Appliances and Software: The Importance of the Buyer's Warranty and the Developer's Liability in Promoting the Use of Systematic Quality Assurance and Formal Methods

Daniel M. Berry

Computer Science Department
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

dberry@csg.uwaterloo.ca

Abstract

A vexing question is why systematic quality assurance and formal methods, despite all their advantages, are not used routinely in software development. Other engineering disciplines use their systematic quality assurance and formal methods in producing routinely reliable products. Perhaps the difference between the other engineerings and software engineers lie in the warranties producers must give for the products and the liabilities suffered by the producers for malfunctioning products. It is argued that software engineers would be more likely to use formal methods more routinely if they or their employers had to guarantee their software and could be sued for damages caused by their malfunctioning software.

1 Introduction

Formal methodologists continue to bemoan the failure of practicing software engineers to employ formal methods in their daily software development work [15, 30, 17, 8, 9, 12, 22, 23, 21, 11]. Early attempts by formal methodologists to convince software engineers to use formal methods focused on the benefits to software quality that would accrue if formal methods were to be used regularly in software development [20, 14, 11, 12]. Surely, once a software practitioner understood the benefits, he or she would start to use formal methods enthusiastically. To fail to do so would be illogical! Illogical or not, software engineers by and large, ignored formal methods, and when forced to use formal methods, they resist and sometimes actively subvert the application of formal methods to do meaningless busy work. When the project fails, partially due to the subversion, they gleefully blame the formal methods for the failure. Successful experimental applications of formal methods to real projects [21, 18, 10, 7, 13, 1, 2, 5, 31, 3, 19, 4] failed to convince most software engineers of the effectiveness of formal methods. The perception is that the project team got lucky or had other strengths going for it or that the project had special security or safety needs that could not be handled with ordinary methods, needs that are not found in normal software.

Formal methodologists began exploring ways to make formal methods more attractive. Most of these were technical solutions aimed at making the formal language, the method, the tools, etc. more palatable, more easily used, more powerful, more automatic, less ambitious, more realistic, more incremental, and even more fun [18, 10, 7, 13, 11, 1, 2, 5, 31, 3, 19, 4]. Each paper about one of these new approaches bemoans the lack of general use of formal methods, diagnoses the lack as the result of some particular problem in the use of formal methods, and offers a new approach that avoids, mitigates, or solves the identified problem. However, none of these has had any real effect on the extent to which formal methods are used. Others try educational, sociological, and managerial approaches [12, 23, 26, 6], and they too have failed to produce the desired bandwagon.

Even non-formal, but systematic methods for ensuring software quality suffer the same lack of use. Advocates of software inspection [16, 29] note the empirical evidence of inspection's effectiveness at finding faults before execution, an effectiveness exceeding that of traditional testing. They also note the reluctance of many organizations to use inspection and the varied, creative excuses given for not using inspection, even when they know its effectiveness.

Yet, when we look at other engineering disciplines, we see that they all have their systematic quality assurance (QA) and formal methods. Civil engineering has mathematical models of load and stress and these allow calculating, on the basis of only a paper design for a bridge, whether the proposed bridge will support the required weight, and then some. Electrical engineering has mathematical models of circuitry that allow calculating, on the basis of only a circuit diagram, whether the proposed circuit will behave as required, will not overheat, etc. We see that engineers in these disciplines routinely apply their formal methods with no complaints of being overburdened with useless work, no complaints of their creativity being stifled, and no complaints of having to use the dull, dreaded *mathematics* in another field.

The questions to ask are:

1. what makes the engineers in the other, more traditional engineerings, apply their systematic QA and formal methods routinely, and

2. can whatever does the trick in these other engineerings be used to get software engineers to use software engineering's systematic QA and formal methods in their daily software development?

This paper explores the quality of different engineering products, some electromechanical, some electronic, and some software, and notes key differences in the warranties offered with these products and the liabilities borne by their developers. Perhaps these differences account for the differences in the willingness of the various engineers to apply their engineerings' systematic QA and formal methods.

2 Recent Experience with Purchased Appliances and Software

In the last two years (as of November, 1999), I have bought* four appliances and four pieces of software. I am still using all the appliances but I have yet to get the two of the programs running, one of these is gathering dust on my shelf and the other has been returned for a refund. The other two programs are working. The four appliances are

1. Sharp Carousel Microwave Oven,

2. RCA Color Television,

3. Toshiba Video Cassette Recorder, and

4. Hoover Futura Vacuum Cleaner.

The four programs are

1. Adobe Illustrator 7.0,

2. Adobe Acrobat Exchange 3.0,

3. Microsoft Office '97, and

4. Languageforce Deluxe Universal Translator.

These eight personal experiences amount to a case study giving anecdotal evidence in support of a popular perception that consumer software is of considerably poorer quality than consumer appliances.

* Strictly speaking, one does not buy software; he or she licenses software. However, common usage is "to buy software".

Observe that all the software in this list is developed by their producers for sale to the mass market of consumers and is different from *bespoke* software developed by one producer under a specific negotiated contract for a specific client.

2.1 Appliances

I assembled the vacuum cleaner according to directions provided in the package and then plugged it in. It worked immediately and has given no problem. I would quibble with the design of the cord wrap-around knobs. It is impossible to turn the machine on when any of the wire is wrapped around the knobs since the on-off switch doubles as a knob. However, I have no real complaints about the machine or its brief instructions.

To get the microwave oven working, I had to set a timer according to directions provided in the package. It took less than 10 minutes to read the entire manual to see what features it had and to decide which ones I would use. I set the clock with no problem and the oven has worked fine ever since. Unfortunately, the oven has no battery back up. Consequently, whenever there is a power hiccup, the oven resets and I have to reset the timer to get the overn to work again. Apart from this minor nuisance, I have no complaints about the oven or its documentation.

The television required a set up. The procedure was well explained, step-by-step, in the user's manual, and the set worked immediately. The set-up procedure had as its main goal to allow the television set to find all the channels in the antenna, cable, or cable box connected to the set. In my current situation, I had cable, and the set found some 50 channels. Two years earlier, in another country, I had bought a television to use with a cable box. That television set was also an RCA. Since the cable box mapped all cable channels to television channel 3, the set-up was trivial, and the television worked immediately.

