

CHEATING the quality triangle

BY GEOFFREY J.S. HART, *Senior Member, Montreal Chapter*

What's our favorite tongue-in-cheek mantra in technical communication? We have many, but "Fast, cheap, good—pick two" always makes the top-three list (right after "read the manual" and "you want that *when?*"). The phrase often takes tangible form in the so-called quality triangle, an illustration of a triangle with the names of each of these factors occupying a side. This triangle analogy speaks to us so clearly because resources always seem limited, and as a result, devoting resources to any two factors prevents us from applying those resources to the third factor. This assumption is so well accepted that you could almost call it technical communication's law of conservation of energy.

Fortunately, the quality triangle isn't really a law of nature, and thus, it's not immutable. It is possible to simultaneously improve speed, cost, and quality—if you can forget about geometry and "think outside the triangle." In this article, I've presented a baker's dozen tips that can get you started.

Faster

Doing something faster always carries with it the implicit risk of compromising quality ("quick and dirty") or imposing additional costs ("hire more writers"). This risk can occur if *fast* means "care-

less” or “throwing money at a problem,” but neither situation has to be the case. How can you increase your speed without increasing cost or decreasing quality?

Think Parallel

Triangles have no parallel sides, so when we discuss the quality triangle, it’s hardly surprising that we completely forget about working in parallel. Accomplishing two or more tasks simultaneously is obviously faster than waiting to complete each task before beginning the next one.

The document approval process is one place where an eminently *logical* approach often becomes *inefficient*. For example, final reviews and approvals may occur at several levels: the documentation manager, the research and development director, the marketing manager, the engineering supervisor, the legal department, and (for some multinational firms) the head office. Traditionally, these approvals are sequential, with each person reviewing a document only after it has been approved by someone lower on the chain of command. Given that each phase in the approval process leads to corrections before the document moves on to the next phase, and given the dead time while the document is in transit between offices, several days can easily be lost between each approval. In contrast, conducting all these reviews simultaneously is often easy to sell to management if the technical and editorial reviews that precede the final approval process produce high-quality documents that require little subsequent revision.

Periodically getting together with your colleagues to evaluate current processes can provide considerable payback. In the case of sequential document approval, the value added by each consecutive cycle of approval plus revision often doesn’t justify the delays this approach entails. Rethinking your processes also lets you determine which parts have grown atherosclerotic and no longer serve a constructive purpose. Many of your findings will lead you to eliminate or streamline existing processes, saving time without costing money or resources.

Know Your Deliverables

Understanding what you must deliver at the end of a project (the *deliverables*)

also helps you move faster: If you don’t know what you *must* produce to meet the needs of your customers, you may succumb to the temptation to document *everything*, no matter how irrelevant. Failing to identify all the deliverables when you start a project means that you’ll generally miss at least one important deliverable while you’re busy working on things that provide little benefit for the customer. You can help all of your users by focusing most of your attention on the aspects that provide the most payback to customers.

Ironically—and somewhat unfortunately—careful identification of the deliverables can pose problems, since focusing narrowly on the goal sometimes leads us to overlook obvious shortcuts. For example, many of us consider the entire printed manual to be the deliverable, and thus, the entire manual becomes the product we deliver for peer, technical, and management reviews. Seen from the customer’s viewpoint, this approach makes little sense, since our customers see *each chapter* and *each topic within a chapter* as the deliverables. It makes more sense to send each chapter or topic for review as soon as it’s complete. Reviewers are only human, and they behave predictably when a 600-page manual lands on their desks: They immediately set it aside and start looking for something easier to do. Breaking the task into smaller chunks diminishes the incentive to avoid the work. The total amount of work remains constant—the whole manual must still be reviewed—but spreading it out helps motivate reviewers to do the work.

Don’t Reinvent the Wheel

The two preceding points demonstrate how understanding delays can help you minimize or even eliminate them. One problem involves failing to reuse approved information or techniques, thereby forcing us to reinvent them each time they are needed. On a large scale, reusing information can become “single sourcing,” in which the print manual becomes the online help or vice versa, but this approach also works well on a smaller scale.

