LIABILITY FOR DEFECTIVE DOCUMENTATION

Published in *Software QA Quarterly,*

In October, 1985, W.H. Daughtrey bought a diamond bracelet as a Christmas gift for his wife from Sidney Ashe, a jeweler.[1] He paid $15,000. After Daughtrey agreed to buy the bracelet, Ashe filled out an appraisal form and put it in the box with the bracelet. The appraisal said that the diamonds were of *v.v.s. quality* (a high grade). Daughtrey didn't see the appraisal until later, probably not until the box was opened at Christmas.

In 1989, Daughtrey discovered that the diamonds were not of v.v.s. quality. Ashe offered a refund. Daughtrey refused, and demanded that the diamonds in the bracelet be replaced with diamonds that were of v.v.s. quality. Ashe refused. Daughtrey sued. He said that the statement that the diamonds were of v.v.s. quality was a description of the goods by the seller. According to the Uniform Commercial Code,

> 2-313(b) any description of the goods which is made part of the basis of the bargain creates an express warranty that the goods shall conform to the description.

Therefore, Daughtrey said, Ashe created a warranty that the diamonds were of v.v.s. quality, and breached it by selling a bracelet whose diamonds were of a lower grade. Ashe argued that this claim couldn't have been a warranty because he never called it a warranty and Daughtrey didn't read the claim until long after the sale. How could this description be part of the "basis of the bargain"?

Ashe won -- in the trial court. But the Supreme Court of Virginia overruled the trial court. Quoting the Official Comments to the Uniform Commercial Code, the Court said:

> The whole purpose of the law of warranty is to determine what it is that the seller has in essence agreed to sell.[2]

and

> The precise time when words of description or affirmation are made . . . is not material. The sole question is whether the language is fairly to be regarded as part of the contract.[3]

The Court concluded that Ashe had agreed to sell v.v.s. quality diamonds, and therefore that he had breached the sales contract by selling inferior diamonds.

So what does this have to do with computer software?

When a customer buys a computer program at a store, there is probably a manual in the box. Just as there was an appraisal in the bracelet box. And the manual makes specific descriptive statements about the program. Just as the appraisal made a specific descriptive statement about the bracelet. You might not read the manual until long after you've bought the program. Just as Daughtrey didn't know anything about the appraisal until after the purchase. A judge in Virginia would probably treat statements in the manual as warranties.[4] The seller of the program would be liable for breach of warranty if the software didn't do what the manual stated.

Customer Care, Inc. publishes a survey every year, *the Customer Care Survey: Service & Support Practices in the Software Industry*.[5] In the 1994 survey, page V-29, 50% of the responding companies said that they don't put their manuals through Quality Assurance. I'm writing this article to tell you that if you're not testing your documentation, you're making a big mistake.

If your manual or your help or your packaging say false things about the program, your company is risking lawsuits for breach of warranty, for deceptive trade practices, and for misrepresentation.[6]

Lawyers' Tricks

Your company's lawyers look for ways to protect your company from lawsuits. There are some things that they can do to reduce your company's exposure. But if your company is at fault, they can't be expected to win every time.

For example, your lawyer can argue that your manual can't be interpreted as a warranty unless the customer read it and relied on its statements when making the decision to buy. That argument will work in a few States. Variations on it will work in most States.[7] But this won't work in other States, such as Virginia. And in most States, the argument will be very difficult if the customer did flip through the manual before buying the program.[8] What will work, in every State, is making sure that the manual describes the program accurately. Your lawyer can't do that. But you can.

The shrink-wrapped warranty disclaimer is another trick we're all familiar with. This is the piece of paper that some software companies put inside the box, that says that the program is sold "As Is" with no warranty. I'll probably write more about this in a later column[9] but the conclusion is a simple one. These disclaimers may convince people who don't know better that they have no rights, but they are not in accordance with the Uniform Commercial Code. American courts throw them out.[10] These disclaimers also fare badly outside of the United States.[11] Several software publishers now take a different approach. They give purchasers an honest warranty that the software will perform substantially in accordance with the packaging and documentation. Other software publishers provide a satisfaction guarantee and let dissatisfied purchasers return the product for a full refund within 30 days, 90 days or even a year.

In short, rather than relying on your lawyers to bail you out, you are better off making sure that your manual, help, and packaging match the program.

Bad Documentation and the Cost of Quality

The cost of quality associated with a product includes the cost of investments that your company makes in developing a high quality product, and the expenses it suffers in dealing with the failings of the product.[12] Examples of quality-related costs include money spent on code inspections, design reviews, defensive programming, fixing bugs, and answering customer complaints.