Among the appliances, the video cassette recorder offered the most trouble. The manual seemed quite clear in that I could read step-by-step procedures, and I could see what functions I wanted to use. What I did not know was that the manual neglected to mention that it was necessary to undo one set up if you want to change to a different set up. The set up has as its purpose to find all the available channels coming in from the antenna, cable, or cable box. It went smoothly according to what the manual appeared to say. The record and play worked immediately. When I tried to program my first timed record, I saw that I had overlooked an important step that is to be done before the set up and that I had done the wrong set up. I did the important step, and the correct set up, and then tried to program a timed record again. It still did not work. Fortunately, I had paid for a warranty that provided in-house service. I called and arranged for service. At first, they wanted me to bring the set in. I told them that I had paid for in-house service. They said that it is better if I bring it in. I said that if they did not honor the service agreement immediately, I would return the set for a refund and tell everyone at the University of Waterloo not to shop at the store to whose service department I was speaking. They sent a technician out who walked through the set up with me after telling me that I had to undo the first set up. It seems that without undoing the first set up, I was leaving elements of the old set up in the computer and it was creating an inconsistent program that caused the whole programmed record to freeze. After the technician left, the programmed record worked perfectly and I have had no complaints since then. I did offer as a project in my graduate requirements engineering seminar the job of rewriting the user's manual.

2.2 Software

Contrast these appliance experiences with my software experiences. The difference is sobering.

I was still in Israel when I bought the copy of Adobe Illustrator 7.0 for Windows from a local supplier. Normally, I like to read the manual for software before doing any installation both to see ahead of time where the problems might be and to learn what features I was likely to use. I verified that my computer had more than the minimum of all the resources required. I followed the installation procedure described in the manual supplied with the software, doing what the install program asked me to do and answering its questions about the features I wanted. In answering these, I tended to include a feature that I was not sure about just to be sure that I got a minimal working set of features. The installer reported that Illustrator was successfully installed. I tried to start up the program, but it froze during its start up phase, never getting beyond the product name announcement screen. I figured that I had too many other programs running and killed them all. I started up Illustrator

again and it froze again. Each time it froze, I could not get the computer and Windows unstuck and had to turn the machine off and then on without going through the proper shut down procedure. I called the local supplier and was told that this program and other Adobe products do not work on the Hebrew version of Windows or even the Enabler version that allowed switching between Hebrew and English. I had to use a pure American version of Windows. I was upset at the local supplier for not telling me this little detail when I bought the program.

I changed my operating system to pure American Windows. Since I did all my work in English, it was no real problem not to have Hebrew or the ability to switch to Hebrew. I could use a departmental machine in the very rare occasion in which I needed Hebrew Windows. Installation appeared to succeed. Illustrator, however, still froze when I tried to run it, even when no other programs were running. Figuring that maybe my PC was too small, I put the software away for my planned move to Canada, at which time I would get a larger PC.

Once settled at the University of Waterloo in Canada, I arranged for a large machine. Of course the Windows on it was purely American. I installed Illustrator on that machine. Illustrator still froze the machine as it started up. I ended up deinstalling the program only after copying the large set of POSTSCRIPT Type 1 fonts Illustrator 7.0 provides; I decided to install them for use with troff on my UNIX platform. By this time it was too late to return the software for a refund.

When the software had failed to work, I tried to get help from Adobe. I could not find in the list of reported problems at the web site any solutions that I had not tried. Adobe provided an 800 phone number to call customer service. However, from overseas, 800 numbers are not free. Also, I do not hear well on the phone. I could find no button that sends e-mail for help. So, I sent e-mail to support@adobe.com, webmaster@www.adobe.com, and postmaster@www.adobe.com, explaining that I could not find my problem solved at their web site and I cannot hear on the telephone. I got no reply. The program sits unused on my bookshelf. I did get a nice set of fonts out of it, and Illustrator costs less than the sum of the costs of the individual fonts, but I feel that I threw out money, and I have no operating Illustrator. In all fairness, it is not clear where the fault lies; the problem could be in the Windows operating system on which I tried to run Illustrator.

I bought a copy of Adobe Acrobat Exchange 3.0 for Windows. This software comes with a hard copy manual describing only installation. The general user's manual is a PDF document readable by the software once the software is installed. The Acrobat Exchange 3.0 package consists mainly of two programs, AcroExch and Distiller, which manipulate PDF files and create PDF files from POSTSCRIPT files, respectively. They installed with no problem. I was able to print a hard copy of the user's manual. I read it and decided what features I wanted to use and how. I have been using both to prepare slide shows of the troff-generated POSTSCRIPT of lecture slides. I use Distiller to convert the troff-generated POSTSCRIPT into PDF and then I use AcroExch to rotate and crop pages and to make hypertext links between items in the slides for faster navigation during a lecture. Given my past experience with an Adobe product, it was a pleasant surprise that this Adobe product installed and works so well. I wish that all mass-produced software worked as well.

Even as I write this paper, the system guru for my Sun workstation is not able to get the publically downloadable Adobe Acrobat Reader 4.0 working on my workstation. It freezes up at the start up window. Fortunately it does not freeze up the machine. A control-C to the invoking window kills the program. Again, I am unable to find my problem discussed in the trouble shooting section of Adobe's web pages, Adobe has shut down the user's forum, and they do not answer my e-mail cries for help even when I explain that I cannot use the telephone.