For example, most writers remember to

copy license statements or disclaimers from existing documentation rather than rewriting them from scratch, but few use an approved chapter in a manual as the template for subsequent chapters. Most of us encounter the occasional difficult writing problem and spend considerable time solving it ourselves, rather than asking our colleagues how they or others have solved the problem. Sometimes, a little lateral thinking can help us discover that the solution is not to attack the problem at all. If we’re having trouble developing the manual, for example, could we simply negotiate a deal with a publishing house (such as the publishers of the “For Dummies” series) to provide printed documentation, leaving us free to focus on the online help? We can also find ourselves repeatedly coping with a problem and developing workarounds rather than seeking the root cause of the problem and preventing it from arising in the first place.

Maximize Existing Technology

Appropriate use of the technology you already have in place also helps the work go faster. Most of us develop only a superficial understanding of our crucial tools, such as word processors, and thus do many things manually that the software could do automatically, much faster, and more accurately.

For example, Microsoft *Word* can automate the process of fixing certain mistakes through underused features such as autocorrect, macros, and customization of the interface. The autocorrect feature monitors your writing as you type and fixes a variety of common problems (such as typing “the” as “hte”). That process alone saves time, but teaching *Word* to autocorrect your own unique typing problems offers even better payback. For example, in a recent manuscript I kept mistyping the acronym OFSWA as OSFWA, something easily fixed by adding my typo to the autocorrect dictionary. There’s another payback from this approach: If *Word* corrects typos as you type, you won’t have to waste time correcting them during the final spellcheck. Similarly, add jargon to your personal spelling dictionary so that *Word* doesn’t repeatedly question unorthodox spelling. If you work in a field such as medicine or law, you can often purchase

special dictionaries that contain most of the jargon from your field.

In some cases, you waste considerable time retyping standard information that recurs frequently. Storing this information in a file allows you to copy the text and insert it into new files during editing. For multi-keystroke tasks that you perform repeatedly, you should program a macro that reduces the task to a single mouse click or keystroke. *Word's* immense potential for customization also works in your favor. For example, you can reprogram the keyboard shortcut Control-F4 (which, by default, closes a document) to launch a macro that automatically updates all fields in the document (a function normally performed by the F9 keystroke), makes a backup copy of the file in your

personal backup directory, and *then* closes the document. (I've discussed macros in more depth in the February 2001 of *Intercom* in my quarterly column, "Effective On-screen Editing.")

Choose the Right Technology

The problem with technology is that it sometimes becomes a roadblock. Once a technology is in place, we often find ourselves using existing tools simply "because they're there," rather than because they're efficient.

For example, the final step in producing printed documentation (the printing process) used to consume considerable time: In the early days of desktop publishing, designers would print camera-ready copy on a high-resolution laser printer, paste up the graphics, then ship the resulting pages to a print shop to create printing plates. Using word processors as our publishing tool lay at the root of this problem, since word processors are poorly suited to an efficient production process. Desktop publishing software, which was designed to facilitate all-electronic production of printed documents, proved to be a more efficient tool, and the only efficient tool for producing color publications. So until recently, if we wanted to print a manual directly from the software in which we created it, we needed a dedicated desktop publishing program. Adobe's *Acrobat* technology now lets writers use word processors to produce electronic files that can be sent directly to the printer. This approach still doesn't work well for full-color documents because word processors don't understand the color models used in offset printing; however, high-speed color inkjet and laser printers can produce acceptable results in many cases, and it's only a matter of time before word processors will let us produce color publications.

Cheaper

Computer technology offers compelling advantages, the more so now that the cost of the hardware keeps dropping even as its speed increases, and the effectiveness of the software keeps increasing while the price remains mostly constant. Both ongoing improvements mean that

investing in appropriate new technologies is less expensive than ever before, and the payback potentially greater. But technology itself is only part of the solution.

