

# WORDS



A Quarterly Bulletin for Technical Writers & Communicators

Volume 3 | Issue 3 | August 2011

## The beginning

- A good many technical writers spend their professional lives documenting software. If they have been in the game for a while, they would have seen how software development seems invariably to lurch from crisis to crisis, with deadlines missed, features dropped and the product born, well over-term, with bugs aplenty. And naturally no pay rises for persevering with all the chaos—for the bank is bare from the inevitable budget-over-run.

The latest strategy for avoiding such dollar-burning chaos is called *agile development*, a natty name whose aptness is yet to be proven. A software project is to be decomposed into numerous small teams each given responsibility for a small feature and given, too, a splash of design flexibility. Like Darwinism on drugs, each feature is, with evolutionary fervour, modelled, prototyped, remodelled, reprototyped, remodelled and ... until what was promised—or something akin to it—is passed as fit for purpose. It is a process of many whirlwinds, with the poor technical writer, trying to make sense of it all, often battered from pillar to post. In this issue, Lana Brindley reflects on a document design methodology that could help writers crawl out of an agile project relatively unscathed.

- Lana is a well-known advocate of open source software. So too is James Hunt. In this issue, James describes a single-source software toolkit the core of which has been around for ages. That core—*Latex*—is presently used by possibly millions of folk in writing technical and scientific documents. James recommends that technical writers have a look at *Latex* as a decent alternative to proprietary software, one more likely to keep baldness at bay.
- This issue continues the critical examination of the Information Mapping documentation methodology begun in the last issue. This time, the notion of *document-wide chunking*, one of the key recommendations of Information Mapping, is put under the microscope—and found somewhat lacking.
- What do newcomers think of technical writing? In this issue, five recent graduates reflect on the ups, the downs and the forlorn expectations as they work through their first few projects as technical writers.

**Geoffrey Marnell**

Editor [geoffrey@abelard.com.au]

## Contents

Gettin' agile wit' it.....	1
Does size matter? Document-wide chunking.....	3
Technical writing with open source software: LaTeX, Subversion and Inkscape .....	10
Fresh reflections on technical writing .....	15
Journals.....	20
Book review .....	21
Tips and tricks.....	23
Miscellany .....	24
Mindstretchers .....	25

## Gettin' agile wit' it

**Lana Brindley**

Many people dream of writing a great novel: a master work, a weighty piece of literary fiction, a sci-fi thriller, or maybe a torrid romance. Few begin. Fewer finish. But for a technical writer, starting a new project happens on a regular basis. Technical writers need to overcome that fear of the blank page, and learn to see it as an opportunity to reach out to their audience and make a valuable connection, not just an empty space to be filled with words, any words. And they need to finish it, on deadline, on spec and on budget.

One of the main tools in a tech writer's arsenal is planning. We all have our own way of doing it, and we all follow different methods of gathering information and sorting out our ideas. Not just the planning stages, but the entire writing process varies from author to author, client to client, and sometimes even from project to project. Authors who have been winging it for years can continue to do so indefinitely. Their experience and industry knowledge serves them well. They can eye up a project scope several months out and do what is needed to deliver as expected. But what about the mere mortals in this industry? What about the poor people who, like me, came into tech writing as a clueless newbie and craved structure and direction? With no one hovering over me and telling me what to do next, I went looking for a framework.

Traditionally, documentation frameworks have been based on a *waterfall model*. In this model, each of a number of steps flows one into the next. A typical example is the popular JoAnn Hackos five-phase model, but others use six-, seven-, and even eight-phase models. They usually follow a fairly predictable arc beginning with defining the problem and conducting a user analysis, and ending with drafting and editing the final product.

Waterfall models have been used successfully for many generations, because most technical development has worked on similar principles. In the past few years, though, software development in particular has started to move to processes that make greater use of *iterative prototyping*. This style of development goes by many names, but they can be grouped easily under the banner of *Agile* software development. With this trend towards iterative development, the production of documentation is being forced to change. Most Agile software development models acknowledge that documenting in an Agile environment can be fraught with difficulty. Document too early, and much of the documentation is out of date by the time development is complete. Document too late, and there is not enough time to complete the documentation to a sufficiently high standard. While the old waterfall-style models still mostly work, the project manager and authors need to be prepared to

be flexible with their processes, and accept that rapid product development can easily lead to last minute changes and updates.

Unfortunately, most of the texts that cover this topic simply advise software developers to get someone else to do it, which is great if you're doing the asking, and a little more difficult if you're the person being asked. However, all is not lost. Because documentation is usually produced by a separate documentation expert (or, if you're lucky, a whole documentation team), they are in a position to work alongside the Agile model, rather than within it.

---

**The Agile dilemma:** Document too early, and much of the documentation is out of date by the time development is complete. Document too late, and there is not enough time to complete the documentation to a sufficiently high standard.

---

Agile models still go through an organised process of determining deliverables and dates for delivery in the early planning stages, hence the documentation team can often use a more rigid development model based on these

requirements. The old waterfall models can be flexible enough to work successfully, but asking development teams to commit to designs and plans early in the project can sometimes cause tension, especially when the developers have no clear idea of where their prototyping and testing will take them. And tension between documentation and engineering is something usually best avoided.

The Planned Rapid Document Development (PRDD) model—developed by Kimball and Hawkins in their book *Document Design*<sup>1</sup>—focuses on users contributing to each draft in a series of iterative drafts. Each draft is analysed, reviewed and commented upon before being sent back to the author, ready for the next iteration. This model works particularly well because it is acknowledged that while users are rarely good designers, they can be good refiners, and putting drafts in front of users early in the process means their feedback can be incorporated and built on right from the beginning.

Kimball and Hawkins also acknowledge the importance of research and planning. They criticise purely iterative models for producing final products that are sometimes incomplete or imperfect, due to their lack of planning and rigid adherence to a finite production schedule. In the *planned iterative model*, the documentation team begins by attempting to determine what the client and the user require, and then using a series of iterations to move ever closer to that goal. Each iteration produced by the PRDD model is subjected to rigorous testing and analysis before the next iteration begins. Kimball and Hawkins recommend that documents go through at least three and no more than six iterations in total. Iterations are intended to increase in fidelity

---

1. MA Kimball & AR Hawkins, *Document Design: A Guide for Technical Communicators*, Bedford, 2007, ISBN: 0312436998.



commercial  
translation  
centre

For Effective  
Translation/Localisation  
Of Technical Documentation  
In 80 Languages

Australia Wide Service Since 1988  
Call 1800 655 224 (Australia Toll Free)  
Email: mail@ctc.com.au  
Web: www.ctc4.com

commercial translation centre

throughout the project, with early iterations being incomplete and fairly rough (that is, having low fidelity) and becoming more complete and polished (that is, have higher fidelity) over time.

The PRDD model acknowledges that both expert reviews and user reviews are important in achieving an effective document design. To this end, they recommend that both expert heuristic analysis and usability testing occur for each iteration. Heuristic analysis is designed to determine how well the document meets general design principles, but it does not give a comprehensive idea of how real users will interact with the document. For this, they recommend taking the prototype to the users and asking for their feedback directly, through usability testing. This can take several forms, depending on the product being developed. Usability tests range from observing users interacting with the product, to asking them to complete a survey, to interviewing them about their experiences, or any combination of methods.

However, the PRDD is just one of many frameworks for producing technical documentation. It is a nice blend of purely waterfall and purely Agile models and can be easily adapted to fit your own

---

The Planned Rapid Document Development model is a nice blend of purely waterfall and purely Agile models and can be easily adapted.

---

organisational requirements. Never underestimate the value in developing a documentation framework that suits your own unique needs as a writer. Every documentation model has its benefits, and the use of any model at all is preferable to blindly filling a blank page with words and hoping it's right.

However, it is also important to carefully review and select the model you use in light of each project, the type of product being developed, the client requirements, and the situation in which the final product will be used. By applying a great model, you are much more likely to produce great documentation.

### Lana Brindley

Lana Brindley has been playing with technology since that summer in the 80s when she spent the whole time trying not to be eaten by a grue. She has been writing since she could hold a pencil, and is currently writing technical documentation for Red Hat. Lana holds business degrees in marketing and information systems, and with any luck will have a technical communicators degree by the end of the year. She works from her home in Canberra, Australia, and occasionally leaves the house in order to berate university students and conference goers about passive sentence construction

---

## Does size matter?

### A critique of document-wide chunking<sup>1</sup>

#### Geoffrey Marnell

In the previous issue of *Words*, I dissected the claim by Robert Horn—the originator of Information Mapping, a writing methodology much loved by many technical writers—that research by American psychologist George Miller on the limitations of short-term memory shows that we should present information to readers in chunks of no more than  $7 \pm 2$  sub-chunks. This numerical boundary is what Horn called the *chunking limit*:

“The chunking limit is a guideline, based on George A. Miller’s 1956 research.”<sup>2</sup>

The result of that dissection? Miller’s research on the limitations of short-term memory in no way supports Horn’s chunking limit. I also quoted Miller himself stating that his research showed nothing at all about a readers’s ability to understand written

texts. Quite simply: Horn *wrongly* quoted Miller in support of Information Mapping. According to Miller, a limit on the capacity of short-term memory does not show that we should present information to readers in chunks of no more than  $7 \pm 2$  sub-chunks.

The fact that Horn erred in basing his chunking limit on the limited capacity of our short-term memory does not on its own disprove that some chunking limit is necessary for comprehension. Perhaps, then, the chunking limit at the heart of Information Mapping is still appropriate even if the work of Miller cannot be adduced in support of it.

Let’s assume, for the sake of argument, that there is some chunking limit. To what chunks in a document might it sensibly apply? Horn contends that it should apply to *every* level in a document:

“Writers should ... apply this  $[7 \pm 2]$  limit at every level of a written document [namely] ...

- blocks [that is, a group of  $7 \pm 2$  sentences about a common topic, or a list or a table]
- maps [that is, a group of  $7 \pm 2$  blocks]
- sections [that is, a group of  $7 \pm 2$  maps] and
- chapters [that is, a group of  $7 \pm 2$  sections].”<sup>3</sup>

Why?

- 
1. My thanks to Ian McGregor, whose thoughtful response to my previous article on Information Mapping prompted many of the thoughts expressed here.
  2. See [http://www.infomap.com/index.cfm/themethod/Mapping\\_FAQs](http://www.infomap.com/index.cfm/themethod/Mapping_FAQs) (viewed 23 January 2011).

“By chunking information the writer improves the reader’s comprehension ...”.<sup>1</sup>

Just how plausible is this claim?

First let’s note that there is nothing in Horn’s claim that chunking information improves the reader’s comprehension to suggest that he is using the word *comprehension* other than in its common-or-garden dictionary meaning:

“**comprehension** *noun* 1. the act or fact of comprehending”

“**comprehend** *verb* (t) 1. to understand the meaning or nature of”<sup>2</sup>

Thus Horn’s claim is that chunking improves a reader’s ability to *understand* what they are reading. Stripped of whatever academic mystique swirls about the concept of *comprehension* should help us see that the relationship between chunking and *comprehension-as-understanding* is a tenuous one indeed. For if such a relationship exists, then it plainly cannot apply to *all* types of writing. A novel, for instance, is largely unchunked. It will have paragraphs (that is, blocks) and it may have chapters. But it won’t have maps and very few novels have sections. And of the chunks a novel does have, no limit is adhered to by the author. A chapter could have hundreds, possibly thousands, of chunks. Does this lack of rigorous chunking in the form required by Information Mapping mean that we do not *understand* what we read in novels? Of course not.