I normally do not use any program in Office '97. I use troff for formatting, vi for editing, troff and Acrobat Distiller for slide shows, and vi and awk for spread sheeting. However, people insist on sending me Word, PowerPoint, and Excel documents. So I bought Office '97. I bemoaned the lack of a real hard-copy manual; the manual provided with Office '97 consists only of a tutorial for each program in the collection. I installed the software. The installation went well, the only problem being a lack of good information about what features I really needed. The programs seem to work well on 95% of the Word, PowerPoint, and Excel documents I receive. However, occasionally I get a document which when printed leaves a lot of characters unprinted. I see each of these characters on the screen, but on the paper, there is only white space in place of the character. I

found a section of Help that described this problem. It said that the problem was that the fonts selected for the missing text is not resident in the printer and that the solution was to request downloading of all fonts. Unfortunately, when I went to the right menu to request full downloading, I found that the option was already in force. An examination of the POSTSCRIPT file generated by Word showed that definitions for all the selected fonts were included in the file and that the missing text was not even in the POSTSCRIPT file. No wonder this text did not print! Occasionally, I ended up printing an image of the Word window to get hard-copy output. The most annoying thing about Word, apart from its very deficient editing capability and its abominable typesetting with irregular spacing between words and lack of ligatures, is the fact that I am asked when I am exiting Word, if I want to update the changes to I made to PDFWRITE.DOT. Of course not, since I did not make any changes to it as far as I am concerned, having done nothing with PDF in the document. However, once I answered too quickly and clicked the "Yes" button. Word would not work properly after that. I had to reinstall Word from the beginning. It has happened several times, when I have accidentally hit the wrong button and I had to reinstall Word each time.

I get documents in languages other than English, e.g., in Hebrew, French, Spanish, Portuguese, and German. I can read these, but occasionally need a dictionary to fill in holes in my understanding. When I saw this new program, Languageforce's Universal Translator, that offered multiple language dictionary functionality, I decided to buy it. It offered a way to avoid the necessity to keep a half dozen dictionaries on hand and to avoid the tedium of flipping pages to find a word. At first, I was really happy about the software. It had a real user's manual. It consisted of a set of scenarios for set up and various different ways of using the software, including as a dictionary. The manual even showed pictures of the screen that the user is supposed to receive after various steps in the scenarios. However, this joy was short lived. As I was following the set up procedure, I saw that a number of the screen pictures were either wrong or out of date. In any case, the installation program reported a successful installation. Then I tried to run the program. It simply would not run. It gave no explanation. I sent e-mail to Languageforce's support group to the address given in the user's manual. I never got *any* reply. However, I know that someone received the e-mail, because to this day, I get announcements of exciting new enhancements from the sales department. Even a message from me telling them that they have some nerve offering enhancements to me when they have not answered my request for help has failed to get these messages turned off. Needless to say, I took the program back for a full refund.

2.3 Software Released Too Early

After these experiences, I started to wonder what can be done to improve the use of quality assurance methods in the development of consumer software. After all, methods and technology do exist to do a better job with software. However, they are not being used in the rush to get software out to the market. In this rush, it appears that software is being released before it is ready. Software is going out for sale to consumers before it is certain that it will run and with the documentation woefully inadequate and even incorrect. Moreover, the manufacturers seem unprepared and even unwilling to service their shoddy merchandise. It might even be that the merchandise is so shoddy that the service people are overwhelmed and the shoddy service is a direct result of this overload. On the other hand, appliances for sale generally work with no trouble and continue to work and when they need service, the manufacturers stand behind the product and do service the products in a reasonable time. Once serviced, the problems seem to be solved.

A major reason that software is released before it is ready is the pressure on the producer to be first on the market with the product. Whoever is first usually gets and keeps a vast majority of the market. The second to the market usually gets very little market and fails as a business unless its product is perceived as at least an order of magnitude better than that of the first.* Therefore, there is a high incentive to release software early. Moreover, since customers accept shoddy software merchandise, there is very little incentive to delay release to improve the product. Any solution to this problem will have to reverse the incentives.

---

* Of course, there are the exceptions that make the rule. For example Apple's Macintosh system beat Microsoft's Windows system to the market by several years, but Microsoft nearly drove Apple out of business.

3 Analysis of Differences Between Software and Appliance Productions

What are the differences between appliances and software that might account for this observed difference in quality?

Certainly software is more complex and has more states than do vacuum cleaners. However, a television and a video cassette recorder are systems of moderate complexity matching that of many programs. Indeed, these machines probably implement some of their functionality with a computer and software.

Certainly the environment on which software runs is more varied than that on which appliances run. Software must run on a variety of CPUs and operating systems and flavors thereof. Appliance environments are far simpler, consisting of an electrical outlet and the television signals coming through a cable wire or over the air. However, in my case, all my PCs had very standard configurations. I bought them strictly for use of the Office programs that were not available on the Sun workstation, on which I do all my real work. I left them in their original, delivered, presumably standard, configurations. I would have expected the software to have been tested for running on my configurations.

In my opinion, one key difference is the difference in the warranty that comes with appliances and with software. An appliance is forced by law in most locales in the U.S. and Canada to have a warranty of fitness for its purpose. That is, the product is guaranteed to function as what it is. If I buy a television set, the manufacturer guarantees that it functions as a television set and as a television set as understood by the man in the street. Mass-produced software, however, traditionally comes with a shrinkwrapped license that says that the manufacturer warrants almost nothing about the behavior of the software. The manufacturer does warrant the medium on which one buys the software, the diskettes or the CD ROM. In other words, the manufacturer refuses to guarantee that Illustrator program actually allows the user to draw pictures, that Word actually formats documents, that PowerPoint actually makes slide shows, and that Universal Translator actually translates. The software manufacturers refuse to make these guarantees, because they are not required to by law, as are appliance manufacturers, and the software customers let them get away with it. What manufacturers are not required to do, they do not do, and the customers suffer.

Another key difference is the difference in liability borne by the producers of appliances and software. Appliance manufacturers are liable for damages caused by correctly used or malfunctioning appliances. Software producers disclaim almost all liability in their shrinkwrapped licenses, accepting liability only up to the cost of the software. Thus, software developers do not have to be as careful with their mass-market products as appliance manufacturers do.

3.1 Warranties

This section examines some warranties supplied with the software products and appliances described in Section 2. Each warranty's text is quoted in a sans serif font, allowing the quotation to be distinguished from comments that interrupt the quotation just after the sentences they discuss.

3.1.1 Software Warranties

Adobe's and Microsoft's End User License Agreement (EULA) are almost identical. Therefore, only one is quoted here. Adobe's EULA says:

> **5. Limited Warranty**. Adobe warrants to you that the Software will perform substantially in accordance with the Documentation

Who gets to decide how large the deviation from the documentation-described behavior is allowed and the software is still considered to perform *substantially* in accordance with the documentation?