Spend Money to Save Money

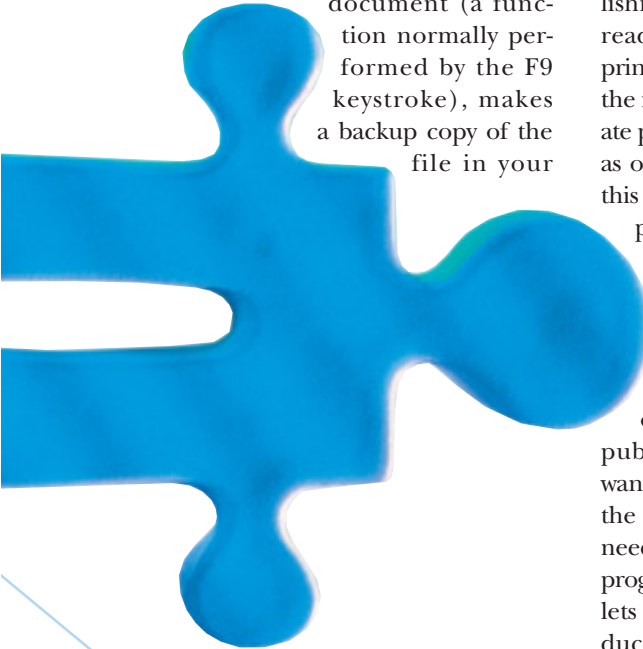
There's an old saying that "you have to spend money to make money," but sometimes you have to spend money to save money. The trick is to buy the tools you need to do your job most efficiently. In many cases, sticking with the current tools (an old computer, a small monitor, or unsuitable software) can actually hamper us to the point that we need contract writers to help with the workload. Even when the software itself is productive, it may entail additional costs in certain phases of the production of documentation.

For example, Microsoft *Word* is an inexpensive desktop publishing solution, particularly since it often comes "free" with new computers, but it's not the best tool for every job. The program lacks the power, reliability, and stability of a dedicated desktop publishing program such as *FrameMaker*, and you can't yet produce color documents in *Word* without expensive manual workarounds. Buying a tool such as *FrameMaker* certainly entails an additional startup cost, but that expense may be amply repaid in terms of reduced troubleshooting costs and downtime, not to mention reduced costs at the printer.

The benefits of using a more productive tool often greatly outweigh the costs of purchasing that new tool and training everyone to use it. More work gets done per billable hour, thereby lowering the client's per-hour costs. Furthermore, it's often possible to improve quality at the same time. True desktop publishing software has superior typographic tools, and by working with imagesetters (high-resolution printers that output pages to film), it provides higher resolution for the text and graphics. In many cases, the costs are also lower than with traditional techniques.

Measure Twice, Cut Once

"It goes without saying," one of my favorite tautologies, often introduces the phrase "planning is crucial." Everyone's heard the expression "Measure twice, cut



The benefits of using a more productive tool often greatly outweigh the costs of purchasing that new tool and training everyone to use it.

once,” but how often do we take this adage to heart and plan a project so carefully that we need cut only once? Unnecessary work and rework cost us time and money.

For example, consider the cost of producing a large volume of printed information that could be better implemented as online help. Manuals are expensive to print, and the greater the proportion of the information you move online, the smaller the manual and the less it costs to print. At the same time, moving *too much* information online or designing ineffective online help may prevent some customers from finding necessary information. That leads to increased telephone support costs, lost sales, and customer dissatisfaction. By understanding customer needs and defining the correct mix and design of online and printed documentation, you can greatly reduce printing costs as well as support costs.

Balance Quantity with Cost

The previous point provides a hint of the tradeoff between *quantity* and cost: Doing more work always costs more, all else being equal, and sometimes you'll discover you've been doing things you no longer need to do. By reducing the quantity of work, you lower your costs, and if you've carefully focused your efforts, you won't affect quality much, if at all.