## The ways a reader might read

Let’s leave novels and their kin to one side and consider instead user guides and policy manuals, the sort of documents Horn is primarily concerned with. Now a reader might tackle such a document in the following ways:

1. read some part of it (a section, map or block) without first paying any attention to the number of parent or grandparent chunks
2. read some part of it (a section, map or block) after paying attention to the number of parent or grandparent chunks
3. read some part of it without first paying any attention to the number of sibling chunks
4. read some part of it after paying attention to the number of sibling chunks
5. read some part of it (a chapter, section or map) without first paying any attention to the number of child or grandchild chunks

3. RE Horn, *Developing Procedures, Policies & Documentation*, Info-Map, Waltham, 1992, p. 3-A-2.

1. *ibid.*

2. Macquarie Dictionary, [www.macquariedictionary.com.au](http://www.macquariedictionary.com.au). Viewed 11 May 2011.

Component	Max. number
document	—
... chapter ...	unstated
... section ...	7 ± 2
... map ...	7 ± 2
... block ...	7 ± 2
... sentence ...	7 ± 2
... word ...	30

Figure 1: Information Mapping: chunking hierarchy

6. read some part of it (a chapter, section or map) after paying attention to the number of child or grandchild chunks

These six scenarios cover all logical possibilities.

## Looking upwards

Take scenario 1: reading without first noticing any higher-order structure. This covers the vast majority of cases. Most readers of technical documentation will go to the index or table of contents for help in finding the chunk—a block, map or section—they want to read, go to that chunk and start reading it. The number of chunks in the *chapter* of which the block, map or section is a part will pass unnoticed. For example, if I want to learn how to set preferences in Adobe Photoshop, the index points me to a particular page. I go to that page and start reading. I don’t even notice that this section (which is a *map* in Information Mapping terminology) happens to be in chapter 1; and I don’t bother—why would I?—to flick through the chapter looking at the other chunks in it. I’m interested only in learning how to set preferences. Now if I don’t pay any attention to the size of the parent or grandparent chunks—the sections and the chapters in this particular example—how can the size of those chunks affect my comprehension of what I am reading?

This way of reading user guides—indeed, of reading most informational texts—is widespread. Few readers, if any, read whole chapters in user guides. They dip into such documents when they want to learn (or be reminded of) how to do something in particular. They want, say, to activate time-shift on their personal video recorder and the steps are not obvious. They might then scan the contents pages or index, or electronically search, for the topic of interest and then read *just that topic*. Perhaps in the effective life of a personal video recorder, the owner might consult the user guide a dozen times but never read it through in its entirety. Much to the chagrin of technical writers—who might



spend many months writing a user guide—user guides are primarily consulted as a last resort: when the product it describes does not work as the user expected it to, or when the information needed cannot be got by asking someone else. And when it is, only a small part of it is consulted at any one time.

Let's explore this scenario a little further. Suppose that A is given version 1 of a user guide and B—of equal intelligence and experience—is given version 2. Suppose further that the only difference between the versions is that version 2 has more sections in chapter 1. A and B are then asked to read, say, a map in chapter 1, a map that is identical in both versions, and to do so without first counting the number of sections in the chapter. Must B find the map more difficult to comprehend than A? Surely not. Something that we do not look at in a document cannot possibly affect our comprehension of something that we do look at. And this will be the case regardless of the number of sections in the unlooked-at parent or grandparent chapter. It is also the case regardless of the type of chunk, even as far down the document hierarchy as the sentence. My understanding of "1. Sort the changes in postal code order" (to borrow an example from Horn) is in no way compromised by the number of unobserved sections or even maps in the grandparent chapter. In other words, a chunk-size limit is entirely irrelevant in this majority-case scenario.

Consider scenario 2: reading after noticing the size of the parent or grandparent chunk. First, note that this is an unrealistic scenario. How many of us, when we want to follow a procedure, first look at how many *sections* there are in the user guide? Few if any. How many of us, when we want to read a section of a policy document, first count the number of chapters in the document? Again, few if any. Still, for the sake of argument, let's explore the logic here. Suppose I read, in version 1 of a manual, "1. Sort the changes in postal code order". This is a one-sentence block in a procedure map. Suppose further that there are  $n$  sections in the grandparent chapter. Twelve months later, a new version of the manual is released and now that chapter has more than  $n$  sections in it. This is a fact I happen to notice. Indeed, just for fun, I count the number sections in the chapter. Now I go back to the procedure I read twelve months earlier and once again encounter "1. Sort the changes in postal code order". Is that step now more difficult to comprehend—and more difficult simply because of the extra sections in the chapter? Hardly. And again, the size of  $n$  is irrelevant. Or returning to our earlier example, will B find the map to read more difficult to comprehend than A simply because B *notices* that version 2 of the

user guide has more sections in it than version 1. Again, unlikely. In other words, a chunk-size limit is also irrelevant in this scenario.

Let's approach this scenario from a different angle: suppose that I want to understand a map in a section and I notice that there are  $n$  other maps in the section. Suppose further that the map I want to understand is very short: it has, say, just two blocks in it and each block is composed of just two single-clause sentences. If a chunk-size limit did affect comprehension of a child block, we would have to say that, although this map should be very easy to understand—it has only four sentences in it—it would become difficult to understand for values of  $n$  above that limit. In other words, readers would then find those four sentences difficult to comprehend, and solely because the section has 15, 20, 50 or whatever *other* maps in it. This is simply nonsensical. Simplicity is simplicity whatever may clutter around it.

## Looking sideways

Consider now scenario 3: read some part of a chunk without first paying any attention to the number of sibling chunks. Suppose I start reading a procedure without noticing that it has  $n$  steps in it. Perhaps there are 5 steps on one page and  $n - 5$  on the next, with the latter not observed until I turn the page. Can it possibly be that my comprehension of any one of the first 5 steps is somehow influenced by the number of steps on the next page? Will I better understand "1. Sort the changes in postal code order" if there happens—unbeknown to me—to be just 2 steps on the next page rather than 20? Again, an affirmative answer seems highly unlikely. And again, the size of the  $n$  seems entirely irrelevant. Further, the same applies regardless of the type of chunk we consider.

For example, my ability to comprehend a section is not compromised by the number of unnoticed sibling sections. Thus a chunk-size limit is irrelevant in this scenario too.

Consider now scenario 4: read some part of a chunk after paying attention to the number of sibling chunks. Again, this is an unrealistic scenario. How many of us count the number of steps in a procedure before working our way through it? Few if any. How many of us count the number of maps in a section before tackling a map? Again, few if any. Still, let's explore the logic of this scenario, if only to cover all bases.

Suppose that I read, in version 1 of a manual, "1. Sort the changes in postal code order" as the first step in an  $n$ -step procedure. Twelve months later, a new version of the manual is released and the very same procedure now has  $n + p$  steps. Suppose that I notice, before I read the updated procedure, that the number

---

**If a chunk-size limit did affect comprehension, how does it do so? Do we fully comprehend all sub-chunks up to the limit and then struggle with those beyond it? Or does the breach of the limit somehow spread its tentacles throughout the entire chunk and affect our comprehension of every sub-chunk in it, even the very first one?**

---

of steps is  $n + p$ . Will I now struggle to understand “1. Sort the changes in postal code order” for certain values of  $n$  and  $p$ ? Again, an affirmative answer seems absurd. Once again, a chunk-size limit is irrelevant.

A parallel argument to one adopted earlier is also relevant here. Suppose that I want to understand a section in a chapter and I notice that there are  $n$  other sections in the chapter. Suppose further that the section I want to understand is very short: it has just one map with, say, two blocks and each block has just two single-clause sentences. If a chunk-size limit did affect comprehension, we would have to say that, although this section should be very easy to understand—it has only four sentences in it—it would become difficult to understand for values of  $n$  above that limit (and where the value of  $n$  is noticed by the reader). In other words, readers would find those four sentences difficult to comprehend, and solely because the chapter has 15, 20, 50 or whatever other sections in it. Again, this is simply nonsensical.

If a chunk-size limit did affect comprehension, how does it do so? Do we fully comprehend all the sub-chunks up to the limit and then struggle with those beyond it? Or does the breach of the limit somehow spread its tentacles throughout the entire chunk and affect our comprehension of every sub-

chunk in it? Let’s assume, for the sake of argument, that the chunk-size limit is 9. Will I fully comprehend the first 9 steps in a procedure and then struggle with the remaining 2? Or will I struggle with all 11 steps? Neither case seems plausible. A step such as “30. Select **Exit** from the **File** menu” would be comprehensible to most, probably all, of the intended audience regardless of how long the procedure is.

### Looking downwards

Consider now scenarios 5 and 6. If we read a chunk in its entirety—as we would have to in order to claim in good faith that we understood it—then we cannot but read the child and grandchild chunks. It doesn’t matter whether we notice the granularity of the sub-structure, that is, the number of sub-chunks: to comprehend a chunk we need to read the sub-chunks. But more than that, we need to understand the sub-chunks. For instance, I can’t be said to understand a chapter without understanding the sections that make it up. And I can’t be said to understand a section without understanding the maps that make it up. And so on as we move down the document hierarchy (shown in figure 1 on page 4). So the question of comprehending a chunk—any chunk—comes down to comprehending the atomic components in the hierarchy, that is, the components that are not themselves chunks. In Information Mapping, such an atomic component—what Horn calls the “first and basic unit of information”—is the sentence. From an Information Mapping perspective a sentence is like a quark: it has no chunkable components. And Horn *explicitly* refuses to apply the  $7 \pm 2$  chunking limit to sentences.

Now if understanding a chunk means nothing more than understanding the sentences that comprise it, then scenarios 5 and 6 are reducible to scenarios 1–4 *applied at the sentence level*. But recall that we used a sentence—*Sort the changes in postal code order*—in each one of scenarios 1–4. We showed that this sentence could be understood independently of the number grandparent or parent chunks, and the number of sibling sentences. This was so even if we noticed the number of parent or sibling chunks. Now there is nothing special about this sentence that gives it a better chance than any other of being independently understood. Hence we can generalise: if a chunk is understood only if we understand the sentences that comprise it, and if a sentence—any sentence—can be understood irrespective of the size of any chunk in the document of which it is a part, it follows that comprehension is in no way impaired by the size of a chunk regardless of where that chunk is in the document hierarchy.

To sum up: contrary to the claims of Information Mapping, document-wide chunking is not a necessary pre-condition of comprehension.

### *Effective Onscreen Editing: new tools for an old profession*

Editors are increasingly being asked to edit on the screen using a word processor, but most are finding it challenging to transfer their skills to editing with a word processor. *Effective Onscreen Editing* teaches the basics you need to learn to make the transition, plus proven tips and tricks to maximise your productivity and effectiveness. The book describes general principles valid for any software, then illustrates the principles using Microsoft Word to make them more concrete.

Available as a printed book from Lulu.com and also available as an eBook optimised for onscreen reading.

Learn more at the book’s Web page:  
<http://www.geoff-hart.com/books/eoe/onscreen-book.htm>

 **Diskeuasis  
Publishing**

## What does it all mean?

It follows from our six scenarios—scenarios that cover all possible reading practices—that comprehension does not impose any limit on the size of the chunks in a document. Comprehension may well impose a limit on the size of sentences—based on the capacity of our short-term memory—but beyond that, size does not matter. So it would be false to claim that there are too many sub-chunks in a chunk for that chunk, or any other chunk, to be comprehended.