> for the ninety (90) day period following your receipt of the Software.

The warranty applies for only 90 days, as if the software decays and might start to perform differently after 90 days. It sounds like this warranty provision was written to get mostly computer-and-software-illiterate legislators off the producer's back and mimics the warranty given with products that can decay and go bad after a

period of use. Ninety days would seem a reasonable amount for the producer to stand behind such a decaying product. I am quite sure that some computer-and-software-illiterate people think that software decays.

> [missing details dealing with fonts that are translated to other formats; the warranty does not apply to these other formats.]
>
> To make a warranty claim, you must return the Software to the location where you obtained it along with a copy of your sales receipt within such ninety (90) day period. If the Software does not perform substantially in accordance with the Documentation, the entire and exclusive liability and remedy shall be limited to either, at Adobe's option, the replacement of the Software or the return of the license fee you paid for the Software.

The producer gets to choose the remedy, not the customer. Moreover, The producer is permitted to replace the software, as if a different copy of the software will behave differently. Here again, it sounds like this provision is written to get naive legislators off the producer's back with something that makes it appear that the producer is really trying to get to the user something that works. I hope that the customer can force the supplier to adopt the money-back remedy when it is clear that another copy of the same program is going to perform exactly the same as the non-conforming copy.

> ADOBE AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE OR DOCUMENTATION. THE FORGOING STATES THE SOLE AND EXCLUSIVE REMEDIES FOR ADOBE'S OR ITS SUPPLIER'S BREACH OF WARRANTY. EXCEPT FOR THE FORGOING LIMITED WARRANTY, ADOBE AND ITS SUPPLIERS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AS TO NONINFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE.

In any case, the producer denies, and shouts this denial of, any warranty for what really matters, in particular for merchantability and for fitness for *any* purpose, including for the very purpose of the software. Note that *all* non-software products are warranted by law in the U.S. and Canada for merchantability and for fitness for their stated and implied purposes.

> Some states or jurisdictions do not allow the exclusion of implied warranties or limitations on how long an implied warranty may last, so the above limitations may not apply to you. To the extent permissible, any implied warranties are limited to ninety (90) days. This warranty gives you specific legal rights. You may have other rights which vary from state to state or jurisdiction to jurisdiction. For further warranty information, please contact Adobe's Customer Support Department.

The reality is that very few places do not allow exclusion of implied warranty for software products.

Elsewhere in the license, it is written:

> This package contains software (''Software'') and related explanatory written materials (''Documentation'').

Adobe Illustrator 7.0 and Microsoft Office '97 come with reasonably good, descriptive manuals describing some typical scenarios the users might wish to do. Therefore, it might appear that the software is being warranted to behave as the manual says it does. However, the warranty specifies only *substantial* compliance with the written documentation, not complete compliance. Who decides how much compliance is substantial enough? In addition, it might be that the software can do *all* the scenarios that are described in the manual, as these were the test cases. Certainly the developer had to get these examples running to get the pictures of the screen that are shown in the manual. However, the software does nothing more general, because the manual describes *all* the test cases. In other words, the documentation means only what it says and not what the average reader generalizes it to say.

The only written material I find in many packages these days is a manual describing only installation. Given the typical EULA as described above, perhaps the producer is warranting only that the installation, and not necessarily the program, will perform substantially, but not necessarily completely, in accordance with the documentation provided. Of course, there is the help system providing documentation, but if the software does not run, and the help system does not work, does that mean that the software is effectively not documented or that if the user cannot get to the documentation, any behavior is allowed for the software because it is undefined in the documentation?

Clearly, the warranty accompanying software is next to useless except for getting one's money back if the software does not work.

3.1.2 Appliance Warranties

The Hoover vacuum cleaner comes with a warranty that says:

> **Full One Year Warranty** (Domestic Use)
>
> Your HOOVER® appliance is warranted in normal household use, in accordance with the Owner's Manual against original defects in material and workmanship

The warranty covers all defects in what comes from the manufacturer. There is no concept of performing only substantially as a vacuum cleaner. The appliance must perform completely as a vacuum cleaner from the beginning.

> for a period of one full year from date of purchase. This warranty provides, at no cost to you, all labor and parts to place this appliance in correct operating condition during the warranted period.

Hoover is saying that it *will* make the appliance work, by replacing whatever is necessary and doing whatever work is necessary to get it running. Presumably, Hoover could replace all parts together as a unit, that is, provide a whole new vacuum cleaner.

> This warranty applies when the appliance is purchased in the United States including its territories and possessions, or in Canada, or from a U. S. Military Exchange. Appliances purchased elsewhere are covered by a limited one year warranty that covers the cost of parts only.

While a customer might not be covered for labor costs outside the U.S. and Canada, the parts and presumably a replacement are covered. Also, if Hoover has built the vacuum cleaner well enough to be sold in the U.S. and Canada, the vacuum cleaner will in all probability not need any such repair. Thus, the lack of a full warranty is not disturbing.

> This warranty does not apply if the appliance is used in a commercial or rental application.

Hoover is covering only normal household use, not heavy-duty use.

> Warranty service can only [sic] be obtained by presenting the appliance to one of the following authorized warranty service outlets. Proof of purchase will be required before service is rendered.
>
> 1. Hoover Factory Service Centers.
>
> 2. Hoover Authorized Warranty Service Dealers (Depots).
>
> [details on servicing omitted]
>
> This warranty does not cover pick up delivery, or house calls; however, if you mail your appliance to a Hoover Factory Service Center for warranty service, transportation will be paid one way.

> While this warranty gives you specific legal rights, you may also have other rights which vary from state to state.

The contrast is striking. For the vacuum cleaner, I got a full, unlimited warranty, and I did not need it. Moreover, I still have a fully functioning vacuum cleaner. For Illustrator, I got a limited warranty, and needed a full warranty, as the limited warranty did not provide a useful remedy. A new copy would behave as the one I had and my money back would leave me with no Illustrator.

