There Is Still No Silver Bullet

David Alan Grier, Djaghe, LLC

Throughout its existence, Computer *has been at the center of discussion over how software should be engineered and how it should be developed. No article has been so central to the discussion as "No Silver Bullet" by Frederick P. Brooks.*

Few articles have so thoroughly entered the consciousness of our profession as much as "No Silver Bullet: Essence and Accidents of Software Engineering" by Frederick P. Brooks. Though the article is ranked 15th on the list of the most influential *Computer* articles, it has the rare distinction of being referenced in the title of 152 articles in the IEEE library. By choosing a clever title, one that referenced both the legend of the werewolf and the popular story of the Lone Ranger, Brooks was able to engage the imagination of a large audience. However, his message is fundamental and touches fundamental issues of our field.

In this article, Brooks argued that software development was a complex, difficult, multidimensional task. Because of that complexity, there was no simple approach that would make software development easy or inexpensive. "There is no single development, in either technology or in management technique," he wrote, "that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity."[2]

When he wrote "No Silver Bullet" in 1986, Brooks had established himself as one of the pioneers of computer

ARTICLE FACTS

- Article: "No Silver Bullet: Essence and Accidents of Software Engineering"
- Author: Frederick P. Brooks
- Citation: *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987
- *Computer* influence rank: #15 with 15,686 downloads and 1,463 citations

Digital Object Identifier 10.1109/MC.2020.3042682
Date of current version: 11 February 2021

0018-9162/21©2021IEEE

EDITOR **DAVID ALAN GRIER**
Djaghe, LLC; grier@gwu.edu

science. He studied for his doctorate at Harvard under Howard Aiken, the designer of the Mark I mechanical computer. After graduating, Brooks joined IBM, where he first worked on the IBM Stretch and then led the development of the OS/360 operating system.

The OS/360 project was one of the watershed activities in computer science. Though the product eventually became quite successful, it had a long and difficult birth. "The magnitude of the task of developing the proposed operating system was grossly underestimated," wrote Emerson Pugh, the IBM historian.[4] It was "a very educational experience, albeit a very frustrating one," Brooks would later admit. "The product was late, it took more memory than planned, the costs were several times the estimate, and it did not perform very well until several releases after the first."[3]

After completing his work on OS/360, Brooks wrote *The Mythical Man-Month*,[3] a book that became an important early contribution to the software engineering literature. The basic argument of the book is that an intellectual task involves a substantial amount of education and learning. It is foolhardy to treat it as aggregated labor that can be easily measured in man-months and divided among a group of undifferentiated workers. Every time you add a new worker to a software project, you face new problems of communication among the staff, familiarization with the details of the project, and testing and coordination of results.

In his book, Brooks articulated his most famous observation about software engineering: "Adding manpower to a late software project makes it later." When a schedule "slippage is recognized," he wrote, the natural "response is to add manpower. Like dousing a fire with gasoline, this makes matters worse, much worse."[3]

Over the history of our field, hardware designers have been much more successful than software engineers in exploiting economies of scale. They've been able to turn hardware development into a mass production process that has a fixed development cost and becomes profitable by delivering large numbers of identical shipped items or circuits.

In theory, software engineers should be able to achieve the same kind of result. Software, after all, has a very low reproduction cost. One can produce copies of an original master very inexpensively and easily distribute them to customers. This was the goal of the software industry as it emerged in the early 1970s. The pioneers of that field felt that it should be straightforward to produce a product that could find a large market and hence allow companies to amortize their development over a large base of customers.

Hardware designers have been much more successful than software engineers in exploiting economies of scale.

As Brooks discussed in his "No Silver Bullet" article, that goal for the software industry has proven elusive in many cases. "We hear desperate cries for a silver bullet, something to make software costs drop as rapidly as computer hardware costs do," he wrote. The reasons for this situation are many. It is expensive to get customers to define their requirements. They often find it difficult to articulate those requirements accurately, and many times, they change their minds as the software progresses. For their part, developers routinely underestimate the effort required to simply design and code software. They often realize the difficulty of a problem only as they are working on the software, and they encounter issues that they could not have foreseen when they began.

Once the software is operational, the costs of product development do not drop to zero. The software needs to be installed and configured. Customers need to be trained. Errors need to be corrected, and new features must be added. It takes a very skilled organization to develop successful software and earn a profit in the process. "Unless a software project has clear definitions of its key milestones and realistic estimates of the time and money it will take to achieve them," wrote software engineer Barry Boehm, "there is no way that a project manager can tell whether his project is under control or not."[1]

So was Brooks telling us that the field of software engineering is a discipline without hope? Of course not. But as the title of his essay so clearly communicated, it is a multifaceted field in which victories are often small and come at the cost of substantial effort. Certainly, in the years since Brooks made his observations, software engineering has seen some considerable accomplishments. The family of agile methods has brought developers closer to their clients. Formal methods have improved the process of requirement engineering. A host of tools, libraries, application stacks, and fast-prototyping environments has improved the process of writing code. We have automated processes for debugging

and new methods of configuration management. Financial and decision models have reduced uncertainty and limited costs.

Yet, almost 35 years after he wrote this contribution to knowledge, Brooks's observation remains true. Software engineering is a complex and demanding field that poses a host of problems to the practitioner, and there is no single solution, that is, no silver bullet, that will provide a simple way to reduce the work required to create a software product.

ACKNOWLEDGMENT

For these 2021 columns, "Body of Knowledge" takes its information from a report prepared by the IEEE Publications office on 15 November 2019, though it updates the statistical information to reflect the downloads as of the date the essay was written. Other citation services can and do give different numbers.

REFERENCES

1. B. Boehm, "Software engineering economics," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 1, pp. 4–21, Jan. 1984. doi: 10.1109/TSE.1984.5010193.
2. F. P. Brooks, "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, 1987. doi: 10.1109/MC.1987.1663532.
3. F. P. Brooks, *The Mythical Man-Month*. Reading, MA: Addison Wesley, 1975.
4. E. Pugh, *Building IBM*. Cambridge, MA: MIT Press, 1995, p. 294.

DAVID ALAN GRIER is a principal with Djaghe, LLC, Washington, D.C., USA. He is a Fellow of IEEE. Contact him at grier@gwu.edu.

Digital Object Identifier 10.1109/MC.2021.3051514