### Residual memory: an epiphenomenon of quantum incredulousness?

It might be retorted that while it is true that to understand a chunk you must understand the sentences in it, it is still possible that you could understand all the sentences but fail to understand the chunk. In other words, understanding the sentences in a chunk is a necessary *but not sufficient* condition for understanding a chunk.

With regards to procedure writing, this claim is highly improbable. Suppose I work my way through every step in a 20-step procedure and, in doing so, I achieve what I set out to achieve: reconcile a bank account, set preferences in Adobe Photoshop, replace a drive belt on a robot, or whatever. Can it make sense to say that although I understood each step—how could I not if I completed them?—I still didn't fully understand the procedure? I might not have understood *why* I had to do a particular step, but that is not the same as not understanding the step (for to

understand a step is to know how to do what it commands). In other words, to understand a procedure is to understand the steps in it, and when you understand all the steps in it you understand the procedure. There is no meaning that the map (in this case a procedure) has over and above the collective meanings of the constituent blocks (in this case, sentences). There is no semantic by-product, no epiphenomenon, no ghost in the machine.

So a procedure is comprehended once we've comprehended the steps in it. But what about conceptual material (such as the material you are reading now)? Might it be possible, say, for someone to understand every sentence in a multi-sentence paragraph but still fail to understand the paragraph?

In one sense this is possible, but that sense is irrelevant to our argument. If we are to keep the focus on the potential relationship between comprehension and *paragraph length*, we need to discard those reasons why a paragraph might be baffling that are obviously unrelated to paragraph length. For example, a paragraph that is poorly formed—one that is lacking the features readers customarily expect to find in paragraphs—can baffle a reader even if every sentence in it is understood perfectly. For example, I might find a paragraph baffling if:

- there is no topic sentence or no clear indication of what the paragraph is about
- some of the sentences in it seem unrelated (so that the paragraph lacks cohesion)
- the way sentences in it are linked is odd and distracting
- it contains a logical fallacy
- it contains sentences that contradict one another or contradict what was written elsewhere
- it is unclear whether the paragraph is about a new topic or continuing the discussion from the previous paragraph

But these are all faults with the way the paragraph has been constructed (faults in the sense that they do not give readers what they conventionally expect from a paragraph). Such faults lead to what might be called a *poorly formed* paragraph. On the other hand, a *well-formed* paragraph is one that meets readers' expectations: it has a topic sentence (or its purpose is clear), all sentences are related and appropriately linked, the logic is impeccable, and so on.

Now we can recast the question at hand as this: is it possible that, *in a well-formed paragraph*, you can understand the meaning of the sentences in it but still fail to understand the paragraph?

To answer that question in the affirmative is to assume the existence of some meaning over and above the aggregate meaning of the sentences. Let's call this epiphenomenon *residual meaning*. We can then ask how are we to detect residual meaning.

  
**CLARICOM**

**Make your Office Applications Sing!**

Frustrated by repetitive tasks in Office?  
Claricom customisations allow:

- ▶ One-click access to repetitive authoring tasks
- ▶ Automatically transferring information from one application to another
- ▶ Linking Office applications with databases
- ▶ Automated formatting and layout conversions
- ▶ Many more improvements (send us your ideas).

Web: [www.claricom.com.au](http://www.claricom.com.au)  
Email: [gary.calwell@claricom.com.au](mailto:gary.calwell@claricom.com.au)



For a start, it is not intuitively clear what residual meaning might be and how it might present itself. Perhaps it is akin to drawing a conclusion that was not stated from a set of premises that were stated. We read and comprehend the premises (the sentences) and then, by the exercise of our logical faculties, deduce the conclusion (the residual meaning). But that doesn't seem to match what occurs when we read. We do not appear to be inferring new meaning or new facts—meaning or facts unstated by the author—from every paragraph we read.

---

Contrary to the claims of Information Mapping, document-wide chunking is not a necessary pre-condition of comprehension.

---

Could residual meaning be tied up with the realisation a reader might have of how all the sentences in a paragraph fit together, contributing to the discussion of the subject that the topic sentence introduced? But if the writer has written a *well-formed* paragraph, the linking of sub-topics and the overall paragraph cohesion should be apparent to the reader. This is typically achieved through linking words. Common linking words or phrases are *therefore, so, hence, accordingly, it follows then, as a result, in contrast, at the same time, however, further* and *for the same reason*:

Doxycycline at 250 mg killed 90% of the bacteria. *In contrast*, penicillin at the same dosage killed at best 60%. *However*, penicillin at 500 mg killed all the bacteria. *Therefore*, our recommendation ...

A pronoun can also link one sentence with another.

Melanoma is a particularly aggressive form of skin cancer. *It* kills 2000 Australians every year.

Such linking is explicitly added by the writer. The paragraph cohesion subsequently attained—and noticed by the careful reader—is formed from the meanings of the sentences in the paragraph, including the linking words, pronouns and the like. It is not an epiphenomenon that arises as a by-product of something else that the writer has done (such as a conclusion that could have been drawn from premises but wasn't). The writer does it, the reader sees it—and sees it in the words that contribute to the meaning of the sentences. There is no spooky quantum-like phenomenon that mysteriously arises from the meanings of words that is not itself just the meanings of the words.

Could residual meaning be the mind's summary of what has just been read (or the knowledge that has been extracted from the words just read)? For a start, most of us don't summarise as we read. We might summarise if we were studying for an exam, but study is a minor part of our total reading life. When we read, say, an annual report, we do not read it with the expectation that we are going to be examined on parts of its contents. We don't even read it with the expectation that we need to remember what we have read. If perchance we needed to recall what was in a report, we can simply re-read the relevant parts.

There would be too much unnecessary effort—and subsequent cognitive clutter—if every time we read, we read in the hope of permanently remembering every fact.

Further, what might there be in a summary that is not also in the words being summarised? The knowledge we gain from reading is indeed separate from the sentences we read: the latter are aggregates of glyphs on paper or on a screen; the former a neurological loop in our long-term memory. But this existential independence doesn't mean that a

neurological loop is *semantically richer* than the sentences that sparked its existence. Indeed, it is likely to be semantically poorer. Otherwise we would all have perfect memories.

It is difficult to see, then, what residual meaning could possibly be. In informational writing (as opposed, say, to poetry), we do not write with the expectation that the reader will get more from what we've written than what the sentences we've written denote. Indeed, that could almost be a definition of *poor* informational writing. If a writer's words cannot tie down the meaning of an otherwise well-formed paragraph, then either the writer is inept or the act of written communication is everywhere and forever mired in ambiguity and vagueness. On the other hand, if a writer's words do tie down the meaning of a well-formed paragraph, then there is no need for the concept of residual memory. The words say it all.

Further, it is difficult to see how residual meaning might present itself, especially if it is tied to a chunking limit. Does such meaning appear at some stage up to the limit—but when?—and then attenuate or disappear once that limit is breached? Our reading experiences do not seem to match this model of a waxing and waning meaning.

The concept of residual meaning is looking somewhat elusive, more at home, perhaps, in the metaphysical than the physical realm. There are certainly no hints in Horn's work that might help us identify and detect this mysterious form of meaning. Perhaps, then, Ockhams Razor can legitimately be wielded: of two competing hypotheses, the one that requires the fewest assumptions is the one more deserving of belief. And that hypothesis is the one that does not rely on the existence of residual meaning in a well-formed informational paragraph. In other words, the better hypothesis is that, in the context of semantic chunking, the whole is no more than the sum of the parts.

Returning to our main point:

**Premise 1:** to understand a chunk is to understand the sentences that make up that chunk, and to understand the sentences that make up a chunk is to understand that chunk. And this applies



regardless of the chunk we are talking about, since our comprehension of any chunk can be reduced to our comprehension of the sentences in each of the lowest-level chunks.

**Premise 2:** readers can understand sentences regardless of how many sibling, parent and grandparent chunks there happens to be in the document they are reading.

**Conclusion:** Comprehension is entirely independent of chunk size. Thus the stress Information Mapping places on chunk size appears misplaced—

and would be so  
*regardless of the chunking  
limit proposed.*

Note that in claiming that to understand a paragraph is nothing more than to understand the sentences that compose it we are not saying that each sentence *on its own* is necessarily understandable. We often use pronouns in writing to avoid endlessly repeating a noun. A pronoun can refer to a noun in the same sentence or to a noun in a previous sentence. When it refers to a noun in a previous sentence, then what it is referring to will not be apparent to a reader unless they had also read the previous sentence. Thus the sentence *It kills 2000 Australians every year* will, if read in isolation, be difficult or impossible to understand. It needs to be read in conjunction with the previous sentence: *Melanoma is a particularly aggressive form of skin cancer*. However, our point is that the meanings of the sentences in a well-formed paragraph, *when taken together*, constitute the paragraph's meaning.

### Addition or diminution?

So far we have considered what might be *added* to our understanding by grouping a set of individually understood sentences into a paragraph. The answer appears to be nothing. If so, our understanding of a paragraph is nothing more than our understanding of the sentences from which it is composed, a conclusion that throws doubt on Horn's belief that comprehension requires chunking.

However, comprehension is sometimes measured by a reader's ability to *recall* what they have read (which, incidentally, takes us beyond the dictionary definition of *comprehension* as *understanding*). Perhaps what Horn had in mind is that chunking improves a reader's ability to correctly recall what they have read: the bigger the chunk they have to read, the more difficult it is for a reader to recall that chunk. This appears to match the folk wisdom that the longer the paragraph, the more difficult it is to understand.

So the fact that our understanding of a sentence is not impeded by the number of sibling or parent chunks might not be the full story. Perhaps our ability to recall the contents of a paragraph might

diminish as its size exceeds a certain limit (in which case, size would matter). Perhaps, then, we should be looking not at what is added as sentences accumulate in a paragraph, but what is lost.

But what does *recall* mean? In the experiments Horn bases Information Mapping on—the experiments reported by George Miller in 1956—recall meant *memorisation*: that is, *perfect* recall. But no reader is expected to memorise what they read. We noted this in the last issue of *Words* and noted, too, that this point is acknowledged on the main Information Mapping website. Thus we could accept

that recall becomes less accurate the longer the paragraph without accepting that this imposes any obligation on the writer to limit the lengths of paragraphs. If readers don't

read to memorise, then why should we tailor our writing in ways that might help them memorise?

However, some readers might want, at least some of the time, to remember what they read. They might not need to memorise material word-for-word, but they might want to be able to recall the salient facts that have been presented. Someone studying for an exam would fall into this group. Perhaps, then, writers should write in ways that meet the needs of such readers. And since we rarely if ever know whether a potential reader will want to *study* what we write, perhaps it is best simply to adopt an overarching principle of always writing in ways that meet the needs of the studious reader. Since the studious reader wants to maximise recall—at least recall of salient points—then perhaps we should be looking for a limit beyond which maximum recall of salient points begins to decline. In other words, although we might understand every sentence in, say, a fifteen-sentence paragraph, our ability to recall the salient points in it is compromised by its length being beyond what we might call the *salient recall chunking limit*. And, on the face of it, the *salient recall chunking limit* should be greater than any corresponding Miller-based limit (a limit that could be called the *perfect recall chunking limit*).

In the next issue of *Words*, I'll present research that shows that a salient recall chunking limit is impractically low. More importantly, the same research shows that it is *conceptual density* and *familiarity* more so than paragraph length that determines salient recall. I'll also present research that shows that our understanding of a paragraph appears unrelated to the number of sentences in it. The conclusion will be that paragraph chunking is important, but that the number of sentences in a paragraph should be a writer's least concern when deciding where to chunk. Other issues are far more important.