The Sharp microwave oven comes with a warranty that says:

> SHARP LIMITED WARRANTY
>
> Consumer Electronics Products
> **Congratulations on your purchase!**
>
> Sharp Electronics of Canada Ltd. (hereinafter called ‘‘Sharp’’) gives the following express warranty to the first consumer purchaser for this Sharp brand product, when shipped in its original container and sold or distributed in Canada by Sharp or by an Authorized Sharp Dealer:
>
> Sharp warrants that this product is free, under normal use and maintenance, from any defects in material and workmanship. If any such defects should be found in this product within the applicable warranty period, Sharp shall, at it’s [sic] option, repair or replace the product as specified herein.

The product must perform completely as a microwave oven, and there is no concept of behaving only substantially like a microwave oven. Also, Sharp is guaranteeing that the customer will have a microwave oven somehow, and that he or she will never have to settle for money back and no microwave oven.

> This warranty shall not apply to; [sic]
>
> (a) Any defects caused or repairs required as a result of abusive operation, negligence, accident [sic] improper installation or inappropriate use as outlined in the owner’s manual;
>
> (b) Any Sharp product tampered with, modified, adjusted or repaired by any party other than Sharp, Sharp’s Authorized Service Centres or Sharp’s Authorized Servicing Dealers;
>
> (c) Damage caused or repairs required as a result of the use with items not specified or approved by Sharp, including but not limited to, head cleaning tapes and chemical cleaning agents.
>
> (d) Any replacement of accessories, glassware, consumable or peripheral items required through normal use of the product, such as earphones, remote controls, AC adaptors, batteries, temperature probe, stylus, trays, filters, etc.
>
> (e) Any cosmetic damage to the surface or exterior that has been defaced or caused by normal wear and tear.
>
> (f) Any damage caused by external or environmental conditions such as liquid spillage or power line voltage, etc.
>
> (g) Any product received without appropriate model and serial number identification and/or CSR markings.
>
> (h) Any consumer products used for rental or commercial purposes.

Sharp is careful to exclude causes of damage that are not its fault or are the fault of the customer or normal

wear and tear. Notice that nothing for which the customer would really expect Sharp to be responsible has been excluded.

> Should this Sharp product fail to operate during the warranty period, service may be obtained upon delivery of the Sharp product together with proof of purchase to an Authorized Sharp Service Center or an Authorized Sharp Servicing Dealer.
>
> [details on servicing omitted]
>
> This warranty constitutes the entire express warranty granted by Sharp and no other dealer, service center or their agent or employee is authorized to extend, enlarge or transfer this warranty on behalf of Sharp.

The period of the microwave warranty is defined elsewhere in a table.

> **WARRANTY PERIODS**
>
> ...
> Microwave Oven 2 years (magnetron 3 additional years part warranty only)
> ...

The limited warranty period is a recognition that an appliance decays and that it cannot operate like new forever.

Basically, for appliances, manufacturers warrant that there are no defects, that the appliance behaves as it is specified, and that they will make the appliance run if the customer finds a defect within the warranty period.

It is my belief that if laws were changed forcing software manufacturers to guarantee fitness for purpose or functionality, their procedures would change so that software is released only after the same kind of quality control that appliances are subjected to.

3.2 Liability

This section examines the liabilities borne by the producers of the software products and appliances described in Section 2. Appliance manufacturers are held liable for damages caused by their appliances, e.g., if an appliance blows up, catches fire, etc. Furthermore, if it can be shown that the manufacturer failed to apply accepted quality control procedures for the engineering disciplines involved in the manufacture, the manufacturer can be judged willfully negligent and can be assessed punitive damages. Consequently, an appliance manufacturer applies whatever methods are available for predicting behavior and assuring quality of its products, including testing and modeling. It also arranges for independent verification and validation (IV&V), for example, by the Underwriters' Laboratory, as part of the process of determining the cost of its liability insurance.

The Hoover vacuum cleaner warranty has *no* limitation of liability whatsoever. The Sharp microwave oven warranty has a limitation of liability.

> To the extent the law permits, Sharp disclaims any and all liability for direct or indirect damages or losses or for any incidental, special or consequential damages or loss of profits resulting from a defect in material or workmanship relating to the product, including damages from loss of time or use of this Sharp product. Correction of defects, in the manner and period of time described herein, constitute complete fulfillment of all obligations and responsibilities of Sharp to the purchaser with respect to the product and shall constitute full satisfaction of all claims, whether based on contract, negligence, strict liability or otherwise.

In many places, the law does not permit Sharp to disclaim all liability, particularly of damages or loss caused by a functioning or malfunctioning product. In other words, if a correctly used microwave oven explodes, Sharp is liable for the damages and loss caused by the explosion.

Software developers suffer no such liability. There are few laws specifying their liability. Furthermore, they usually write into their shrinkwrap, mass market licenses a disclaimer for liability for damages beyond the cost of the software itself. Adobe's EULA shouts out a very strong limitation on liability; Microsoft's EULA has a very similar shouted limitation on liability.

> **6. Limitation of Liability**. IN NO EVENT WILL ADOBE OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL, OR SPECIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, EVEN IF ADOBE REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY THIRD PARTY. Some states or jurisdictions do not allow the exclusion or limitation of incidental, consequential or special damages, so the above limitation may not apply to you.

In most jurisdictions, the producer has no liability whatsoever for any damages caused by the software's inability to do its function or for any damage done by malfunctioning software. The consumers accept the useless warranty and the limitation of liability. More than that, consumers accept the poor quality software and keep paying for upgrades, which are often little more than corrections of flaws in a product that they already paid for.

It is my belief that if laws were changed forcing liability on software developers, their procedures would change to use all available methods for assuring quality, including inspections, testing, IV&V, and even formal methods. They will do anything to stave off a claim of willful negligence in the event of damages from the execution of their software.

