Automated Software Engineering, 10, 399–402, 2003

Desert Island Column

MARTIN GLINZ glinz@ifi.unizh.ch
University of Zurich, Switzerland

If I were marooned on a desert island and could wish for some reading, I would, frankly, not choose any software engineering book. But, for the sake of this column, let's assume a constraint stating that only software engineering literature is at hand. So what would I choose? The task proved to be more difficult than I had thought. After a rather long period of reflection (and a friendly reminder from the ASE editor that my ship to desert island was about to depart) I decided to solve the problem by making it harder. So I changed the given requirement "may be seminal, thought-provoking or simply a pleasure to read" from a disjunction to a conjunction: "shall be seminal *and* thought-provoking *and* a pleasure to read".

This turned out to be a very effective filter. None of the textbooks on software engineering that I know passes it (if you, dear reader, know one which does, please let me know). Just a few textbook chapters come to my mind that pass my filter, for example the *Introduction to Software Engineering* (Chapter 1) from Richard Fairley's *Software Engineering Concepts* (1985) and also his Section 5.1 on *Fundamental Design Concepts* or the beautiful Chapter 3 on *Software Engineering Principles* in *Fundamentals of Software Engineering* by Carlo Ghezzi, Mehdi Jazayeri and Dino Mandrioli (1991).

When filtering all the other books on software engineering that I know, again nearly nothing passed. Even very famous books were filtered out—for example, Barry Boehm's Software Engineering Economics (1981) which is seminal, but no pleasure to read.

A small list of books and articles survived my filter. As desert island reading is a very personal issue, I finally added another constraint: "shall have impacted my way of thinking, researching or teaching".

The book with the highest scores in all four criteria is Fred Brook's Anniversary Edition of *The Mythical Man Month* (1995). Next comes William Kent's *Data and Reality* (1978). In third place I have the *February 1986 issue of IEEE Transactions on Software Engineering* (1986). Finally, I have two books on software requirements on my list (the reader should be warned here that requirements engineering is my field of research, so I am probably biased). These are *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudice* by Michael Jackson (1995) and *Exploring Requirements: Quality before Design* by Don Gause and Gerry Weinberg (1989). I will portray the winner in depth and then discuss the others briefly.

The Mythical Man Months is seminal: how many other books on software engineering still sell after 27 years and have seen only one revision during this period?

It is thought-provoking: the original 15 essays are crammed with experience and insight into how and why things go right or wrong in software engineering. Among the four essays added to the anniversary edition is the famous 'No Silver Bullet' essay (originally an invited

paper presented at the IFIP World Computing Conference in 1986 and then reprinted in IEEE Computer where it stirred a hot debate in the software engineering community).

And finally, this book is definitely a pleasure to read: the style is lucid and elegant, the layout pleases the reader's eyes, and the illustrations really do illustrate the points that the essays make.

Let me mention some remarkable points, just in the order that they come to my mind.

- Many computer books written in the 1970ties read today like tales of bygone days, whereas most of Brooks' essays sound as if they had been written just recently. Frequently, only accidental markers—for example, speaking of 'men' where one would use 'persons' today or mentioning all the product names from the 1960ties—remind the reader that the original 15 essays are 28 years old. Of course, many of the concrete examples are obsolete, for example PL/1 being "the only reasonable candidate" for a high-level language. However, the general arguments for using high-level languages are still valid.
- The insight into the joys and the woes of software development (in the first essay *The Tar Pit*) are about human nature; they probably have no expiration date at all. The essays on software project management are also about the essentials—problems that you might find characterized in the Dilbert comic of this week. There are just a few things that the advances of technology have rendered obsolete, for example the problem of turnaround time in batch-mode software development or some paragraphs on debugging (in *The Whole and the Parts*).
- The plea for software architects and architectural integrity of systems (in *Aristocracy, Democracy and System Design*) sounds stunningly modern—it could be taken from a recent textbook on software architecture.
- Metaphors are a powerful means for conveying ideas. In recent years, the Extreme Programming movement has brought this old fact to the attention of the software engineering community again. Nearly every chapter title and their accompanying illustrations in Brooks' book are metaphors, starting with the beautifully illustrated tar pit metaphor.
- When following today's disputes about process-oriented vs. agile software development, you find some of the essential issues behind this discussion in the essays *The Surgical Team* and *Aristocracy, Democracy and System Design*.
- The essay *Plan to Throw One Away* highlights the role and the importance of prototyping and pilot systems—written long before others recognized it and still up-to-date.