**Geoffrey Marnell**

# Technical writing with open source software: LaTeX, Subversion and Inkscape

James R. Hunt

Most technical writers work with proprietary tools: the overwhelming bulk of the world's software documentation is written in Microsoft Word, and much of the rest is written in Adobe FrameMaker. There are many technical writers who have used no other products than these. However, the proposition that Microsoft Word is not a good tool for technical writing is widely accepted, and the shortcomings of the product have been described in detail in many places—often with some passion.

I will argue that Microsoft Word can be replaced by a superior alternative, which is stable, bug-free for all practical purposes, and has been used by millions of writers over several decades. Furthermore, it is free, and in use involves much less effort than does Word. You can save time and aggravation for the rest of your life.

I will further describe in a little detail how this tool, in conjunction with a freeware version control system and a freeware vector drawing program, could be used to perform most of our documentation tasks more efficiently than is the case with our current proprietary tools. Unlike, say, fiction writers, technical writers are usually in control of a complete end-to-end process: we research our material, write it, edit it, design and check the formatting, and deliver documents to end users.

Are there any other writers like us? The answer is yes: engineers, physicists, mathematicians, and even computer scientists, who write journal articles, books and reports for publication. These writers are found largely, but not exclusively, in universities and research institutes. There may well be a few million of them—no-one knows for certain.

There is a difference: these writers do not use Word, but instead use a simple text markup language and typesetting program called *LaTeX*.

## How long has this been going on?

In the 1960s, the stone age of computing, the American mathematician Donald Knuth was dissatisfied with the appearance of one of his books. It had been typeset with the new technology of the time: a phototypesetter. He decided to take six months off to develop a computer-driven typesetter that could produce good-looking books and handle mathematics.

Ten years later, the project was completed. It was named *TeX* (pronounced *tech* or *tek*).

At that time, there were no computer fonts as we know them today, so Knuth designed a font called *Computer Modern*, which is still the default font used by TeX and its derivatives. Knuth also devised a font creation program called *Metafont*. However, font designers by and large found it too hard to use, and the *great font explosion* of the late twentieth century was driven by other, simpler, tools.

TeX was a low-level tagging language with about 600 commands. In the 1980s, the American computer scientist Leslie Lamport devised a set of macros based on TeX and named it *LaTeX*, almost certainly after himself. LaTeX commands were much easier for writers to use. They could now specify both a publication type (article, book or report) with predefined formats, and then the parts of the publication by commands: `\chapter`, `\section`, and so on.

LaTeX was extended to handle a number of PostScript fonts, in addition to Computer Modern. However, the developers of LaTeX did not foresee the advent of desktop computers stuffed with hundreds of fonts: that was someone else's vision.

TeX was originally designed to be completely self-contained and thus universal, able to run on any operating system, with the printed output always being exactly the same. However, this also meant that TeX (and LaTeX) had no access to system fonts.

In 2004, Jonathan Kew devised an extension of LaTeX that he called *XeTeX* (zee-), and this extension was able to access system fonts and also use Unicode fonts.

LaTeX is not the only macro set built from TeX commands. In the early 1990s, a system called *ConTeXt* was developed by the Dutch educational publisher Hans Hagen. LaTeX has a core-and-add-ons design whereas ConTeXt is monolithic, comprising several hundred high-level commands. ConTeXt is very powerful, and possibly has a great future, but you should explore LaTeX first.

## Is LaTeX useful to us?

Is it possible to learn from all of those LaTeX-using writers and build on their experience? Yes, and I will argue that we should do exactly that. LaTeX is not hard to learn: you can learn the basics in a day and become quite skilled in a week. The mastery so obtained is yours for a lifetime, because the core of LaTeX does not change. Books on LaTeX are often dauntingly thick. This is because LaTeX was developed for typesetting mathematics, and large



chunks of LaTeX books are devoted to that topic. We don't do much mathematical typesetting in technical writing. We don't need to learn about the complexities of academic referencing either, because most software and engineering user guides do not contain bibliographies and citations. When these items are left out of the average LaTeX book, the remainder comprises a discussion of about fifty tags —not many more than basic HTML. This is what can be learned in a day. Going beyond the default style options, which give very good results anyway, will take a few more days work. Authors of books on LaTeX are often apologetic about the “steep learning curve” involved, but it is steep only if you need to do some mathematical heavy-lifting.

## How to write in LaTeX

A word processor like Microsoft Word is a WYSIWYG system. This means that text is typeset as it is entered: the writing and typesetting tasks are conflated in a single window. This has the unfortunate result that the input file is continually modified by the typesetter, and errors in the typesetting process will instantly corrupt that input file. We have all seen this happen.

LaTeX does not work like this: the writing and typesetting stages are kept separate. Writers write in text editors, not word processors, and insert formatting tags (comprising a back-slash escape character and a string of letters) into their text. For example:

The charge was `\emph{grossly}` inflated  
would, when typeset, be rendered as:

The charge was *grossly* inflated

Text editors produce pure text files, with no hidden formatting information of any kind. When a typeset version of a file is required, the text file is sent to the background typesetting program, which reads the formatting tags in the text as commands and produces the required PDF output. The input files remain unchanged by this process. If the result is unsatisfactory, you will have your unmodified input text files for checking and editing.

In fact, one does not write in LaTeX at all: writing is done in a text editor, and typesetting is not done continually, but only when you think it necessary.

There are quite a few editors, and they are interchangeable because they all produce only text files. Most are freeware. Some are minimalist (such as *TeXShop*), and some have more buttons and control panels than a nuclear power station (for example, *texmaker*). Other common editors include *TeXworks*, *Kile*, and the prehistoric *emacs*.

Figure 1 shows the contents of a very short LaTeX file.

```
\documentclass[10pt, a4paper]{article}
\begin{document}
Type the instructions here!
\end{document}
```

Figure 1: A very simple LaTeX document

## There's a package for that...

The core of LaTeX is small and easy to learn, and if you want to do something outside the scope of the core, then you will need a package (that is, an add-on) designed to do that task. A package is a set of commands designed for a particular purpose. Quite a few packages are included in LaTeX distributions.

Having numerous feature-specific packages is both a strength and a weakness: a strength because your workspace is not cluttered with stuff you don't need, and a weakness because you may need research skills and time to find the package that solves your problem.

## Where to start?

The [Formatting Information](#) website provides a good short course in LaTeX. A [PDF version](#) is also available. It takes only a day or so to cover this material.

For instructions on downloading a LaTeX distribution, visit [Starting out with TeX, LaTeX, and friends](#). For a more substantial treatment, read [LaTeX](#), a Wikibook.

## A few MS Word problems solved by LaTeX

### Clickable links in indexes

In the index in a PDF version of a LaTeX document, the page numbers are clickable links. This is not the case in Microsoft Word.

### The PDF page-size problem

Reading an A4- or letter-sized PDF document on a computer screen can be annoying. It would be so much better if PDF documents came in two orientations: a portrait-oriented version for printing, and a landscape-oriented version for reading on screen. This is easy to do with LaTeX: just change a few specifications in the document (orientation, and perhaps margins and number of columns), and typeset it again.

### Hiding text

Many people have been caught out by the fact that hidden text in a Microsoft Word document can sometimes be revealed. Lines and blocks of text can be excluded from a PDF document generated by



LaTeX, in two ways: by putting a% tag at the beginning of every line or block of text that you want left out or by placing the text after the `\end{document}` tag.

## Master documents

When working with LaTeX, it is customary to keep separate chapters in separate text files, and set up a wrapper document for the book. Individual chapters are chained together by `\include` commands and processed together. The master document–subdocument approach is not something to be avoided: it is the best way to write books in LaTeX.

The set of instructions in figure 2 shows how this is done. It will produce a four-page document with a title page, a blank title page verso, a headed but empty table of contents, and a headed but empty index. (A log file will display three error messages, stating that the three required chapters could not be found.)

```
\documentclass[10pt, a4paper]{book}
\title{A Software Publication}
\author{}
\begin{document}
\maketitle
\tableofcontents
\include{chapter01}
\include{chapter02}
\include{chapter03}
\begin{theindex}
\end{theindex}
\end{document}
```

Figure 2: Chapters in a master document

## Typography and OpenType fonts

The function of typography is to improve the readability of texts, while not being obvious about it.

LaTeX uses a hyphenation and justification (H&J) algorithm that is much more sophisticated than that used in Microsoft Word, and produces better laid-out and more readable pages. Lines are not justified individually, as they are in Word. Instead, an algorithm is applied to a paragraph as a whole, in order to get the greatest evenness of word spacing over the paragraph. Hyphenation points are also decided by an algorithm. The end result is an electronic version of the decision-making of typesetters of past centuries, who set pages to get the greatest possible evenness of appearance.

XeTeX and the *fontspec* package allow the use of OpenType fonts, which have a number of interesting and useful typographical features. Quite a few OpenType fonts are free. Visit [FontShop](#) for more information.

## Subversion and writing projects

A version control system is a software product that is used to track changes to documents over time. Version control systems are a vital part of any software development project, where the documents being tracked may be text files containing computer code, or complex LaTeX documents and their component parts. Version control systems work best with text files: they do not work at all well with the binary files produced by word processors. Further, a version control system can be used as a collaborative tool. There are many version control systems, and one of the most popular is a freeware product called *Subversion*.

### Basic concepts

In Subversion, a directory and its contents are stored in a special area called a repository. A database management system maintains records of changes made to the directory and its contents. Ideally, the Subversion repository should be placed on an external server, but it can reside on your own computer if required.

There are a number of graphic front-ends for Subversion: *TortoiseSVN* (for Windows only) and *RapidSVN* (for all platforms) are the commonest, and are freeware. All of the operations mentioned here can be carried out using these front-ends.

Subversion uses a centralised model of version control. There is a central repository from which a local working copy can be checked out. (Checking out a working copy needs to be done only once.) The working copy comprises a directory and its components, which may include subdirectories and constituent files. You can make whatever changes you require to the working copy. Once you are satisfied with the changes, you then commit these changes back to the repository.

### Working with Subversion

There are four basic steps in working with Subversion, which should be carried out every day.

#### Updating

You should always begin by updating your working copy. This brings into your working copy changes in your area of the Subversion repository that were made by other writers (if any).

#### Changing

There are two kinds of changes that can be made in the working copy: changes to individual files, and changes to the structure of the directory. Changes to individual files are made in the usual way. There is no need to use any Subversion commands. Changes to the structure of the directory, however, will require the use of Subversion commands.



## Reviewing

Before committing your changes, it is a good idea to review the changes you have made. Subversion has some useful tools for doing this.

## Committing

If you are satisfied with your changes, it is time to commit them to the Subversion repository. When a change is committed, the Subversion database records four important pieces of information:

- who committed the change
- what the change was (a description of the change)
- when the change was committed
- where the change was made (which files or subdirectories were changed)

There are two aspects of committing that require discipline. First, it is easy to be lazy, because there is nothing forcing you to commit your changes. You should commit often, because the more commits you make, the more detailed is the record you have of the changes that you made. In particular, you should commit before deleting anything—just in case you want to use the material you are thinking about deleting at a later time.

Second, when you commit, you will be invited to enter a commit message. If you commit changes to multiple files, your message should detail the changes to each of these. This may at first seem onerous, but a trail of detailed commit messages can be useful for reviewing progress.

## The *svn-multi* package

If you are keeping your LaTeX document in a Subversion repository, it could be useful to display information about the current revision somewhere within the PDF version of that document: for example, in the footers of chapters.

The *svn-multi* package obtains revision information from the Subversion repository, and provides commands to display this information in the LaTeX document.