3.3 Mass-Market vs. Bespoke Software and Negotiating Power

Note again that I am talking about consumer software developed at a producer's own expense and risk for the mass market. For bespoke software, especially for systems with high reliability and safety concerns such as in aircraft, automobiles, telecommunications, and process control, the producer warrants the product and is subject to liability as a result of the contract negotiated face to face between the client and producer. Here, the client has the power to force the producer to warrant the product and accept liability, because the client can always go to another producer. In the consumer market, in which there is no face-to-face negotiation of a contract, a contract warranting nothing and limiting the producer's liability is foisted on the consumer through the shrink-wrap mechanism. Because for a given function, there is often only one product that runs on a customer's system or that all those interacting with the customer can use, the customer is forced to accept this product and its license; the producers have the power to force consumers to accept an agreement that strongly favors the producers. Perhaps this imbalance of power is the reason that consumers accept the poor quality of software and the unfavorable terms of the shrinkwrap consumer software license agreement.

3.4 Wrap Up

This discussion of warranty and liability is best concluded by examining a letter written by Justus Pendleton of Somerville, MA to *IEEE Computer*, January 1999 [28]. The letter was written ostensibly to to point out that it is no easier to verify fitness for use of normal commercial shrinkwrapped software than of open-source software. He was replying to an earlier letter to the same journal that claimed that open-source software offers full protection only if the user has the resources to conduct a full inspection. Pendleton wrote:

> There is a fitness-for-use disclaimer in virtually all software that usually says something to the effect ''this [information, computer program] is being provided with all faults, and the entire risk as to satisfactory quality, performance, accuracy, and effort is with the user.'' The buyer of shrinkware as to either take the vendor's word that the software is fit to use or subject it to black-box testing (the results of which cannot be published without the vendor's explicit and prior permission), which is arguably more difficult that a thorough inspection of source code. ... These are the same vendors that tell us the next version, which is due out next month sometime, will fix all the problems we are having.

4 Quality Assurance Methods and Warranties and Liabilities

In a number of engineering disciplines, there are systematic and sometimes formal procedures for verification and validation that are to be followed while the product is in design stage. Electrical engineers routinely apply mathematical models of electronics to determine if their designs will function correctly and will meet safety requirements. Civil engineers and architects routinely apply mathematical models of structures to verify that the structures they are designing will support the load to which they will be subjected and that they will withstand the environmental forces that may push on them. The reason that these engineers routinely apply their QA procedures is that if they do not and the product does not work as it is supposed to, their employers may be inundated by customer complaints, may suffer massive returns with refunds, and may, in the worst case, be sued for damages. The employers may then take disciplinary and, in some cases, job action against the engineers responsible for the malfunctioning product. Also, if these engineers do not apply their QA procedures and the product causes damages, the failure to apply the QA procedures in the construction of the product may subject the manufacturer to a negligence claim and punitive damages beyond the just the base cost of the damages.

In these engineering disciplines, the manufacturers establish procedures to be followed during design, development, and manufacturing. These procedures include a variety of tests, ranging from inspection of documents, through actual usage of prototypes of and samples of the developed products, to exercising mathematical models. The manufacturers require employees to follow these procedures and to document that they have followed the procedures. The documentation may be subpoenaed in a damages lawsuit. Failure to follow these procedures subjects the offending employee to disciplinary action and, in some cases, job termination. These procedures and penalties for failure to follow the procedures is the manufacturer's best defense against a negligence claim.

The professional requirements for a medical doctor or physician are instructive. A physician is held to *the standard of care* in his or her community. Failure to provide at least the current standard of care may subject the physician to a negligence complaint and to malpractice action. The definition of this standard varies and depends on

1. what is taught at medical school,

2. the results of recent medical research, and

3. what the physicians in the community regularly do, given the resources available.

The community standard of care is determined case-by-case in malpractice cases from the testimony of expert witnesses, usually other physicians*.

In medicine, the standard of care for a community is a baseline and may not be all that close to the state of the medical art. It consists of what the doctors in the community consider to have been demonstrated as effective treatment, modulo the facilities and resources available to carry it out. While it is not required for a physician to apply the latest treatments, which may be only experimental, it is not an acceptable defense in a malpractice suit to say that the applied out-of-date treatment is what the physician learned in medical school. The physician is required to keep up to date and learn new treatments that have been demonstrated effective against diseases in his or her specialty. The standard of care for a community evolves continually with new treatments established by research as effective.

Anyone with a duty to be careful in a treatment is considered negligent and is liable for damages if he or she has not applied the accepted standard of care, the care causes damages, and there was no independent, intervening cause of the damages. The standard of care is higher for a relevant professional than for others. For a non-physician, the standard of care for medical treatment is what the reasonable person-in-the-street would do in the circumstances. For the professional physician, not to apply the community's standard of care for

* These expert witness doctors are paid by one side or another in the case. Thus, it is not hard to find doctors who, for a fee, will testify that another doctor's care was not up to the standard of care for the community. It is also not hard to find other doctors who, for a fee, will testify to the exact opposite. In the end the judge or the jury have to decide if the care was up to the standard.

physicians is considered malpractice.

In medicine, the standard of care does not require using not-yet-widely used treatments and, in fact, may require *not* using them, especially if they are as yet unproved. However, in other areas, one might be expected to use a new technology even it is not yet widely used. In such a case, the standard of care drives adoption of new techniques. There was a famous case from the 1920s or 1930s in which the operators of a tugboat, the *T. J. Hooper*, were held liable for the boat's sinking in a storm because there was no radio on board with which to listen to weather reports. The operators were held liable even though, at the time, most boats did not have radios. This case spurred the adoption of radios as standard equipment on board boats.

5 Loss of Exclusion of Warranty and Liability

What will happen if warranty and liability limitations for software producers are brought in line with those of other manufactured products, and software producers become as accountable for product quality as other manufacturers. Please allow me to speculate. I believe that the software producers will have to start applying community and professionally accepted standards of care, both to produce more reliable software and to serve as a defense against liability should products cause damages despite the care. They will need to establish procedures that must be followed during specification, design, development, and deployment. These procedures will include a variety of tests, ranging from inspection of documents, through uses of prototypes and production code in runs against test data, to formal model checking and verification. They will require the software engineers working for them to follow these procedures and to document that they have done so. Finally, they will provide for disciplinary action and even job termination for failure to follow these procedures.