When, for example, was the last time you saw a tutorial on how to use Microsoft Windows in a software manual? This kind of introductory material was common back when our audience was making the transition from text-based interfaces such as MS-DOS to graphical user interfaces, but increasingly, we can safely assume that our audience understands the basics of a graphical interface. What invalid assumptions do we continue to make about our documentation? The risk of challenging such assumptions, of course, is that the assumptions arose for a good reason, and challenging them risks overestimating our audience's knowledge. To be safe, you must perform at least a cursory audience analysis to confirm that you've correctly understood your audience's needs.

Quantity also becomes important in the context of *Pareto optimization*, a business

approach which recognizes that a small portion of a product may provide the majority of the product's benefits. The consequence of this notion for documentation? We might need to provide truly comprehensive, detailed information for only the 20 percent of our documentation that readers use 80 percent of the time; for the 20 percent of the features that cause 80 percent of the problems; or for the 20 percent of the tasks that have the most serious consequences if the user fails at the task. For everything else, it might be acceptable to provide less comprehensive or less detailed information. Of course, we'll still have to spend the time and resources to find out what that crucial 20 percent happens to be, and doing so is neither cheap nor fast.

Cut Back on Paper

The final cost-saver I'll propose revolves around the proverbial paperless office. I'm a huge fan of paper-based documentation, in part because of the current primitive state of online help systems, but that doesn't mean I'm blind to the importance of striking an appropriate balance between online and paper information for each audience.

Most software that I use presents information inappropriately: In many cases, the information has been dumped online purely so that the developer doesn't have to pay printing costs for manuals. Providing documentation in PDF format, unlinked to the software and formatted for letter-size paper rather than the screen, shifts the printing costs to our customers; sure, it saves *us* money, but you can bet the customers aren't happy. Discussions with my colleagues suggest that this trend is continuing.

Minimizing our use of paper has obvious consequences for costs, but doing so without sacrificing customer satisfaction requires us to learn how to replace paper documentation dumped online with true electronic performance support systems that integrate the documentation and interface so well that there's less need for printed documentation. But paper documentation won't be entirely eliminated any time soon, and we're going to have to do a better job of using online information judiciously.

Better

Quality is too complex an issue to handle in such a short article, but searching the Web using the keywords "quality improvement" will turn up more resources than you have time to consult. Here I've provided just four ways you can improve quality that don't cost you more time or money, and may in fact save you both.

Produce More Usable Products

Yet another of our mantras is "you can't fix a bad interface through good documentation." That's true on several levels, but producing a product that requires less documentation has important benefits in terms of documentation quality. How?

- The more usable a product is, the less user assistance it requires, and thus the less documentation. Consider, for example, how much of a typical documentation project involves developing workarounds for unclear or inefficient interfaces.
- Reducing the amount of documentation a product requires lets you devote more attention to improving the quality of the material that remains to be written.
- The less material you write, the less material there is to review and revise; devoting the same amount of reviewer time to the reduced amount of material improves review quality and reduces the resources (people and time) required for the review.
- Identifying and resolving interface problems as you write the documentation forces you to work more closely with the developers, helping you to forge long-term relationships and synergies that strengthen the overall product over time. In particular, helping developers to better understand their audiences leads to better interfaces that require less documentation.

There's a myth that technical communicators can't work with developers. It's not true. Start working with your developers today.

Perform Better Reviews

I've already touched on reviews, but it's worth repeating that the review stage

is the easiest place to correct the most serious quality defects. Unfortunately, the most common review procedure I hear discussed concerns asking several subject matter experts to review the entire documentation package.

Apart from the time problems we considered earlier, significant quality problems arise from this approach. Given that reviewers have a finite amount of energy and time to spend on a review, asking them to spread that energy over the entire documentation package in a single step leaves fewer resources to focus on each portion of the package. Breaking the task into smaller chunks focuses their energy more intensely on each chunk, thereby improving the quality of the review. Two approaches can make better use of this energy:

- Continue sending reviewers the entire document for review, but break it into smaller chunks, sent at more or less regular intervals. This alternative lets reviewers recover their energy between

reviews and do a better job of each individual review.