## Document reviews

There are two ways to allow reviewers to comment on drafts: allow them to have full access to the Subversion repository, which may not be practical, or give them PDF versions into which they can insert comments. The latter procedure is usually preferable. There are a number of PDF annotation tools available for this task.

The commonest PDF reader, *Adobe Reader*, cannot be used to insert comments into PDF documents generated by LaTeX. *Adobe Acrobat* can insert comments, but that product is not freeware.

*Foxit Reader* for Windows is freeware, and can be used to annotate PDF documents generated by LaTeX. The PDF viewer supplied with OS X, *Preview*, contains annotation tools.

## Illustrations

Many software manuals are illustrated solely by screenshots. Engineering manuals may be illustrated by drawings of considerable complexity.

## Screenshots

Screenshots in various formats—including PNG, JPG, and PDF—can be included by reference in LaTeX documents. Quite a few free screen capture programs are available: you probably use one already.

## Inkscape

The open-source scalable vector graphics editor *Inkscape* is a tool for creating and manipulating shapes, both geometric and free-form, as well as lines, text, colours and gradient fills. Inkscape has many of the capabilities of Adobe Illustrator.

Scalable Vector Graphics (SVG) are produced by a text-based graphics language that describes images with vector properties, embedded raster graphics and text. Inkscape illustrations can be saved in SVG format.

## Including an SVG image in a LaTeX document

You can export an SVG image from Inkscape as a PDF file with a LaTeX wrapper around it. When this combination is included by reference in a LaTeX document, any text included in the image will be typeset in the same font as the rest of the document. If you change the document font, the font used in the diagrams will change as well. All the text in the image will be processed by LaTeX, and so the diagram may contain custom LaTeX tags. This feature would be useful in managing translations of labels in drawings (discussed on page 14).

## Online help systems

Because there has never been a need for them in the scientific world, there are no LaTeX-based tools for writing context-sensitive help systems of the kind produced by *RoboHelp*, *Authorit*, *Flare*, *Help & Manual*, and the like. These products are designed to work with Microsoft Word: specifically, they will accept as input a Microsoft RTF file, process it and display the results in a word-processor-like window.

Accepting RTF input is a useful feature of commercial help-writing tools, because many technical writers produce a first draft of the online help system by importing a draft of the user manual after having written it in Microsoft Word. From that point, the online help and the user manual diverge—

the online help system usually has a reduced text and a more complicated navigation system than does the print form of the user manual.

None of the commercial help-writing tools accept LaTeX-coded input. All is not lost, because there is a utility for converting a LaTeX file to Microsoft RTF: *tex2rtf*. The output of this utility can be used as input to a help-writing tool.

## Examples of projects

When we consider writing projects of various sizes—from one-writer projects to giant projects involving many writers in several locations—it becomes clear that Microsoft Word is not a scalable solution. In other words, the difficulty of managing the project increases much more rapidly than does the length of the manuals or the number of writers. Long Microsoft Word documents are notoriously unstable, and the lack of management tools means that a project with only a few writers can easily run out of control. The problems of *ad hoc* formatting are well known. There are just too many ways in which things can go wrong, and too many details to manage.

Can these problems be avoided by using the tools described here? The answer is yes.

Before continuing, a note of caution. When managing projects with several writers, it would be all too easy to replicate those giant XML- and SGML-based writing and publishing systems beloved by defence bureaucracies and their dependent manufacturers. Those systems present writers with a fixed set of tags and allow writers no control at all over the appearance of the final publication. Many writers are quite happy working with systems like these. However, to others, using these systems involves working conditions akin to that of the Israelites in Egypt: in slavery, endlessly making bricks in an obscure corner of the military-industrial complex.

### Several writers working on a single book

Let us suppose that you are the lead writer on an enormous software project, and that you have assigned books or different chapters of various books to several writers. Each writer writes in a text editor. At regular intervals they test their work by making sure that it can compile to PDF without errors. They then commit the text files (not the PDF files, which are only temporary) to a Subversion repository. You can then extract copies of the various updated files from the Subversion repository and compile the full PDF versions of the various manuals. This could even be automated as part of the daily build process used in developing the software product.

## Managing translations

Your company has decided to localise its software product and sell it from Iceland to China with local-language manuals. You are in charge of preparing those manuals. How would you go about it?

First of all, the text strings displayed in the windows of a modern software product are stored in one file. When they are to be displayed, they are imported by reference. To produce other-language versions of the software, new files of text strings must be prepared by translators. Further, a user must be able to select a language and store it as a preference. Thus it is possible to produce a great many different language versions of the software in a reasonably short time. But producing the matching manuals will be more complicated.

If your base-language manual includes screenshots, you will have to capture the corresponding screenshots for all the other language versions. Name the new screenshots systematically, and include a language code within the filename, for example:

screenshot\_01\_de.png

screenshot\_19\_fi.png

Send copies of the base-language text files (for version 1.0, say) to the chosen translation houses. The translators will not need to change the tags, only the text: LaTeX tags do not change with the text language. Eventually, you will get back sets of text files of the translations. Change the language tags in the screenshot filenames, to make sure that the screen-shots all go into the right manuals.

Insert a tag specifying the language at the start of each master document (to ensure that the correct hyphenation algorithm is used for each language) and compile the PDF versions. You will then have a collection of elegantly typeset manuals, quite a few of which you probably cannot read.

Producing the first translations of a manual set is hugely expensive. However, when it is time to produce the various translations of the version 1.1 manual, the cost will be much lower. The aim is not to translate anything that was translated before, and translate only the changes made in the base-language version.

The changes between versions can be found by using Subversion tools. The translators can find the matching lines in the foreign-language text files, make the required changes, and save the modified files, with an incremented version number, for each of the various languages.

## Post-amble

Much software and engineering documentation is produced using unsuitable tools, and the results are often not commensurate with the effort involved. We really need better writing, illustrating and project control tools, and, fortunately, excellent freeware products are available for these tasks.

LaTeX is a stable program, proven in use by many writers in scientific and technical fields. It is easy to

learn, and the core of the program has not changed for many years. LaTeX, Subversion, Inkscape and PDF document annotation software can be used together in documentation projects of considerable complexity. There is a strong case for adopting them as our new toolkit.

### James Hunt

James Hunt is a Brisbane-based technical writer.

## Fresh reflections on the technical writing profession

Five newcomers to the field of technical writing in Australia were asked to write of their first impressions of their new profession. Their candid responses follow. The *Words* team thanks them for their time and honesty.

### Jonathan Brent

If I believed in omens, my first day of professional technical writing would have been a little bit unnerving. After being ushered around the office, where I shook many hands and desperately tried to retain as many names as I could, I was shown to my desk.

The view from my desk: a tombstone factory. In terms of raw symbolism, it stood in stark contrast with the birth of my potential career as a technical writer.

But, with 12 months of incident-free technical writing under my belt, it is safe to say that there was nothing to worry about. With this in mind, I would like to share with you the following reflections on the profession.

#### The Devil is in the detail

It perhaps goes without saying that attention to detail is critical to good technical communication. There were a few moments during my last contract when this point was thrown into sharp relief.

The most dramatic close-call became apparent when we noticed an issue with one of our documents. The document in question was a guide containing instructional material for a tertiary course. The code for the guide's corresponding TAFE unit had been mistyped. In fact, it had been mistyped in 36-point text, on the cover of the guide.

For a moment, I imagined the fate of the poor misnamed guide had it been released into the world—one incorrect keystroke cleaving it from its kin, wandering aimlessly, unrecognised by educational bureaucracy.

Our other brush with mishap was more amusing. The offending term came to light while we were preparing a document for final review by the technical advisory committees: “exophthalmic”.

“Exophthalmic” had come from one of the resources that we were using to build up the guide in question. And, because it is, in fact, a real word, the word processor never highlighted it as a typo and it survived the editing process until the 11th hour.

The word that we really wanted was *exothermic*, referring to a chemical reaction that releases energy in the form of heat.

The word that we nearly committed to print, it turned out, had the following senses:

1. Of, or relating to exophthalmos (an abnormal protrusion of the eyeball from its socket).
2. Having prominent eyeballs.

The two words were clearly not interchangeable. Accordingly, the guide went under the knife, and the exophthalmos was addressed.

#### SME's just not that into you

Subject matter experts are indispensable, but they certainly are a unique breed. Under the right circumstances, they are unstoppably productive.

Under normal circumstances, they are totally feline in their aloofness.

In my experience, many conversations about SMEs have ended up at the point where they sound like a

discussion of failed courting.

‘Still no call from Gary.’

‘I know, right? I got the “Don’t call me; I’ll call you” email from Pat.’

‘I really don’t think Gary is into this.’

Of course, this is in no way meant to disparage the good work of the many selfless SMEs who give up their weekends and evenings to furnish us with their

---

Subject matter experts are indispensable, but they certainly are a unique breed. Under the right circumstances, they are unstoppably productive. Under normal circumstances, they are totally feline in their aloofness.

---

know-how. It is, however, best to be prepared for material to arrive at the 11th hour, and, sometimes, the 13th.

## On SMeEnglish

Of course, should you actually manage to tap a rich vein of SMeExpertise, there are certainly no guarantees that the information that you receive will be rendered in human language.

At times, there appears to be an inverse relationship between the volume of data you receive from a subject matter expert and its legibility.

At its most extreme, SMeEnglish seems to occupy some kind of space outside the realm of syntax. Spelling? An extraneous luxury. In these particularly pronounced cases, it can be a challenging lateral-thinking exercise just to extract some meaning from the prose.

There are undoubtedly some tricky linguistic riddles to untangle out there. But, in my experience they are rarely insoluble, and sometimes all it takes is fresh pair of eyes to help resolve them.

But, then, if the archetypal SME were as passionate about expression as they are about subject matter, we would all be out of a job. I suppose, in the end, it is a beautiful symbiosis.

## Naomi Grant

After ten years working as an office manager, I decided to return to study and complete the Postgraduate Diploma of Editing and Communication at Melbourne University. As an office manager, I was the primary go-to person for all general staff needs. I enjoyed the role of problem solver, although I found some challenges more interesting than others. The replacement of an expired biro was a little less rewarding than the documentation of work practices and procedures.

My office management experience had included the application for, and maintenance of, ISO:9001 accreditation. This experience made me appreciate the value of good document management. It also made me aware of the resources required to maintain these systems. I had worked primarily within the health sector, which ensured an allocation of resources for the maintenance of quality systems, but which still had to operate within budgetary limitations. I had been employed by a couple of different employers and within different departments, which meant that I did not have long-term experience with a single document management system.

During the first semester of my course, I had the opportunity to complete an internship with a book publisher. I soon realised that my initial expectations

were wide of the mark. In my role as office manager, I had been responsible for administering the work of others. I was disappointed to realise that the general editing department within this publishing house operated in much the same way. Their primary role was to manage the writing, editing and production of work that was outsourced to external parties. It was this experience that sparked my interest in the technical writing subject and the opportunity to create rather than administer material.

At the conclusion of my course, I was fortunate to gain an entry-level technical writing role. I was excited about learning new skills and acquiring new knowledge. The prospect of developing content was

a little daunting initially but the Australian Standards provided a framework for much of the material. I was also fortunate that my employer had developed a

comprehensive template, which meant that many of the decisions had already been made. I found the review process interesting and it reinforced for me the value of developing material through drafts.