Similar procedures are established by many software producers in order to obtain CMM or ISO 9000 certification [27]. Because of the artificial imposition of the procedures that have no observable direct positive impact and seem to mire a project in process, many employees doubt the effectiveness of the procedures and may even subvert their imposition. Moreover, many software manufacturers, with no chance for or interest in government contracts, simply do not bother with certification at all. Loss of warranty and liability limitation will force all software producers to adopt systematic QA procedures, possibly those suggested by CMM or ISO. In addition, the effect of failure to follow the procedures will be felt more directly and swiftly, in the form of job and legal actions, thus encouraging better compliance by employees.

Indeed, here we may have the solution to the inexorable pressure of the rush to market that give a high incentive to premature release of software. The loss of warranty and liability limitation changes the economics of early release. Early release may increase exposure to warranty and liability claims, which can be very costly. Thus, there would be a higher incentive to slowing down to release higher quality SW. It would become a tradeoff of exposure to warranty and liability claims versus loss of market.

These procedures will become the accepted standard of care against which all work will be compared, particularly if the work has led to a substandard product or one which has caused damages. Of course, in software development, the standard of care will vary depending on the product. The more critical the product, the higher the standard of care for its development. For software driving a system on which lives depend, the standard of care will be considerably higher than for software driving a recreational game. For some life-critical systems, the standard of care would likely include formal methods such as model checking and possibly even formal verification. For a program to play solitaire, the standard of care would be considerably lower, probably including only inspection and basic testing.

The standard of care for software development will depend on

1. what is taught in software engineering degree programs,
2. the results of recent software engineering research, and
3. what software engineers in the community regularly do, given the resources available and the domain of the software.

Quite likely software engineering will be judged to be a field in which new techniques, not yet widely adopted, will be expected to be used when the situation warrants it.

Where are formal methods in all of this? Formal methods are taught in software engineering degree programs, formal methods are explored in software engineering research, formal methods are used in the most critical projects. Finally, formal methods, while not widely adopted, have been shown to be of benefit in development of critical software. Therefore, it seems clear that formal methods will be part of the standard of care for some software developments and that the exposure of software producers to liability will drive them to adopt formal methods for the development of critical software, and possibly, of some less than critical software.

6 The Software Engineering Profession

There is a move to make software engineering a full-fledged engineering profession [24, 25]. Just as the practitioners of other professions, engineering, medicine, or others, are expected to apply the profession's standard of care in their work or face malpractice action, so will the software engineer be expected to apply software engineering's standard of care. After all, an engineer's responsibilities include making sure products he or she produces are fit for use, contrary to what current software warranties claim. If this standard of care includes formal methods, software engineers would be compelled to apply formal methods in appropriate circumstances. The software engineer who does not apply this standard of care would find himself out of a job or facing legal malpractice action.

Normally, the company that produces a product is liable for the product and the individual employees are not. However, an individual licensed engineer assumes liability for those products whose fitness he or she has guaranteed with his or her signature. This liability borne by the licensed engineer who signs off on a product is scary and probably accounts for the resistance of many current software developers, who call themselves "software engineers", to licensing of software engineers under standard engineering charters. The engineer can lose his or her profession and face severe legal consequences if a product he or she develops and guarantees fails.

7 Conclusion

Once product warranty and liability applies to software products, a producer of software will be compelled not to release software until it can guarantee that it behaves as it is supposed to, for fear of consequences such as a flood or complaints, having to refund lots of buyers, having to recall the product, having to stand behind a faulty product, and possibly even paying damages if the product causes damage as a result of not behaving as it is supposed to. I believe that to meet the required level of quality, software producers will be force to establish systematic QA procedures. They will be forced to put teeth into the procedures in order to force employees, the software engineers, to comply with these procedures. The software engineers will face disciplinary action and possible termination for failure to follow the procedures. The procedures will likely include formal methods for certain classes of critical systems.

Perhaps one day, commercially available software will be as reliable as commercially available appliances. While appliances are by no means pefect, they are a whole lot more reliable than software. I could live with software being as good as my microwave oven!

8 Added in Proof

Just as I was preparing this paper for submission, an extremely relevant article authored by Joseph Menn, a *Times* staff writer, appeared in the newspapers on 4 February 2000. It also appeared in the WWW at

http://www.latimes.com/
business/updates/lat_rights000204.htm.

The article at the web site is titled "Software Makers Aim to Dilute Consumer Rights" with a subtitle of "Technology: Companies push legislation at state level that would dramatically alter contract law in their favor."

Microsoft Corp. and other powerful software companies are quietly pushing state

> legislation across the nation that would dramatically reduce consumer rights for individuals and businesses who buy or lease software and database information.
>
> The push comes as software companies are beefing up their lobbying effort to pass favorable laws while their industry is at peak popularity among politicians who want to keep their local economies booming, consumer groups say.
>
> "[This] is an example of newly powerful software giants using the promise of high-tech jobs to push through legislation that restricts consumer and business-customer rights," said James Tierney, former Maine attorney general, who opposes the effort.
>
> The tech bills spring from a proposal with an arcane name, the Uniform Computer Information Transactions Act (UCITA). Should states pass this legislation, the impact on consumers would be dramatic:
>
> ....
>
> But in dozens of ways, large and small, the bills tip the balance of power toward software companies, according to law professors, consumer groups, more than 20 state attorneys general and some corporate software buyers that are beginning to organize an opposition to the UCITA campaign.
>
> If these UCITA-sponsored bills pass, "it will dramatically change the law," said Herschel Elkins, head of the California attorney general's consumer department. He said the legislation would put buyers into a legal corner with little way out. "It's pay first, find out what you bought later," he said. "The refund right disappears when you click twice on 'I agree.' "
>
> ...
>
> "It's very difficult to understand," [Temple University law professor Amy] Boss said of the bill. Under the legislation, customers who install software in their computers have already lost some of their basic rights, she said. The tech bill "gives the consumer no way to disagree with the terms," she said.
>
> Microsoft's [Rick] Miller declined to discuss some of the complex bill's provisions. Other supporters of the legislation said its critics misunderstand the effect of the measure.

It can be even worse than I thought. In Section 3.3, I observed that the public seems to accept software producers' claims of no warranty and no liability. However, it is not clear that the courts would accept these claims if enough members of the public were to sue. The courts would likely apply standards for normal consumer products, if for no other reason than judges and juries really do not understand software. UCITA, however, would override any such court decision by explicitly legislating the provisions of no warranty and no liability found in most shrink-wrapped licenses.

