Editor: **Cesare Pautasso**
University of Lugano
c.pautasso@ieee.org

Editor: **Olaf Zimmermann**
University of Applied Sciences
of Eastern Switzerland, Rapperswil
ozimmerm@hsr.ch

Insights from the Past

The *IEEE Software* History Experiment

Željko Obrenović

With now 200 issues, *IEEE Software* has been making software engineering history since 1984. Advisory board member Željko Obrenović discovered this anniversary during his archeological expedition through the magazine archives. Starting from the surface (by collecting the magazine's covers), he then went deeper to gather insightful quotes, bibliometrics, and topic trends. So, we invited him to share his reflections on the past so that it might not be reinvented but become a source of inspiration for the present and future. —*Cesare Pautasso and Olaf Zimmermann*

THE *IEEE SOFTWARE* history website (obren.info/ieeesw) is a curated site complementing the official *IEEE Software* website (www.computer.org/software). It offers a look at *IEEE Software*'s history at a glance. Here, I discuss its genesis, how it illustrates the practical value of historical data, and how it offers a glimpse into the magazine's future.

The Website's Genesis

The website has developed organically. There never was an official project to develop an elaborate overview of the magazine's history. Rather, the website evolved somewhat out of curiosity and as a volunteer initiative, partly in reaction to positive feedback about its developing content.

The idea for the website arose during the 2016 *IEEE Software* editorial-board meeting at the Software Improvement Group (SIG) in Amsterdam. As an organizer of the meeting, I was looking for ways to create an *IEEE Software* atmosphere. I decided to print the magazine covers and put them up as wallpaper (see Figure 1).

The board members, who were from both academia and industry, liked such an overview of topics and trends that were once considered important. For many, it brought back memories or created awareness of missed topics. In addition, the covers are attractive. SIG kept them on the wall for several months after the meeting.

After receiving requests to share digital versions of the covers, I created a simple website to display them. Thus, the original idea of the history website was only to create that display.

While collecting the covers, I discovered that the July/August 2017 issue would

0740-7459/17/$33.00 © 2017 IEEE

FIGURE 1. *IEEE Software* front covers displayed at the 2016 editorial-board meeting.

OTHER HISTORIES OF SOFTWARE ENGINEERING

The *IEEE Software* history website (obren.info/ieeesw) complements other resources describing software engineering history, such as these (links to which are also on the website):

- "History of Software Engineering"; en.wikipedia.org/wiki/History_of_software_engineering.
- N. Wirth, "A Brief History of Software Engineering," *IEEE Annals of the History of Computing*, vol. 30, no. 3, 2008, pp. 32–39.
- "A Brief History of Software Engineering," Viking Code School; www.vikingcodeschool.com/software-engineering-basics/a-brief-history-of-software-engineering.
- A. Brennecke and R. Keil-Slawik, eds., *Position Papers for Dagstuhl Seminar 9635 on History of Software Engineering*, 1996; www.dagstuhl.de/Reports/96/9635.pdf.

be the 200th issue. So, I decided to use the history website to celebrate this anniversary. I extended the website with several types of content, including these:

- *More than 1,000 quotes*. This was the most rewarding part of creating the site. These curated quotes make the website much more than a simple metadata index. The quotes have also been important in creating interesting content to promote the history of *IEEE Software* on social media such as Twitter because they're short but informative.
- *Indexes of all 3,000+ articles and 4,000+ authors*. These indexes enable quick exploration of articles and authors in a historical context. I also added a historical timeline for search results.
- *A citation index* (based on Google Scholar searches) correlated with the publication year. This helps show *IEEE Software* articles' broader impact. It also aids identifying the most cited articles, authors, or themes.

For more on the website's content, see the sidebars.

Historical Data's Practical Value

Although the history website is just six months old, I've gathered enough experience to reflect on its value. I classify these lessons learned into the following categories.

Seeing Trends

Historical data enables us to see trends in software engineering research and practice. In many aspects, this is the history website's main value, compared to digital libraries such as the IEEE Computer Society Digital Library (CSDL; www.computer.org/csdl) and IEEE Xplore (ieeexplore.ieee.org).

To illustrate the possibilities of seeing trends, Figure 2 shows word clouds created from terms in *IEEE Software* article titles, for four decades. Although *IEEE Software*

FIGURE 2. Word clouds with terms in the titles of *IEEE Software* articles. (a) 1980s. (b) 1990s. (c) 2000s. (d) 2010s.

covers diverse topics, each decade has had a few topics that were more popular.

In the 1980s, topics related to different programming paradigms were popular. For instance, parallel and distributed programming were important topics. Six theme issues and cover articles discussed programming:

- 1984, no. 2. Programming: Sorcery or Science?
- 1986, no. 4. Firmware Engineering: The Interaction of Microprogramming and Software Technology.
- 1988, no. 1. Parallel Programming: Issues and Questions.
- 1988, no. 3. What Is Object-Oriented Programming?
- 1989, no. 4. Parallel Programming: Harnessing the Hardware.
- 1989, no. 5. A Compositional Approach to Multiparadigm Programming.

In the 1990s, the focus shifted toward process-related topics. Measurements, metrics, and quality assurance also received significant attention. Ten theme issues covered process management and metrics (and their combination):

- 1990, no. 2. Using Metrics to Quantify Development.
- 1991, no. 4. Process Assessment.
- 1992, no. 4. Reliability Measurement.
- 1993, no. 4. The Move to Mature Process.
- 1994, no. 4. Measurement-Based Process Improvement.
- 1996, no. 4. Managing Large Software Projects.
- 1997, no. 2. Assessing Measurement.
- 1997, no. 3. Managing Risk.
- 1998, no. 4. Menace or Masterpiece? Managing Legacy Systems.
- 1999, no. 2. Metrics for Small Projects.