The anniversary edition comes with four new chapters (including the 'No Silver Bullet' essay), so I prefer this edition to the original one. In his final essay, Brooks reviews the original 15 essays after 20 years. He discusses where he was right and where he was wrong, what has become obsolete and what is still an issue. This essay concludes with a thought which in my view is at the heart of the software engineering problem: "One can expect the human race to continue attempting systems just within or just beyond our reach". In other words: in software engineering, we are continuously eating up the progress we make by forever building systems that are larger and more complex than ever before.

Now, let's turn to numbers two to five on my desert island reading list.

When I was a young researcher, I came across an IBM Technical Report from 1976 that definitely changed my way of thinking (Kent, 1976). It began with the unforgettable

words "A message to mapmakers: highways are not painted red, rivers don't have county lines running down the middle, and you can't see contour lines on a mountain." Two years later, this report became a book: *Data and Reality* by William Kent (1978). This book is another item from the small list of computer science books that are more than 20 years old and are still in print. It is a masterpiece about creating models of the real world and about interpreting these models properly. Kent points out that modeling is not just about mapping reality to software: it is also about creating reality and shaping our perception of reality. All those poor guys who believe that the world is object-oriented (and that its objects are just sitting there, waiting to be discovered) should be prescribed reading this book as a therapy.

The February 1986 issue of IEEE TSE (1986) is one of the best issues that was ever published in TSE—if not *the* best. Let me just mention a few articles. My favorite is David Parnas' and Paul Clements' seminal paper on how and why to fake a rational design process (1986). They convincingly argue that it is normally impossible to develop software according to a rational process (where programs are derived systematically from a set of requirements), but that it is nevertheless worthwhile writing the documentation such that it looks as if the software had been produced according to such a rational process.

This issue of TSE also contains one of the first papers on object-oriented design, written by Grady Booch (1986). Furthermore, there is a wonderful overview of JSD by John Cameron (1986). Finally, let me mention Harlan Mills' and Richard Linger's thought-provoking paper about programming without arrays and pointers (1986). And there is more to discover in this issue.

Michael Jackson's book on *Software Requirements and Specifications* (1995) was excellently portrayed by Colin Potts in his desert island column of October 1997 (1997), so I need not add anything but recommending you to re-read Colin's column.

Finally, in *Exploring Requirements: Quality before Design*, Don Gause and Gerry Weinberg (1989) take a fresh and thought-provoking look at requirements engineering, which is different from nearly all the books on requirements engineering that have been written before or after. Instead of focusing on process, methods, tools, management, and so on, they take a human-centric standpoint and demonstrate how to work with humans in requirements engineering, how to elicit their requirements and how to validate them.

So here we are, ready to go to desert island. However, don't tell my university where I am going. If they knew, they would immediately start convincing me that a desert island is not a place for lazy reading but a wonderful retreat for writing annual reports, research proposals, examination regulations, academic program guidelines, and many more of these genuine and ever-lasting contributions to the advancement of our discipline.

References

Boehm, B. 1981. *Software Engineering Economics*. Prentice Hall.
Booch, G. 1986. Object-oriented development. *IEEE Transactions on Software Engineering*, SE-12, pp. 211–221.
Brooks, F.P. 1995. *The Mythical Man Month*. Anniversary edition. Addison-Wesley.
Cameron, J.R. 1986. An overview of JSD. *IEEE Transactions on Software Engineering*, SE-12, pp. 222–240.
Fairley, R. 1985. *Software Engineering Concepts*. McGraw-Hill.
Gause, D.C. and Weinberg, G.M. 1989. *Exploring Requirements: Quality before Design*. Dorset House.
Ghezzi, C., Jazayeri, M., and Mandrioli, D. 1991. *Fundamentals of Software Engineering*. Prentice Hall.

IEEE TSE, 1986. Special issue on software design methods. *IEEE Transactions on Software Engineering*, SE-12, p. 2.

Jackson, M. 1995. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addison-Wesley (ACM Press Books).

Kent, W. 1976. *Describing Information (Not Data, Reality?)*. IBM General Products Division, Palo Alto, Technical Report TR03.012.

Kent, W. 1978. *Data and Reality*. North-Holland.

Mills, H.D. and Linger, R.C. 1986. Data structured programming: Program design without arrays and pointers. *IEEE Transactions on Software Engineering*, SE-12, pp. 192–197.

Parnas, D.L. and Clements, P.C. 1986. A rational design process: How and why to fake it. In *IEEE Transactions on Software Engineering*, SE-12, pp. 251–257.

Potts, C. 1997. Desert island column. *Automated Software Engineering*, 4(4):463–466.