One of the key goals of quality engineering is reduction of the total cost of quality of a product. For example, it is often cost-effective to spend more money preventing bugs, thereby reducing the amount needed to support the product in the field. Successful problem prevention can reduce the total cost of quality.

It costs money to thoroughly test a manual. I'll explain what I do, to thoroughly test a manual, later. Here, the issue is money. My rule of thumb is that it takes about 15 tester-minutes per page of the manual. I haven't seen thorough tests go much faster than this. In one extreme case involving a particularly buggy program and a problematic manual, I spent about an hour per page on the manual.

You might test two or three drafts of a manual. It usually costs less to test a page for the second time, but I don't budget for less than 7.5 minutes per page.

Suppose that you spend a total of 100 tester-hours testing three drafts of a 200-page manual. Add 20% for administrative overhead (meetings, status reports, etc.) and suppose that your overhead+benefits+salary-weighted cost is $50 per hour per tester. Testing the manual cost $6000 over three tester-weeks. This is a significant quality-related expenditure. Some managers automatically say "too much, don't do it" in the face of this proposed expense.

To decide whether it really costs too much to test the manual, do some quality engineering. What does your company save by making this three week, $6000 investment? Here is part of that analysis, for a software publishing company. When you think carefully about your own company, you'll probably be able to add several additional internal benefits that a good manual provides in your company, and several additional risks that an error-prone manual poses for your company.

Within Product Development (including the Testing Group), the manual might serve several functions, such as the following:

- ***It's a test plan.*** It takes you on a tour of the entire program. No matter how carefully you think you've tested, a thorough test of the manual against the program is likely to highlight problems

that you missed. Apart from its effectiveness as a revealer of new bugs, the manual is a valuable test plan because of the credibility it provides for bug reports. Many bugs that you find initially look small, and are deferred. But if one of these bugs is exposed again by following the instructions or suggestions in the manual, you should re-open it, or re-report it. The bug will appear much more significant if you can run across it by doing simple things that any user would do (such as following the instructions in the manual.)

- ***It's a training tool.*** New programmers and testers who join the project fairly late use the manual to learn about the program. Errors in the manual confuse them and can result in new software errors or unproductive testing.
- ***It's the external specification***. Many product development groups stop maintaining the external specification (if they ever maintained it) (assuming they ever wrote one), and rely on the manual to serve this purpose. Errors in the manual cause all of the problems that you get from errors in the specification.

Here are some of the problems that an error-ridden manual causes the Marketing and Sales departments:

- ***Delayed collaterals***. Before the product ships, your company probably creates brochures, application notes, output samples, and several other materials that highlight what the program can do. The people who create these are probably not experts with the program. They probably rely on the most recent draft of the manual for guidance. Combine a few errors in this draft of the manual with a few bugs in this pre-release version of the program, and it can take several days to produce output that should have been done in an hour.
- ***Delayed packaging***. The Marketing staff probably create sample output that will be photographed and put onto the program's box. As with the collaterals, bad documentation can delay this work. Also, they'll probably use the manual as source material for box copy – all those descriptions of what the product can do, what equipment it's compatible with, etc. Errors in the manual can result in errors on the box. If your company is lucky you'll discover these before the box goes to the printer, so the cost is just the cost of delay, not reprinting or stickering the box. These delays, however, can be very expensive. Your company can save money, if it is manufacturing thousands of boxes, by booking time with a printer well in advance. However, if you did this, but your company doesn't get your artwork to the printer on time, it might have to pay for that reserved press time, along with having to pay extra for last-minute printing arrangements. These expenses can dwarf that $6000 testing cost.
- ***Erroneous ad copy***. Claims in the manual can turn into claims in the advertising materials. Errors in the manual turn into errors in advertisements. Now you have to spend money on corrective advertising. And some people will accuse you of false advertising or will want to treat your ads as warranties. You will discover the hard way that $6000 buys more tester-hours than lawyer-hours.
- ***Problems from pre-sale use of the manual.*** The manual might be available at trade shows, or sent to user groups or prospective customers on request. People who review your product for magazines will rely on the manual for information and instructions. In each of these cases, errors in the manual can be embarrassing and expensive, and dealing with them can waste the time of your senior Marketing or Sales staff.

Here are some of the problems that a weak manual will cause your Training staff.

