Alex Groce (agroce@gmail.com), Oregon State University

Frederick P. Brooks, Jr.'s *The Mythical Man-Month* is the single most obvious choice for a column reviewing classics in software engineering. Addison-Wesley issued a 20th anniversary edition in 1995, including Brooks' famous essay "No Silver Bullet: Essence and Accidents of Software Engineering." In 2002 John Knight placed *The Mythical Man-Month* first on a list of desert island books related to software engineering. If you accost a random Software Engineering Notes reader on the street (I don't recommend this), and ask them to name a classic, or even favorite, book in software engineering, they will think you are crazy. But they will name Brooks' wonderful book, nine times out of ten.

This makes it a difficult book to review! A book review can be useful when it recommends something that everyone should read, but no one does read. It can also be useful when it says "don't read this." A review that states the obvious -- "read this book, which you've almost certainly already read, and will read again" -- has to go to some lengths to justify its existence. Otherwise, just picking up your handy copy of *The Mythical Man-Month* and reading a couple of random pages would be a better use of your time, reader. What can be said about this book that has not already been said? Possibly very little.

The problem of saying things about an already much-discussed book is not new. *The Mythical Man-Month* may be the "Bible" of software engineering, but it has not endured two-thousand years or more of concentrated examination like the actual Old and New Testaments. Therefore, it should be open to approach by the method Donald Knuth used to discuss the actual Bible, in his offbeat work *3:16 Bible Texts Illuminated*, which combines scripture study and calligraphy. Knuth proposes to get a feeling for a large and long-discussed book by performing a random stratified sample of the text: in his case, examining in detail the 16th verse of the 3rd chapter of each book of the Bible. For a short review of *The Mythical Man-Month* an equivalent approach is easily devised. The 20th anniversary edition has 15 chapters from the original book, plus the 16th chapter reprinting "No Silver Bullet," which has become as well-known as the main text (and some further chapters commenting on these core 16 chapters). This gives 16 "real" chapters, still too many for a short review, so we take every other chapter, a reasonable choice in this case since Brooks' book does not change genre, style, topic, and author, unlike Knuth's targeted text. The third page and eighth sentence of each chapter matches Knuth's choices well, adjusting for length (most pages don't have 16 sentences). Brooks might appreciate the origin of this method, as *The Mythical Man-Month* itself is full of rich Biblical metaphors. One thing a random sample can discover is whether a book really spends its time talking about what we vaguely remember it as being about. What does this random sampling tell us about *The Mythical Man-Month*?

> Chapter 2: "Men and months are interchangeable commodities only when a task can be partitioned among many workers *with no communication among them* (Fig. 2.1)."

Well, this is off to a good start. Perhaps a random sample really can capture essence and accident! The first sample is Brooks' most clear statement of why the "man-month" is a myth in software, and why late projects become later when you "throw people at the problem." Even if readers forget all the other deep wisdom in the book, everyone remembers than you can't divide amount of code to be written by people to get a schedule, because people have to communicate with each other to create software.

> Chapter 4: "It is not enough to learn the elements and rules of combination; one must also learn the idiomatic usage, a whole lore of how the elements are combined in practice."

The second sample comes from a chapter discussing conceptual integrity, and noting that simplicity is not the only criteria for an easy-to-use design -- "straightforwardness" is also required. Brooks uses examples from programming languages, where a language may be simple (with few basic concepts) but not straightforward, in that doing useful things is difficult. This is again a fairly major concern of the book, so sampling is still working well, if not giving deep new insights into the "real" concerns of the book.

> Chapter 6: "Many tools are available for formal definition."

The sixth chapter continues some themes from the last sample, discussing how, once an architect has created a design with conceptual integrity, that can be communicated. Brooks suggests that in the long run, both formal definitions and natural language prose will be necessary for good specifications. Maybe random sampling is *too* good --- all we're learning is that *The Mythical Man-Month* is about exactly what we think it's about, even if we last read it ten years ago.

> Chapter 8: "Of these, Fig. 8.2 is the most detailed and useful."

At first glance, this is a failure for random sampling. In fact, it's from a chapter on estimating effort in software engineering projects, and for this topic Brooks turns more empirical than usual, an early bellwether for the modern emergence of empirical software engineering. If you want to predict things, look at real data, closely, he's telling us.

> Chapter 10: "**What: objectives.** This defines the need to be met and the goals, desiderata, constraints, and priorities."

Brooks is just looking at the most important pieces of documentation for a project in this chapter, and unsurprisingly a basic statement of project goals is one of the key points.

> Chapter 12: "An auxiliary 1401 with terminals was used to schedule runs, to keep track of the thousands of jobs, and to monitor turnaround time."

Here we run into one of the few places where a book that is, in theory, largely about the development of an operating system announced in 1964 actually might appear dated. The chapter as a whole is about tools, and written before interactive programming became the normal practice, so the discussion of good tools for scheduling project hardware may seem an archaic concern. Those readers who have ever worked on embedded projects may know that in fact this problem, too, is still around and still difficult.

> Chapter 14: "It also shows how much an activity can slip before it moves into the critical path."

Unfortunately, the problem of late software deliveries remains a constant of software engineering. Little about Brooks' discussion of "disastrous schedule slippage" will sound outdated. Chapter 14 encourages charting and planning for potential schedule slips, a precursor to the risk management and other methods since developed as defenses against "losing a year, one day at a time."

> Chapter 16: "The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions."

"Chapter 16" is the essay "No Silver Bullet -- Essence and Accident in Software Engineering," written 10 years after the first 15 chapters, which, unlike *The Mythical Man-Month*, provoked not only admiration but vociferous argument. This sentence explains what Brooks thinks is the essence of software, which directly leads to his argument that there will be no silver bullet, because the essence of software is conceptual, and working with complex concepts is inherently hard.

Stratified random sampling of *The Mythical Man-Month* shows that this book is so well-organized and thematically unified that picking arbitrary sentences may miss any particular favorite topic of a reader (I was sad that the extreme difficulty of testing and debugging completely disappears), but will produce a remarkable number of sentences that elegantly capture the topics that have dominated software engineering for the last 40 years. Almost half of the samples don't even need much elaboration of their context, which is a better ratio than Knuth obtained with the Bible. The one indisputable classic of software engineering is itself a symbol of the conceptual integrity it advocates. In a deeper sense, however, random sampling fails, because it cannot convey the conversational charm and metaphorical richness that most reward the reader, and bring us back again and again.