modem products for years, and they didn't seem adversely affected by the public disclosure of product problems.

92. Even more importantly, I strongly recommend that companies devise a policy for handling customer complaints, and publish that po licy to the customers. Give details like telephone numbers to call, information to collect before calling, and how to track a problem through to a fix. If a company has a charge -per-call system, indicate how reporting a real bug will not result in a char ge to the consumer.

CONSUMER ADVOCACY GROUPS NEED TO LEAD THE EDUCATION

93. We have a large number of consumer advocacy groups working to improve the lot of the consumer. I would call on the advocacy groups to form an industry association, and come to some common ground on the issue of consumer protection and consumer education. Generate materials to be distributed via the advocacy groups, and that association should encourage publishers to adopt and document appropr iate ways to improve the communication between buyer and seller---particularly ways to decrease the cost to both seller and buyer.

94. My comments about costs earlier referred to the almost uncontrollable cost of support for software publishers. What most people forget is that there are other costs: the time the consumer takes to try to resolve a problem; the productivity lost when a piece of software breaks and there is no work-around; the consequential losses when work turns up late. In my translating business, there is nothing more upsetting to our clients than when we have a hardware crisis that not only stops the work, but has us lose the last six hours' work as well. I'm not saying that publishers should have 24-hour turnaround on bugs, but publishers need to understand that there needs to be a balance between profit and service.

COMPUTER PROGRAMMING IS ART, NOT SCIENCE

95. Stanford University professor Donald Knuth wrote what is considered one of the most important series of books about computer programming. It is no accident that the title of the series is *The Art of Computer Programming*. I emphasize the word "Art" in the title. Even though computer programming sprang from our schools of mathematics, in fact the creation of instructions for computers involves the same sort of ingrown skill that separates poets from hack writers, separates master painters from daubers, separates effective leaders from rabble-rousers.

96. Frankly, you either have what it takes to be a programmer, or you don't---no amount of force-feeding is going to make a programmer out of just anyone. In most cases the best programmers come with attitudes and belief systems that are at odds with those of the typical non-programmer employee---Scott Adams has captured many of these interesting traits in his comic strip *Dilbert*. Some people can deal with the zaniness (and Microsoft has shown that catering to the zaniness reaps big rewards) and others can't stand it.

97. The creation of software---and don't fool yourself, it *is* a creation effort---is not an exact thing. Attempts to dictate methods, procedures, and style have created schools of programming, not unlike schools of painting, and with the same unifying effect---none. The results are pretty much the same: the code that comes out of programming teams has some number of inappropriate constructions. Many of these inappropriate constructions are harmless, which is why even a mature operating system has been found to have a number of ten-year-old "bugs" that don't trip anyone up.

98. In particular, the most insidious bugs to find and squash are those involving real-time processes. The aptly-named *race conditions* (which occur in hardware as well) depend on two tasks that used to always operate A->B being executed B->A instead…and if the processes A and B have a shared resource, the change in sequence can have astonishing results. Furthermore, if the timing is caused by external stimuli, then that can lead to so-called "every-other-fortnight bugs"---bugs that are very, very hard to find and eliminate.

99. The various schools of programming style have provide partial solutions to one major problem: software maintenance. Software maintanence is the process of fixing software (repair) and adding features (extension). The issue at hand is that a large software project will have many hands involved in the writing of the package. In many cases, the original developers will move on to new projects once the product is released, so that any subsequent development will be done by a different person. (There is an axiom in the industry that a person will change over time, so that the same person approaching six-month-old code is in the same situation as a completely new person looking at the code.)

100. Software maintenance is where new bugs get introduced, or previously working code is broken. This is why an update can take longer to release than the original product. First you have to detect that you made an unexpected change, then you have to find the change that caused the error (some programmers don't bother with history), then you have to develop a fix for the problem. Of course, the fix can also create a bug, so this becomes a spiral…or a whirlpool.

101. It's very hard to prove a negative, which is why the search for zero-bug programming has turned into a Holy Grail of the profession…and has yielded the same results in the search, next to none. Oh, some interesting methods of *reducing* the number of bugs have come out of the research, and the growing desire to reuse the exact same software (on the theory that there are fewer bugs in old, stable code than in brand-new, untried code) has recently come to the fore.