- ***Your customers will need more training***. If you provide training with your product, more of your customers will need the training if your manual is bad. If you provide "free" training, your training costs rise. You train more people, and you probably have to train them for more hours per student. If you charge for training, your customers will be aware of all the extra money they're spending to be able to use your product.
- ***Your Training department might write its own manual***. I've seen this several times. The Training staff decide that they don't like the manual, so they write their own and give it to their students. Along with the expense of duplicated effort, this creates new testing requirements and new opportunities for providing incorrect claims to your customers.

- ***Training sessions can become adversarial***. If the manual has errors, some students will quote them to the Trainer during the session. When the Trainer says X, the student says "But the manual says Y. Why does it say that?" This can be disruptive and embarrassing, especially if it happens several times. It reflects badly on the product and on your company, not just on the Trainer.

And finally, here are some of the problems that errors in the manual will cause your technical support and field support staff:

- ***It takes longer and costs more to prepare answer books***. As part of preparation for release of a new product, a well-organized Support staff prepares books or a database of answers to the questions that they expect will be frequently asked, or that will be difficult to answer. Errors in the manual make this task harder and more prone to error.
- ***It makes calls run longer***. Sometimes the best way to handle a call is to alert the customer to the relevant section in the manual. This can often be done without insulting the customer, and if that section provides a dear, detailed answer to a complex question, it might be just what the customer needs. A question that would take an hour of handholding and explanation over the phone becomes an easy two-minute call. On the other hand, if the manual is untrustworthy, Support can't push the customer back to the manual. Calls that should be short become long.
- ***It results in more calls***. If the manual says that the program works a certain way, and the program doesn't work that way, people will call and ask what the problem is. Along with generating calls about the errors, errors in the manual teach some people that the program (or the manual) is untrustworthy. Therefore they feel more justified in calling more often and more quickly for more help, and in demanding that the help be provided free of charge because, after all, the program is full of errors.
- ***It results in more difficult calls***. Over the past two years, I've interviewed several Support mangers and staff as part of my research for a new book. One of the common threads was the difficulty of dealing with people who were misled by the manual. Imagine dealing with someone who followed the instructions but wasn't able to achieve the result that the manual promised, or worse, who followed the instructions but lost data or got into trouble. These calls are especially hard if you don't have an easy solution to the callers' problems. Calls like these burn Support staff out. People quit and your company has to hire and train new staff. This is an outrageously expensive cost of a low quality manual.

As a Test Group Manager and as a Documentation Group Manager, I've run into management resistance to doing what was required to test the manuals thoroughly. I respond with a rough sketch of quality-related costs and risks. I lay out the near-term benefits of good testing and the near-term and long-term risks of poor testing of the manual. I present this to my manager, and to my manager's manager's manager if necessary.

This is persuasive stuff. To people who care about quality, you are talking about customer satisfaction. To people who care about schedules, you are talking about efficiency – a well-tested manual helps you find bugs sooner and train Development, Marketing and Support staff faster. To people who care about money, you are spending a little to save a lot. To people who are afraid of making decisions, you are providing a business case that makes this decision look safe and obvious. Finally, note that when you identify the in-house staff who will benefit from a thorough test of the manual, you create a list of people who have a stake in lobbying their management to make sure that you have the budget you need to do this task well. Go talk to them.

Doing the Testing

When you test the manual, you should use the program exactly as the manual says. Try every example. Verify every Note, every Caution, every Tip. Try every suggestion. Check every explanation of every error message. Check every definition in the manual's Glossary to be sure that they all make sense in the context of this program.

Check every limit (such as a claim that the program can store 100,000 records) and ask whether the program's behavior is reasonable at the limit. For example, if a program can store 100,000 records, don't

just generate 100,000 records and see if it holds them all. How long does it take to retrieve one or to enter one as you approach this limit? It might be reasonable for the program to slow down as the database gets bigger, but if adding each new record takes 36 hours of computer time, you know that the manual should be setting readers' expectations at a lower number, or there will be significant dissatisfaction.

Limits aren't the only claims made about the program's capabilities. Check all the other claims too. Be skeptical. If the manual says that printed output is gorgeous on all supported printers, check it with an old 9-pin dot matrix, or some other printer that is likely to provide less gorgeous output than your samples in the manual. Would the owner of that printer agree that you were getting as gorgeous a result as you could get from that printer?

Every time you see a mismatch between the program and the manual, note the mismatch on the manuscript (for the writer) and file a bug report. The bug report says that the manual and the program differ, and asks which one is correct. Close the report as Fixed when either the program or the manual is changed to eliminate the mismatch.