My next contract was a more autonomous role but within a larger team. The project had already been running for a couple of years when I came on board. It was interesting to observe how the material had evolved as the project had grown. I came to appreciate the challenge of planning for a project when the scope of the product is not fully realised at the outset. It also reinforced the value of a style guide and thesaurus, particularly when working with a larger team and project. I realised that having certain decisions signed-off can save a lot of time in production and review, particularly when larger numbers of people are involved.

I am yet to experience the delivery of a product I have produced. As a result of my experience with quality systems, I wonder about the maintenance of the product after delivery. I know from experience how difficult it is to get staff to comply with a quality system, even if adequate resources have been allocated. I am also curious about access to the information for the end user. From my limited experience, the staff responsible for maintaining and providing access to information have operated independently of the technical writing team. I am looking forward to future experiences that might provide greater insight into the life of a technical writing product after delivery.

## Bloss Oliver-Skuse

A little over a year ago, I traded the satisfying pedantry of wrangling someone else's writing into something like recognisable English (the job description said "editor"), for the equally pedantic pleasures of technical writing. I hadn't heard of technical writing until I enrolled in the subject at uni

---

There appears to be an inverse relationship between the volume of data you receive from a subject matter expert and its legibility.

---



to fill an elective block. Embarrassingly enough, I arrived to the first class of semester and asked if I was in the right room for “the scientific editing subject”. Despite the inauspicious beginning, technical writing struck a chord for me.

I was lucky enough to have the opportunity to move straight into the industry, working as part of a small team of writers producing fire protection training materials. I had left uni with utopian visions of rigidly enforced project-specific thesauruses, teams of highly cooperative subject matter experts and, above all, unlimited access to examples of the fire protection equipment I was to write about. I quickly learnt that some projects evolve in a way that doesn’t allow for these luxuries.

---

*I’m not sure if having a lobe of my brain dedicated to thinking (and writing) like my project manager is a good skill for a technical writer or the sign of an incipient personality disorder.*

---

Trying to keep vocab and writing style consistent across five writers without a project-specific thesaurus has taught me to foresee how text will be edited and to write in someone else’s style. At this point, however, I’m not sure if having a lobe of my brain dedicated to thinking (and writing) like my project manager is a good skill for a technical writer or the sign of an incipient personality disorder.

After being in an editorial department, the review process often endured by technical writers came as a shock. In my experience, the willingness of subject matter experts to provide feedback is usually inverse to our desire to have it; when we want it most—after the draft is finished and before the deadline—they seem least keen to give it to us. However, they are often more than generous after our final deadline has passed. SME reviews, when we are lucky enough to receive them, highlight why engineers seldom make good instructional writers. Many comments consist of a group of miscellaneous noun phrases, linked only by their proximity on the page, disguised as a sentence by means of a full stop at the end.

However, as difficult as it can be to try and wring some sense out of these reviews, it is a good affirmation that technical writers are needed. Likewise, getting the final approval from the SMEs reassures me that effective materials can be written by someone with no previous knowledge of the area (who hasn’t even seen a gaseous fire suppression system).

## Shilpa Shankar

I was a technical writer for about five years back in south India, where I hail from. My nit-picking and obsessive-compulsive nature, microscopic eye for detail, maniacal sense of organisation and innate finickiness made me ideally suited to being a technical writer and editor. It also helped form an

awfully wry sense of humour! (My favourite’s always been: “Don’t be afraid to play with the computer. It won’t bite you. You can use a mouse (one without fur) in conjunction with the window system of your computer.”<sup>1</sup>)

Back in India, most IT companies (they were the ones that largely employed people like me), seemed to be saturated with tech writers who stumbled into tech writing because they didn’t make it as developers, testers or PCB designers. With a fair command of the language and a passion for technology, they became tech writers. Most tech writing teams in most companies preferred their writers to become SMEs, each sticking with and growing in a particular technical domain. The domain foisted on me was mainframes! Yes, the archaic, but extremely robust, green-screen monsters of the post war era. This move was not in tandem with my own vision and I grew crankier by the minute. I knew time had come to review my career path and make the break from tech writing, moving perhaps into a sister-concern like e-Learning, instructional design, corporate communication, marketing communications, learning and development, training, and the like.

Still weighing the pros and cons of steering away from the only trade I knew, but daring to plunge head-on into the abyss of the unknown, I decided on the quasi-safer mid-way path: take a break from the workforce and re-enter the ivory towers of academia, overseas, far from my comfort zone, away from the madding crowd. I moved lock, stock and barrel, in true Noah’s ark-like fashion (just one suitcase, a modest laptop, a pair of oars and a rickety raft) and set sail for sunny Melbourne, seeking higher education and enlightenment. I chose Australia because I thought everything Australian had to involve the sun, beer and a BBQ: a wonderful combination.

A lot changed since then, including gaining a two-year Masters in Editing, Publishing and Communications at Melbourne University. Unsure of my subsequent employment prospects, I deliberately took an esoteric inter-disciplinary degree with such subjects as Print Production and Design, Ethical and Legal Issues in Publishing, Writing and Editing for the Digital Media, Database Systems and Information Modelling, and Interaction Design and Usability. I also took Technical Writing and Editing, primarily to get a sense of the local tech writing community and industry should I decide to return to tech writing after all. I was fairly sure that I wasn’t—but I’d give it one last shot. A safe buffer. I’m mighty glad I did, because as it happens, I am now a tech writer in Melbourne, and this is likely to have helped me get my temporary residency. And

---

1. *Read Me First!: A Style Guide for the Computer Industry*, 2nd edn, Prentice Hall, 2003, “Chapter 3. Writing Style”).

although I'm still upset at discovering that the Department of Immigration has removed tech writing from the desired-skills list, making me now ineligible to apply for permanent residency in this lucky country, I would like to stick on a little longer as a tech writer here.

My first impressions? People here respond much as they do in India when I tell them that I'm a tech writer by profession. Yes, I still have to use an analogy to explain my trade! I think I have met only two non-technical-writers who knew exactly what tech writing is. So I'm no longer as surprised that most people are clueless about the trade and profession of tech writing. But that's the general population. In more esoteric circles, most people know of the trade and its tradesmen and women.

---

What I've found most heartening and promising is that tech writing in Australia is not viewed in isolation, as an adjunct to other more important activities. The documentation life cycle and the documentation team are considered essential components of the product development process and are well-integrated into it.

---

I have been over-joyed to find kindred spirits among fellow tech writers in Australia. They know what they love and love what they know and don't know. They love to design and write procedures, irrespective of the subject matter, style, form and format. They are sensitive to the profiles and demographics of users, open to adopting a more contemporary English language and yet are as pedantic about consistency in style and usage as I am. But more than anything else, what I've found most heartening and promising is that tech writing in Australia is not viewed in isolation, as an after-thought or adjunct to other more important activities and departments. The documentation life cycle and the documentation team are considered essential components of the product development process and are well-integrated into it. Inter-team liaison and collaboration is perhaps the healthiest and most attractive aspect of being a tech writer in Australia.

On the down-side, I do wish there was more social interaction among the tech writing community in Australia. A national forum or body for instructional designers, trainers and tech writers would be an invaluable resource for the community of professionals that roughly falls under the umbrella of instructional writers.

I'm grateful for the life-changing two years I've had here. Not only have my pre-conceived notions about sunny Australia filled with marsupials and tanned suffers changed dramatically, replaced by a more realistic perspective that includes on-going dilemmas over immigration, asylum seekers, carbon-tax, uranium exports, AFL grand final deadlocks and replays—but I've also regained my love for tech writing.

## Martin Wojczyns

It was the tail-end of a university group project, and all the members of my group had worked hard to research and write their contributions for the project's final report. All that was left for us to do was to sew these contributing pieces together and then to edit them to make them read like one cohesive body of work.

'How do we want to put it all together?' one of my colleagues asked. I noticed that everyone averted their eyes. I had noticed this reaction quite often in previous group projects: the general consensus was that the editing was a chore, a necessary evil at the end of a project.

But that's not the way I saw it.

'I'll do it!' I eagerly offered. The expressions on my colleagues' faces told me that they thought I was crazy for being so keen to volunteer for this task, but that, at the same time, they were grateful that they weren't lumped with this apparently boring job.

But I don't find this type of work boring, I had noticed that whenever I was working on an assignment, it wasn't only *what* I was communicating that interested me, but also *how* I communicated it. I enjoyed reading and writing about the content, but something I enjoyed just as much, if not more, was ... well, everything else that goes on *around* the content: from macro considerations like overall layout, right down to micro considerations like the best word to use to convey the right meaning, or the judicious placement of a comma.

The placement of a comma might seem like a trivial thing to get excited about, but I remember finding these kinds of things so much fun that, when I wasn't short of time, I sometimes purposefully declined using tools like *EndNote* that automated the process of constructing bibliographies, because it meant that I could manually create the bibliography entries from scratch according to the desired reference style. OK, even I will admit that that is a little bit crazy!

But what fascinated me about considerations such as the placement of commas was how such small things, in combination with the big picture, can affect overall comprehension. I loved thinking about the best way to clearly communicate information and make it look good along the way so that it is easily and pleasantly accessible to my audience.

So you can imagine my joy when, later in my university life, I discovered that there was a profession that required its practitioners to do just that. This profession, of course, was technical writing! That's right—apparently there would be someone willing to pay me to have fun constructing and reconstructing documents to read as well as possible. I was aware of and enjoy other forms of editorial work, but what seemed to draw me to

technical writing was something about procedural-based writing and the fact that the entire purpose of technical writing is to help people: the company the technical writer works for, and the company's customers. These aspects of technical writing seemed to add weight to the necessity for the writing to be clear.

Enrolling in a technical writing course at university soon convinced me that this profession really was for me and really was as good as it sounded. Before long, it was time to put what I learnt into practice in the form of an internship. It wasn't just working as a technical writer that would be a new experience for me, but also working in an office environment. Admittedly, I always held the naïve preconception that working in an office would be a very formal, cold, solitary experience and that the people would be all business: more robot than human.

Maybe some companies are like that, but not the one that I interned at. That's not to say that the professionals I worked with didn't work hard, but rather that they worked hard and got the job done to an impressive standard and, at the same time, managed to have fun and be a pleasure to work with. From the first day of my internship, my preconception of a lifeless office environment was shattered as I discovered that the work that I would be doing would only make up one half of the enjoyment that I would get from being a technical writer; the other half would come from the people I would be working with.

I was very fortunate to have my internship turn into a series of contracts, and I quickly discovered that technical writing doesn't just give its practitioners skills for writing high-quality user guides, but it also gives its practitioners skills that can be applied to all forms of communication. For example, I created information movies that, although not procedural, required the types of considerations that must be kept in mind when creating a procedure. For instance, I had to keep audience, tone, clarity and familiarity in mind when creating the movies. There were also additional considerations in the movies that aren't relevant to other forms of technical writing, like manuals, such as the narrator's accent, which had to be considered to achieve maximum familiarity for the audience. I was quickly realising that the things under the banner of technical writing were much more far-reaching than I had thought. This realisation would be reinforced in my next position in another organisation. In this position I learnt about a specialised form of technical writing: tender writing. But I also learnt that a technical writer

can excel at all tasks that require professional writing, editing and layout skills. These tasks can vary widely, from creating forms to writing newsletters. Again, technical writing skills are perfectly suited to a huge variety of tasks. There is no chance of getting bored.

I felt lucky to even have had the opportunity to get this latest position, because when the time came to find a permanent position (which is the type of work that I felt would be the best learning experience for me at this stage), I was quickly reminded that when searching for work as a young technical writer, at this early stage it's almost guaranteed that no matter how much I practised writing at university, no matter how enthusiastic I am, there is a high chance that another candidate going for the same job as me is going to have more experience than me, and therefore I will never be the successful applicant.