102. The growth of graphic user environments such as Windows and the Macintosh has only made the problem worse by introducing the possibility of real-time race bugs into *every* program written. This situation occurs because now just about everything is controlled by an external event ---mouse click, keyboard press, external I/O---that can come at any time and in any order. Because of the randomness of external events, debugging has become harder, and almost impossible to debug "over the phone" with a user unless the programmer has installed lots of debug tracing into production applications---a practice which can bloat the quantity of resources needed to run the program and add another level of complexity to the software's interface.

103. Clearly, there is a trade-off between the desirability and the cost of finding all bugs in a program. Go too far toward detection and the publisher pays in terms of higher development costs and delays in market release. Go too far away from detection and the consumer has to deal with an unworkable or useless program. Setting this balance needs to be done in the marketplace, not the courtroom.

THE ISSUE OF CERTIFICATION OF PROGRAMMERS

104. When the subject of "warranty" comes up, there is always someone in the audience who speaks up and points to the "Certified Engineer" who is required for many public-works projects such as bridges. This is the person stuck with the respons i-

bility for the safety and performance of the project. When a platform falls, this is the target at whom everyone points their lawyers. A similar concept, a "Licenced Engineer" or "Licenced Contractor" has been brought up a few times, too. I believe this is a dead end.

105. There has been a number of calls for certification of software programmers and software engineers. Every single proposal I have examined has been lacking, because the certification process lacked any means for detecting if there was any talent for the profession. Most of the proposals, if translated to a certification for landscape painters, would concentrate on the mechanical processes of squeezing paint onto a palette, cleaning brushes, telling the difference between rock, tree, and water, the depiction of light sources, and balancing light and dark.

106. Other certifications depend on the vetting of a programmer by someone. Usually this "someone" is a member of a University faculty, a previously certified practitioner, or a bureaucrat. That doesn't work well, because the attitudes and personal habits of the certification candidate can easily offend any or all of the examiners. At an annual conference of very competent people (many who are well-respected programmers) the steering committee for that conference makes this flat statement: "Being obnoxious is not sufficient reason to 'un-invite' someone to the Conference." I can personally attest that this policy is indeed observed, to the letter.

107. And how do the candidates feel? In many cases, the certification candidate has no respect for the examiner, especially if the examiner has not done any significant work for quite a while. (After all, if the examiner has time to do examinations, he obviously is lousy at his craft; otherwise, he would be writing code!)

108. I believe that the search for the perfect certification system for programmers and computer engineers is as hopeless as the search for bug-free code.

CONCLUSIONS

109. In this contribution, I have tried to answer not only the questions asked in the Notice but also other questions surrounding warranty of software products. I have focused on the dual name of software as "expression" and "function", and how any consideration of warranty regulations for software products needs to take into account this duality. This is particularly true with regard to Magnuson-Moss.

110. Current practice is for software publishers, including the publisher of my own software, to write licenses that disclaim as much responsibility as possible. This is not an indication that the program code is bad, but is rather a mechanism for staying out of messy litigation. Because there is no penalty for over-reaching in software licenses, publishers will do everything they can to cover their butts, even when some clauses stray from the legal and proper.

111. In avoiding liability and loss of sales, the publisher is encouraged by current law and precedent to keep secret that which the consumer needs most to make an in-

formed decision: accurate information about the software being considered for purchase. When a consumer does find a problem, the publisher can make it easy or hard to report the problem---and there is a trend that the difficulty of reporting and obtaining a fix is proportional to the size of the publisher.

112. Any solution that imposes implicit warranties on the publishers need to provide means of shielding responsible publishers from unreasonable litigation. Any such proposal should include a model process that the consumer *must* follow before having any recourse to the Courts for resolving problems.

113. Bug-free software is a desirable thing, but impossible to attain without significantly increasing the price of the software. Certification of software programmers and software engineers is no answer. The balance of correctness and market price and availability needs to be made in th e marketplace. Disclosure would allow the market to work.

114. I look forward to your public forum. I would suggest you consider mechanisms that would enable participation for those who are unwilling or unable to travel to Washington DC.

---30---