Acknowledgments

I thank Connie Heitmeyer and Dino Mandrioli for bibliographical references and comments and suggestions that led to improvements in the paper. I thank David Kay for an e-mail discussion about standards of care for medical doctors. Finally, I thank Egon Boerger, Martin Feather, and three anonymous reviewers for their sharp criticisms of a previous version of this paper. These criticisms led to a major revision of the paper. I was supported in parts by a University of Waterloo Startup Grant and by NSERC grant NSERC-RGPIN227055-00.

References

[1] *IEEE Computer* **23**(9) (September 1990), Special Issue.

[2] *IEEE Software* **7**(5) (September 1990), Special Issue.

[3] *Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques*, IEEE Computer Society, Boca Raton (April 1995).

[4] *Proceedings of the Fourth NASA Langley Formal Methods Workshop*, NASA Conference Publication 3356, Hampton, Virginia (September 1997).

[5] *Journal Systems and Software* **40**(3) (March 1998), Special Issue.

[6] Berry, D.M., "Formal Methods, the Very Idea, Some Thoughts on Why They Work When They Work," *Electronic Notes in Theoretical Computer Science* **25**, Elsevier (1999), http://www.elsevier.nl/locate/entcs/volume25.html.

[7] Bharadwaj, R. and Heitmeyer, C., "Applying the SCR Requirements Method to a Simple Autopilot," in *Proceedings of the Fourth NASA Langley Formal Methods Workshop*, NASA Conference Publication 3356, Hampton, Virginia (September 1997).

[8] Bowen, J.P. and Hinchey, M.G., "Seven More Myths of Formal Methods," Technical Report PRG-TR-7-94, Oxford University Computing Laboratory (1994).

[9] Bowen, J.P. and Hinchey, M.G., "Ten Commandments of Formal Methods," Technical Report, Oxford University Computing Laboratory and University of Cambridge Computer Laboratory (1995).

[10] Chan, W., Anderson, R., Beame, P., Burns, S., Modugno, F., Notkin, D., and Reese, J.D., "Model Checking Large Software Specifications," Technical Report, Computer Science Department, University of Washington, Seattle, WA (September 1997).

[11] Ciapessoni, E., Coen-Porsini, A., Crivelli, E., Mandrioli, D., Mirandola, P., and Morzenti, A., "From Formal Models to Formally-Based Methods: An Industrial Experience," *ACM Transactions on Software Engineering and Methodologies* **8**(1), p.79–113 (January 1999).

[12] Cleland, G. and MacKenzie, D., "Inhibiting Factors, Market Structure and the Industrial Uptake of Formal Methods," pp. 46–60 in *Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques*, IEEE Computer Society, Boca Raton, FL (April 1995).

[13] Easterbrook, S. and Callahan, J., "Formal Methods for Verification and Validation of Partial Specifications: A Case Study," *Journal Systems and Software* **40**(3), p.199–210 (March 1998).

[14] Gerhart, S.L., "Program Verification in the 1980s: Problems, Perspectives, and Opportunities," ISI/RR-78-71, USC Information Sciences Institute, Marina Del Rey, CA (August 1978).

[15] Gerhart, S.L., "Applications of Formal Methods," *IEEE Software* **7**(5), p.6–10 (September 1990).

[16] Gilb, T. and Graham, D., *Software Inspection*, Addison Wesley, Wokingham, UK (1993).

[17] Hall, A., "Seven Myths of Formal Methods," *IEEE Software* **7**(5), p.11–19 (September 1990).

[18] Heimdahl, M.P.E. and Leveson, N.G., "Completeness and Consistency in Heirarchical State-Based Requirements," *IEEE Transactions on Software Engineering* **SE-22**(6), p.363–377 (June 1996).

[19] Hinchey, M.G. and Liu, S., *Proceedings of First International Conference on Formal Engineering Methods*, IEEE Computer Society, Hiroshima, Japan (November 1997).

[20] Hoare, C.A.R., "An Axiomatic Basis for Computer Programming," *Communications of the ACM* **12**(10), p.576–580,585 (October 1969).

[21] Jackson, M., "Formal Methods and Traditional Engineering," *Journal of Systems and Software* **40**(3), p.191–194 (March 1998).

[22] Jones, C.B., "Whither Formal Methods: A Plea to Investigate New Applications," pp. 5 in *Proceedings of First International Conference on Formal Engineering Methods*, ed. M.G. Hinchey and S. Liu, IEEE Computer Society, Hiroshima, Japan (November 1997), Invited Lecture.

[23] Knight, J.C., DeJong, C.L., Gibble, M.S., and Nakano, L.G., "Why Are Formal Methods Not Used More Widely," pp. 1–12 in *Fourth NASA Langley Formal Methods Workshop*, NASA Conference Publication 3356, Hampton, Virginia (September 1997).

[24] McConnell, S. and Tripp, L., “Professional Software Engineering: Fact or Fiction?,” *IEEE Software* **16**(6) (November/December 1999).

[25] Parnas, D.L., “Software Engineering: An Unconsummated Marriage,” *Communications of the ACM* **40**(9), p.128 (September 1997).

[26] Parnas, D.L., ““Formal Methods” Technology Transfer Will Fail,” *Journal Systems and Software* **40**(3), p.195–198 (March 1998).

[27] Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, C.V., “Key Practices of the Capability Maturity Model,” Technical Report, CMU/SEI-93-TR-25, Software Engineering Institute (February 1993).

[28] Pendleton, J., “Shrinkwrap Is No Safer,” *IEEE Computer* **32**(1), p.9 (January 1999).

[29] Pressman, R., “Software According to Niccolò Machiavelli,” *IEEE Software* **12**(1), p.101–102 (January 1995).

[30] Wing, J.M., “A Specifier’s Introduction to Formal Methods,” *IEEE Computer* **23**(9), p.8–24 (September 1990).

[31] Woodcock, J.C.P. and Larson, P.G., *FME’93 Industrial-Strength Formal Methods*, Springer, LNCS 670 (1993).