When you see outdated information in the manual draft, give the writer your notes on the recent design changes to the program.

When it appears that the writer didn't understand the purpose of a feature, try to provide the writer with an explanation (if you can do it quickly).

If a section of the manual is confusing to read, test this area of the program. This is a sign that this part of the program's design is probably too complex. The area's bug count will probably be high. This is an example of an important theme – let the manual suggest to you areas of the program that will need more intense testing.

When you are testing Help, you have additional concerns, because the Help is coded with jumps and branches, just like any other program. It can have bugs, maybe serious ones. You have to test these, along with the content. My experience is that moderately thorough Help testing can easily take twice as long as testing of the manual.

Along with testing Help and the manual, you should be testing box copy, brochure copy, and other information that the company prepares in order to send to many customers.

In Closing

Thoroughly testing your documentation might be the most cost-effective thing you can do to significantly reduce the probability that your company will be sued. It's also an effective way to find bugs in the program, a strong assistance to the writers, and a valuable source of information for your company's in-house users of the not-yet-released program. So many companies fail to take the accuracy of their documentation seriously that you may have to educate your company before they'll grant you the time you need to do this job properly. Think your case through in terms of quality-related costs, and schedule/efficiency-related benefits. A strong analysis and presentation along these lines will probably get you the approval you need for the testing, and enhance your credibility as a business decision-maker at the same time.

---

[1] *Daughtrey v. Ashe* (1992) South Eastern Reporter, Second Series, volume 413, p. 336 (Supreme Court of Virginia).
[2] *Daughtrey v. Ashe* (1992) p. 339.
[3] *Daughtrey v. Ashe* (1992) p. 339.
[4] Rather than relying on subsection (b) of Section 2-313 of the Uniform Commercial Code, which says that a description of the goods is a warranty, the Court might use subsection (a) which says that "Any affirmation of fact or promise made by the seller to the buyer which relates to the goods and becomes part of the basis of the bargain creates an express warranty that the goods shall conform to the affirmation or promise." Manuals are large, detailed collections of statements (affirmations) of fact about the program that they document.
[5] Customer Care, Inc., 235 Martling Ave., Tarrytown, NY 10591.
[6] In the case of *Burroughs Corporation v. Hall Affiliates* (1992) Southern Reporter, Second Series, volume 423, p. 1348 (Supreme Court of Alabama), Burroughs was held liable for fraud on the basis of false statements from its sales representative to a customer. There was no claim that the salesperson *deliberately* misled the customer.

Alabama and several other States allow customers to hold sellers liable for innocent and negligent misrepresentation, as well as for deliberate fraud.

[7] The laws vary across States. Good discussions are in Clark, B. & Smith, C. (1984, supplemented 1994), *The Law of Product Warranties*, Warren Gorham & Lamont, and Rosmarin, Y.W. & Sheldon, J. (1989, supplemented 1994) *Sales of Goods and Services* (2nd Ed.), National Consumer Law Center.

[8] If the customer looks through the manual before the sale, the court is likely to analyze the manual in the same way as it would analyze advertising materials, such as brochures. Statements of fact in brochures are generally treated as warranties. Published cases involving manuals are rare, but a customer did examine an operator's manual before buying a hay baler in the case of *Schlenz v. John Deere Co.* (1981), Uniform Commercial Code Reporting Service, volume 31, p. 1020 (United States District Court, District of Montana). Statements in the manual were held to have created an express warranty of safety.

[9] If you urgently need a detailed discussion from me now, I can send you a few pages from a draft of David Pels' and my forthcoming book, *Bad Software: Get Treated Fairly When You Buy Computer Software*.

[10] *Step-Saver Data Systems, Inc. v. Wyse Technology and The Software Link, Inc.* (1991), Federal Reporter, Second Series, volume 939, p. 91 (United States Court of Appeals for the Third Circuit) and *Arizona Retail Systems, Inc. v. The Software Link* (1993) Federal Supplement, volume 831, p. 759 (United States District Court, District of Arizona).

[11] Lemley, M. (1995) "Intellectual Property and Shrinkwrap Licenses," *Southern California Law Review, volume 68*, page 1253, reviewed court cases and statutory law in 34 countries.

[12] Campanella, J. (Ed.) (1990) *Principles of Quality Costs: Principles, Implementation and Use.* (2nd Ed.) American Society for Quality Control, Quality Press.