- Consider sending reviewers only the sections of the documentation for which they're truly capable of doing expert reviews. This approach avoids wasting reviewers' time on material they don't understand. However, because overfamiliarity with a product can lead experts to miss simple errors, it makes sense to include some other reviewers in this approach.

Another important solution relies on careful editing. Writers commonly complain that technical reviewers provide more comments on the writing style than on the important technical points they've been asked to review. The solution? Provide reviewers with a thoroughly edited document that has no grammatical or spelling errors. In the absence of these errors, reviewers must, by default, focus on the substantive issues.

Stay on Track

I've mentioned planning as a means of reducing costs, but planning also has salutary effects on quality. Every time you take off on a tangent and produce documentation that wasn't part of your original plan, you're wasting time that could be spent improving the quality of the truly important material. You also run the risk of forgetting a deliverable or running out of time to document something important. Well-designed documentation plans help ensure that you adhere to the plan, and that changes arise only in response to important unforeseen needs. After all, even the best planner occasionally misses something important. But most of the time, it pays to stick to the plan.

Perform Iterative Reviews

Iterative reviews are another means of improving quality. In this approach, early reviews of relatively small portions of the documentation let you identify certain recurring problems such as the use of passive voice, incorrect terminology, or ineffective presentation of information. Correcting these problems early in a long project means that you won't have to cor-

rect them in all subsequent work or (worse yet) in one marathon session at the end of the project.

Maintaining consistency throughout a long document can be difficult even for trained editors. Using iterative reviews helps you become increasingly consistent in your approach and reduces the amount of work required to impose consistency later. It can also save enormous amounts of time along the way, since figuring out how to proceed early in the project means that you can immediately implement that solution in each subsequent phase. The time you save can be spent on improving quality.

Pick Any Three?

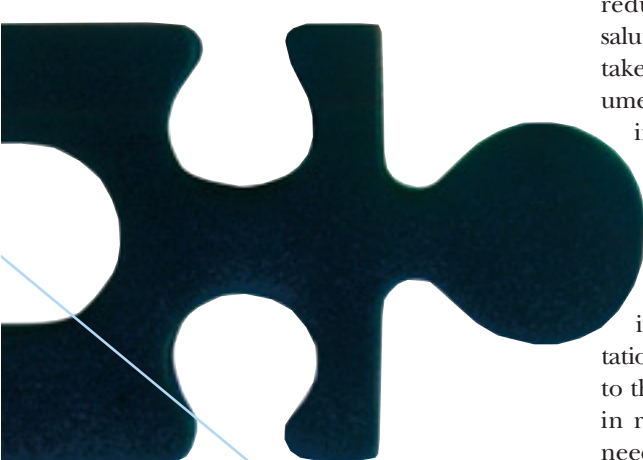
None of the approaches I've suggested changes the simple fact that we must produce documentation fast, with good quality, and at a reasonable cost. In that sense, the quality triangle really does reflect a fundamental aspect of our work. Cheating the quality triangle doesn't violate any immutable laws of nature; rather, it involves thinking about what we're doing and figuring out how to make those laws work *for us*. The result? Even when inadequate resources force you to sacrifice some degree of speed, cost, or quality, you can still minimize the magnitude of those sacrifices. Better still, you may be able to start with a faster process, a lower cost, and a higher level of quality than would have been the case if you simply accepted the inevitability of tradeoffs.

I've suggested more than a dozen ideas to get you started on that path, but this list isn't exhaustive. Nor do I claim that these suggestions will work under all circumstances. Instead, I hope to get *you* thinking outside the triangle and finding even better solutions that fit your own specific context. **i**

SUGGESTED READINGS

Hart, Geoffrey J.S. "The Style Guide Is Dead: Long Live the Dynamic Style Guide." *Intercom*, March 2000: 12–17.

Hart, Geoffrey J.S. "Ten Technical Communication Myths." (Guest editorial) *Technical Communication* 47(3): 291–298.



Provide reviewers
with a thoroughly
edited document
that has **no**
grammatical or
spelling errors.