This was the pattern that repeated in most of my technical writing job applications. I knew that I was on the right track because most of the positions I applied for asked me to come in for an interview, and most of the interviews felt very successful. But I never quite made it all the way through the application processes. So I felt very fortunate that this one particular company was willing to take on a relative junior to the industry.

One thing that I realised helps very much when looking for your first permanent technical writing job is the kindness of others who are willing to give you a chance. Not just the person who offers you a position, but the people who come along earlier in the process: the people who let you know about relevant opportunities and that, even earlier, let you learn and get experience so that you can put together a portfolio of work. It would certainly be a more difficult without the help of these people.

Now when someone asks me what my profession is and I explain that I write user guides and other documentation for a living, I sometimes experience similar reactions to the ones I got from my university colleagues when I expressed an eagerness to take on editing tasks; I can tell from the expression on their face that they are wondering why I would choose to do something as boring as writing a user manual. But that's the thing — to me it's not boring. It's a fun task.

But that's not the only reason why I love technical writing. It seems to me that it has a nobler cause than, say, writing a novel or a screenplay. As fun and rewarding as other types of writing can be, technical writers have the goal of helping people — helping clients and customers save time and hopefully make

---

What fascinated me was how such small things like comma placement, in combination with the big picture, can affect overall comprehension.

---

---

Technical writing skills are perfectly suited to a huge variety of tasks. There is no chance of getting bored!

---



their days smoother, and help the company that I'm writing for by improving customer satisfaction.

When I first learnt of technical writing, I thought it was too good to be true. But it turns out to really be as good as it sounded. Technical writing may not be

for everyone, but for the people it suits, it really is a fascinating profession. I feel that I've learnt much in a very short space of time and I feel that I'm still learning every day, and I look forward to seeing what technical writing has to teach me in the future.

## Journal of Technical Writing and Communication

### Contents of the current issue

Click the heading below to read the editorial, abstracts and reviews, or to purchase an article or subscribe.

#### Volume 41, Issue 3, 2011

- Technical Writing and the Development of the English Paragraph: 1473–1700, Elizabeth Tebeaux
- Practicing "Safe" Technical Communication, Paul M. Dombrowski
- Design and Usability: Beginner Interactions with Complex Software, Michael J. Albers
- Visualizing Banking and Financial Products: A Comparative Study of Chinese and American Practices, Han Yu
- Emphasizing Research (Further) in Undergraduate Technical Communication Curricula: Involving Undergraduate Students with an Academic Journal's Publication and Management, Julie Dyke Ford and Julianne Newmark
- Autobiographical Writing in the Technical Writing Course, Mark Gellis

award-winning, authoritative international voice, publishing the latest research by recognized scholars from around the globe



Every article ever published in the JTWC is now available in clear, concise, comprehensive PDF format.

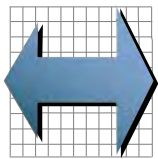
**New Online-Only Delivery Options**

- Enhanced subscription (all articles from volume 1 through current volume)
- Current volume subscription
- Individual articles on pay-per-view basis

**Click here for all the details and ordering information.**

- Read all article abstracts free
- Download a complimentary sample issue

**BAYWOOD PUBLISHING COMPANY, INC.**  
<http://baywood.com>



**Australian Society for Technical Communication (NSW) Inc.**

PO Box R812  
Royal Exchange NSW 1225  
Phone 8250 7056  
Fax 9252 0441  
email [info@astcns.org.au](mailto:info@astcns.org.au)

### Conference 2011 — More than words

Friday–Saturday, 28–29 October 2011  
Citigate Central Hotel, Thomas Street Haymarket

*More than words* investigates and celebrates the things that technical communicators do beyond typing words on a keyboard – the value we add through project management, illustration, information management and delivery, and customer relationships.

To join the Australian Society for Technical Communication, visit [www.astcns.org.au](http://www.astcns.org.au) or [www.astcvic.org.au](http://www.astcvic.org.au)



## Book review

*Teaching Intercultural Rhetoric and Technical Communication: Theories, Curriculum, Pedagogies and Practices*, Barry Thatcher and Kirk St Amant, Baywood Publishing, 2011

Reviewed by Vian Lawson

This book, part of the Technical Communications Series published by Baywood, is intended for teachers of technical and instructional writers. It contains thirteen articles, some of which have appeared in the *Journal of Business Communication*, a quarterly publication of the Association of Business Communication. As the title suggests, it is intended for, and marketed towards, those who teach business or technical communication in tertiary courses in the USA, and is divided into four sections which examine aspects of intercultural interaction in the classroom, and in the business world. Although it is intended primarily for educators, there is a good deal to interest technical communicators, particularly those who work in a multicultural or multinational environment, or who must internationalise or localise documentation. It is written primarily for the American audience, although there are articles which deal with technical writing and practice in Europe, India and New Zealand.

Technical communication as a profession has grown on two fronts in the last fifty years. First, it is increasingly recognised as a set of specific skills, and demand for those skills has increased globally. This has seen the rise of vocationally oriented technical communication courses, particularly in the US, but more and more outside it. Second, technical writing was once tied to the manufacture and maintenance of physical products. But with the rise of online media and the growth of the information economy, technical communication is increasingly being applied to services and processes as well as physical products. In short, more people are writing technical content, and more people are reading it.

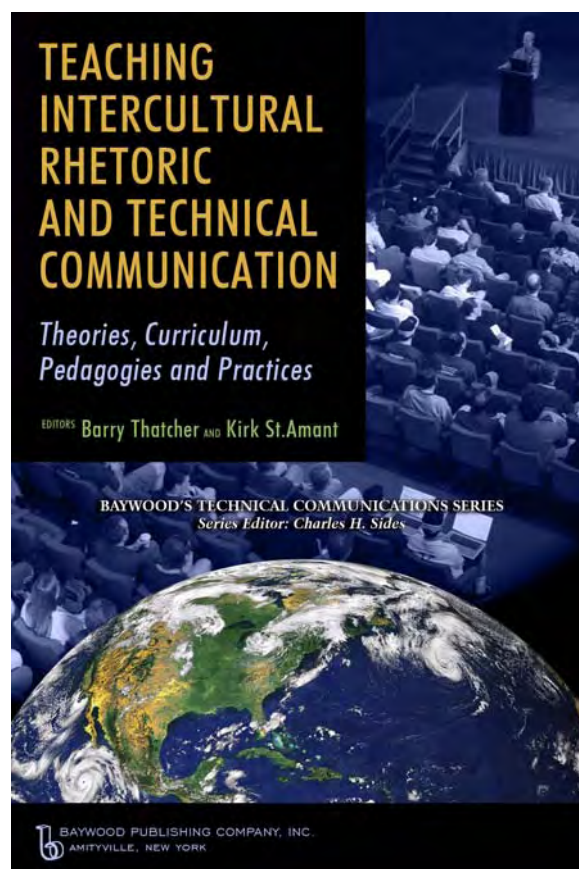
In the twentieth century, a technical writer would usually work in the same location as the designers, manufacturers and end users of the product for which they were developing documentation. Today, however, products are developed by teams working all over the world, and global distribution networks mean that the potential users of a product may be on a different continent to the people who designed and made it. In order to work effectively, technical writers must therefore be able to work with people from a range of cultures, and must also be able to develop materials which can be localised or internationalised quickly and easily.

Working in a global industry requires that a writer must be able to recognise and negotiate cultural differences, develop information products which are culturally appropriate and adapt to methods and customs not used in their local

environment. American (or Western) work habits, formats, approaches to information design, and even American English, are often inappropriate in other parts of the world. This apparently comes as a shock to many American students, many of whom have never travelled outside the US, and most of whom are predisposed to judge cultural difference as inferiority, if they see them at all. Perhaps America's recent global and financial problems have caused its educators to question its cultural supremacy. Whatever the cause, this move towards recognition of cultural diversity is promising.

The aims of integrating cultural-awareness training into classes, then, are to make students aware of cultural difference, and then to teach them that the world is full of groups who go about successfully using values, principles and world views that are quite different from theirs. Teaching cultural relativism involves getting students to empathise with members of a culture not their own, even if they can not sympathise with it: adapting to the fluid concept of time in India, for example, or the gender politics of Japan, or the differences between high-concept and low-concept societies without judging them.

The best way to teach students to perceive these cultural differences is by immersion in another culture. Many universities have overseas posts these days, and a fortnight in Italy or India, not as a tourist



but as an analyst of local customs and communication practices, is a breakthrough experience for many a young writer. The last third of the book details many teaching experiments on foreign campuses of US universities, where students are encouraged to look at materials like tourist brochures not as consumers, but as potential developers and producers. European students are at a definite advantage here. The geography is sympathetic, and the EU has structures in place to encourage student exchanges and internships. Australians are at something of a disadvantage. Our students must rely more heavily on local multiculturalism, and on the Web, for exposure to non-local content.

As you would expect, many of the articles recommend Plain English principles, and suggest graphical content as a way to bridge the language gap in high-concept societies where English is not the first language of the intended audience but is the language of the writer. But a more intriguing possibility is offered in a few articles dealing with the challenges of globalisation. Emily Thrush and Angela Thevenot, in a section rightly titled *A Radical Proposal*, suggest that if you are developing documents for an audience which does not speak Standard English, you should incorporate the non-standard forms in your documentation: adopting the written equivalent of an accent, or accepting the accent of non-native speakers of English who, for example, don't invert verb order when asking a question ("What you are seeing now?"). They argue that just as we adapt to vocal accents, we must learn to adapt to their written equivalents. How one non-native speaker on the factory floor is supposed to adapt to the written accent of another non-native speaker without a baseline English as common ground is left as an exercise for the reader. Thrush and Thevenot's research is based on classroom exercises with non-native speakers who have all passed the TOEFL to get into an American university, and who therefore have at least some common expectations of English. I respectfully submit that the globe is not ready for quite that much cultural relativism, and understanding a plethora of written accents is hard enough for native speakers, but would become a Herculean task for different non-native speakers.

Still, the idea that Western (US) standard written English is the equivalent of BBC received pronunciation is a useful one. Acknowledging that there is no One True English can only serve to make us more careful, more responsive to local usage and idiom and more effective communicators. Many Australian technical writers adapt their English

already, of course, when writing for audiences with no post-secondary education, or for whom English is a second or third language—the idea of an accent in written English is in many ways an extension of this practice.

The perils of cultural relativism are not confined to dialect, written or spoken. The two articles written by Indian technical communicators demonstrate another cultural difference in language use. Indians who speak and write English use the language as a marker of elevated social status, authority and education. The articles by Indian academics are written in ornate, academic Indian English, and contain sentences like this:

While it would be correct to say that traditionally, Indian verbal and nonverbal rhetorical patterns can be classified as indirect, it must also be considered that Indian modern culture has assimilated (and continues to assimilate) influences from Western media, business models and communication styles.

Even for an academic audience, this is the sort of hi-falutin' stuff which would have any proponent of Plain English reaching for their pen and invoking

Orwell for all they were worth. The author of that sentence goes on to suggest that in the interests of globalisation, the rhetorical practices of Indian technical communicators should be

---

**Acknowledging that there is no One True English can only serve to make us more careful, more responsive to local usage and idiom and more effective communicators.**

---

integrated into global curricula. But even if we approved of this sort of inflated writing, we could never use it in documentation for a general audience, as it is a dialect designed to exclude or impress the majority of the population, rather than to give them information. Plain English, and an arguably Western standard, may not be perfect, but they are the best of our imperfect options for the moment.

While I quibble with some of the ideas on language in this work, I completely endorse the notion that cultural sensitivity is an invaluable tool for modern technical communicators. It is essential to understand how an audience will receive, process and use the documentation we develop, how (or whether) they will offer feedback, and what formats and delivery methods will be most effective and appropriate. Assuming that all audiences are the same is a strategy that may have been passable last century, but as the planet gets ever smaller, and the need for content in an expanding information Age gets ever larger, the skills which help us to evaluate the different needs of our audience will become more necessary. Although some of the purported solutions in this work are better in theory than in practice, it deals with an important question, and gives a good justification for expanding our cultural framework as professional communicators.

**Vian Lawson**

## Tips and tricks

### ■ Adding reusable text snippets to an Outlook message

Do you often find yourself retyping the same sentence or paragraph in emails and wished there was a snippet library: a place to store frequently used text for easy re-use? Here is an effective, if somewhat inelegant, solution.

#### Create a snippet

1. In Microsoft Outlook, create a new email message.
2. As the subject of the message, type a name for the snippet of text you want to re-use (say, **Telephone**). This is just to help you identify one snippet from another.
3. In the body of the message, type the snippet (for example, **Feel free to call us on our freecall number (1800 601 116) if you need further information.**).
4. Save and close the message. By default, the message is saved in your **Drafts** folder, but you can move it to any folder you please.

#### Insert a snippet

1. In a message you are writing, place the cursor where you want the snippet to go.
2. Select **Insert > Item [2003]**, **Insert > Attach Item [2007]** or **Insert > Outlook Item [2010]**.
3. Select the folder where the snippet is stored (step 1 in figure 1).
4. Select the name of the snippet (step 2 in figure 1).
5. Select **Text only** (step 3 in figure 1).
6. Click **OK**.
7. The text snippet and subject line is added to your message. Delete the unwanted subject line.

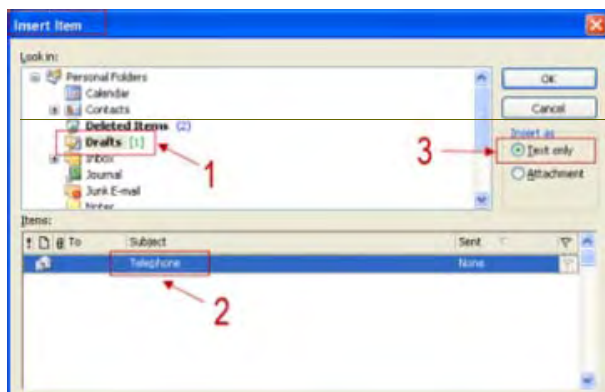


Figure 1: Inserting a text snippet in an Outlook email

Note that if you only need one text snippet, you can leave the subject line blank. You then won't have to delete it each time the snippet is inserted.

### ■ Restoring the correct colour to a bullet in PowerPoint

If you change the colour of text that immediately follows a bullet in a PowerPoint slide, the colour of the bullet also changes. This may not be what you want. To restore the original colour:

1. With your cursor in the bulleted paragraph, select **Format > Bullets and Numbering**.
2. If necessary, select your preferred bullet style.
3. From the **Color** drop-down menu, select your preferred bullet colour.
4. Click **OK**.

### ■ Finding the cursor on a FrameMaker page

If you have wandered elsewhere in a FrameMaker document and want to quickly return to the page where your cursor is:

1. Either:
  - press **CTRL + G** OR
  - choose **View > Go to Page** or
  - click in the page number area of the status bar

The **Go To Page** window appears:

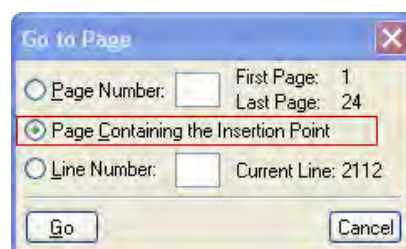


Figure 2: Navigating a FrameMaker document

2. Select **Page Containing the Insertion Point** and click **Go**.

### ■ Switching between body and footnotes (and vice versa)

#### Microsoft Word

**To text:** You can quickly jump back from a footnote to the corresponding place in the body of the text by double-clicking the footnote number.

**To footnote:** Double-clicking the footnote marker in the body of the text will move the cursor to the start of the corresponding footnote.

#### Adobe FrameMaker

**To text:** With your cursor anywhere in the footnote, select **Special > Footnote**.

**To footnote:** Place your cursor directly beside the footnote marker (or highlight it) and select **Special > Footnote**.



# Miscellany

## The cost of poor writing

Benjamin Franklin, one of the so-called *Founding Fathers of the United States*, is credited with the well-worn adage that *time is money*:

“Remember, that *time is money*. He that can earn ten shillings a day by his labor, and goes abroad, or sits idle, one half of that day, though he spends but six pence during his diversion or idleness, ought not to reckon that the only expense; he has really spent, or rather thrown away, five shillings besides.”<sup>1</sup>

So what money does society throw away in tolerating poor writing?

Most workers—whatever the shade of their collar—spend some part of their working day engaged in reading: reports, emails, operating instructions, requests for tender, policies and so on. For some it might be little more than 10 minutes a day; for others it might be five or six hours. Suppose that, on average, 15 minutes of reading time is spent disentangling the intended meaning from poorly expressed language (including the time taken to send emails or make phone calls seeking clarification) and reading words that are unnecessary. Since those 15 minutes are on top of the time the reader would have taken to read everything *had everything been immediately obvious*, we can put a figure on the opportunity cost of poor writing: 15 minutes per day or 1.25 hours per week. That is unproductive time, time that could have put to more profitable uses. In countries where the typical working week is 40 hours, it equates to approximately 3% of paid time. Not much, perhaps—until you see the figure converted to currency and applied nationwide.

Take Australia as an example. As at May 2010, there were approximately 11.5 million employees of which 63.3% (or 7.2 million) were in full-time employment. The average weekly earnings of full-time employees was \$1313 (that is, \$68,000 per annum).<sup>2</sup> Three per cent of \$68,000 is \$2400. Thus the opportunity cost of poor writing in Australia—the money that could be put to uses other than teasing out the meaning of impenetrable texts and wading through verbosity—is at least \$14.75 billion per year: \$14,750,000,000. And that is not counting part-time employees.

Based on a comparison of populations alone, the opportunity cost of poor writing in the United Kingdom is in the order of £25 billion, in Canada in the order of CA\$23 billion, and in the USA in the

order of US\$210 billion. That is, in total, close to US\$300 billion every year in lost productivity. Surely a billion or two a year in remedial English classes wouldn't go astray.

## STC documentation competition

The Australia chapter of the Society for Technical Communication is once again holding a documentation competition. The competition is an ideal way for technical writers to get feedback on their work from an expert judging panel of technical communication professionals.

The competition covers three categories:

- technical publications
- technical art
- online communication

Four levels of achievement are recognised:

- *Best of Show* (which is awarded to one entry in each competition category)
- *Distinguished*
- *Excellence*
- *Merit*

Winners of the *Distinguished* and *Best of Show* awards qualify to enter the STC international competition.

Entries close on 31 October 2011 (and judging takes place throughout November).

For more information and entry forms, contact the competition coordinator at [competitions@stc-aus.org.au](mailto:competitions@stc-aus.org.au).

## Getting familiar with the new Word order

Microsoft Word guru, thought-provoking blogger and indefatigable cyber-correspondent Christine Kent has recently published two new books to help those navigating the choppy waters between earlier versions of Microsoft Word and the current version (Word 2010). The books are entitled:

- *Microsoft Word 2010 Upgrade from 2003: A New Way of Working*
- *Microsoft Word 2010 Upgrade: Building on Word 2007*

You can preview and purchase the books—and see Christine's other books on Microsoft Word and Excel—at <http://www.lulu.com/christinekent>.

## MS Word finally passes the numeracy test

Good news. Microsoft Word 2010 knows when a word is word and when it is not. The **Word Count** feature no longer counts bullets and numbers as words, as in earlier versions of Word. Hallelujah.

---

1. From a letter to an unnamed tradesman, dated 1748. Emphasis added.

2. All the stated statistics can be found on the Australian Bureau of Statistics website at <http://www.abs.gov.au>. Viewed 13 March 2011.



# Mindstretchers

Geoffrey Marnell

The puzzle in the last issue of *Words* posed the following:

Using a common beam balance and any number of identical steel balls, how is it possible for there to be more balls on one side of the balance and yet the balance to be tilted to the other side? Try to think of three distinct answers.

## Solutions

1. Suppose that I place 3 balls on the left-hand side of the balance and 3 identical balls on the right-hand side of the balance. The beam of the balance should, then, be perfectly horizontal. Suppose that the balls can roll off the end of the beam but cannot roll towards the fulcrum. If I now place a fourth ball on the right-hand side of the balance, the balance tilts towards the right and the balls start to roll off. When one ball has rolled off, there are 3 balls on each side of the balance, but owing to the momentum of the rolling balls the beam is still, at this very point, tilted towards the right.

Before the beam has returned to its original position, another ball will have rolled off the beam and, at this instant, although there are 3 balls on the left and only 2 on the right, the beam is still tilted, though momentarily, to the right.

Indeed, depending on the characteristics of the beam balance and the balls, it can happen that, by the time the balance has returned to the horizontal, only one ball remains on the right-hand side of the balance.

2. The further an object is from the fulcrum of a beam balance, the greater the torque and so the greater the *measured weight*. Consequently, if you placed 2 balls 20 cm along the right-hand side of the beam and 3 balls 10 cm along the left-hand side of the beam, the beam will tilt toward the side with fewer balls.

3. Magicians love canes—especially hollow ones. No doubt you have seen a magician place a cane across the edge of a table so that more of the cane is projecting into the air than lying on the table. Miraculously (or so you are to believe), the cane does not fall to the floor. Then the magician turns the cane around so that the end previously resting on the table is projecting into the air. Again more of the cane is projecting into the air than resting on the table; and again the cane refuses to fall to the floor. How is this done?

The cane is hollow, and inside it there is a lead weight that can slide from one end of the cane to the other. The magician simply ensures that the lead weight is always at the end of the cane resting on the table. When the magician turns the cane around, it is tilted slightly so that the weight slides to the end the magician is about to place on the table.

Relating this back to our puzzle: so long as one arm of the beam balance is heavier than the other—either because it is a little longer or because it conceals weights not carried by the other arm—it is possible to place just 2 balls on the doctored arm and 3 on the other arm and find the balance tilting towards the side with fewer balls.

## The *Words* team

- Artwork: Christine Weaver
- Copy-editor: Claire Mahoney
- Editor: Geoffrey Marnell
- Contact: [words@abelard.com.au](mailto:words@abelard.com.au)

© Individual contributors or Abelard Consulting Pty Ltd, 2011  
[unless otherwise noted]



**MACQUARIE  
DICTIONARY  
ONLINE**

[www.macquariedictionary.com.au](http://www.macquariedictionary.com.au)

Subscribe to the complete Macquarie Dictionary online, updated annually with new words and definitions.

Also available online is the full Macquarie Thesaurus - that perfect word is just a click away.

**Try it out now for FREE!**

Macquarie Online is offering free extended trial access. Simply contact Macquarie Online to set up your 3 months free access. Quote code: 3mfTrialAC

Macquarie Online Support  
phone: 1800 645 349  
email: [support@macquarieonline.com.au](mailto:support@macquarieonline.com.au)



*Australia's national dictionary*