The 2000s were clearly the age of requirements engineering. Overall, *IEEE Software* has published 163 articles with "Requirements" in the title. Of those articles, 91 (56 percent) were published in the 2000s. Seven theme issues covered requirements engineering:

IEEE SOFTWARE BIBLIOMETRIC DATA

The *IEEE Software* history website (obren.info/ieeesw) combines data from the IEEE Computer Society Digital Library (CSDL), IEEE Xplore, and Google Scholar. The CSDL and Xplore provide useful data about the number of articles and authors.

Approximately 4,500 *IEEE Software* articles are indexed in Xplore. However, this number includes front and back covers, tables of contents, and ads. I built a script that extracts only articles with authors. This leaves around 3,250 "proper" articles. Approximately half of those articles are peer reviewed; the other half includes columns and invited content.

In total, more than 4,200 authors have contributed to *IEEE Software*. Of those authors, 819 have contributed multiple times—for example, Diomidis Spinellis (75 articles), Grady Booch (68), Robert Glass (57), Christof Ebert (46), and Forrest Shull (43). These 819 authors authored or coauthored approximately two-thirds of the articles. Fifty-one percent of the articles (mostly department articles) have one author; 49 percent have multiple authors.

Figure A shows the number of authors and articles per year.

The history website also contains citation data extracted from Google Scholar in February 2017:

- The cumulative *IEEE Software* citation count is 161,042 (the sum of all "cited by" fields).
- The magazine's *h*-index is 181; approximately one-half of the citations are from these top 181 articles.
- The most cited year is 1990, followed closely by 2003 and 1994.
- The most cited articles are "The 4+1 View Model of Architecture" (2,786 citations), "Reverse Engineering and Design Recovery: A Taxonomy" (2,594 citations), and "Software Risk Management: Principles and Practices" (1,925 citations).

Figure B shows the number of IEEE citations per year of publication.

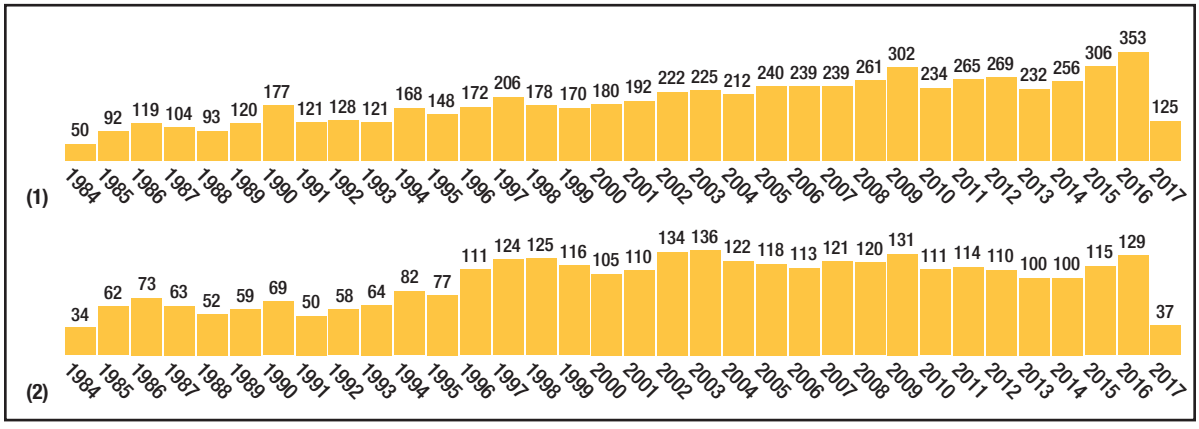

FIGURE A. The number of (1) authors and (2) articles per year in *IEEE Software*.

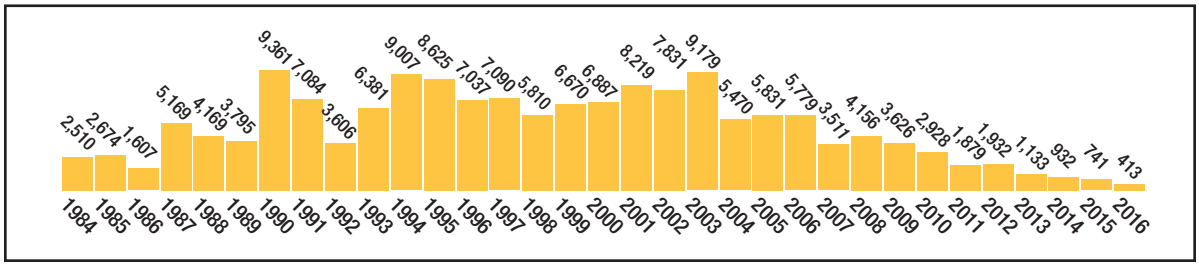

FIGURE B. The number of IEEE citations per year of publication for *IEEE Software*.

HISTORY TWEETS—INJECTING THE PAST INTO SOCIAL MEDIA

The *IEEE Software* history website (obren.info/ieeesw) also promotes *IEEE Software* on social media. From October 2016 to June 2017, to draw attention to the magazine's 200th issue (July/Aug. 2017), I've been daily tweeting *IEEE Software* covers and quotes and interesting historical findings. In this way, each *IEEE Software* issue has been mentioned at least once before the publication of our 200th issue.

The tweets have provided an interesting way to engage with a broader, younger audience. Many of the old articles from the 1980s have received significant attention. Social-media interaction has also enabled us to reconnect with some of the early authors.

Also, on Twitter under #SE_history (twitter.com/hashtag/se_history) are more than 500 tweets about *IEEE Software* history. We plan to tweet there again as new issues are added. Figure C shows a few interesting tweets.

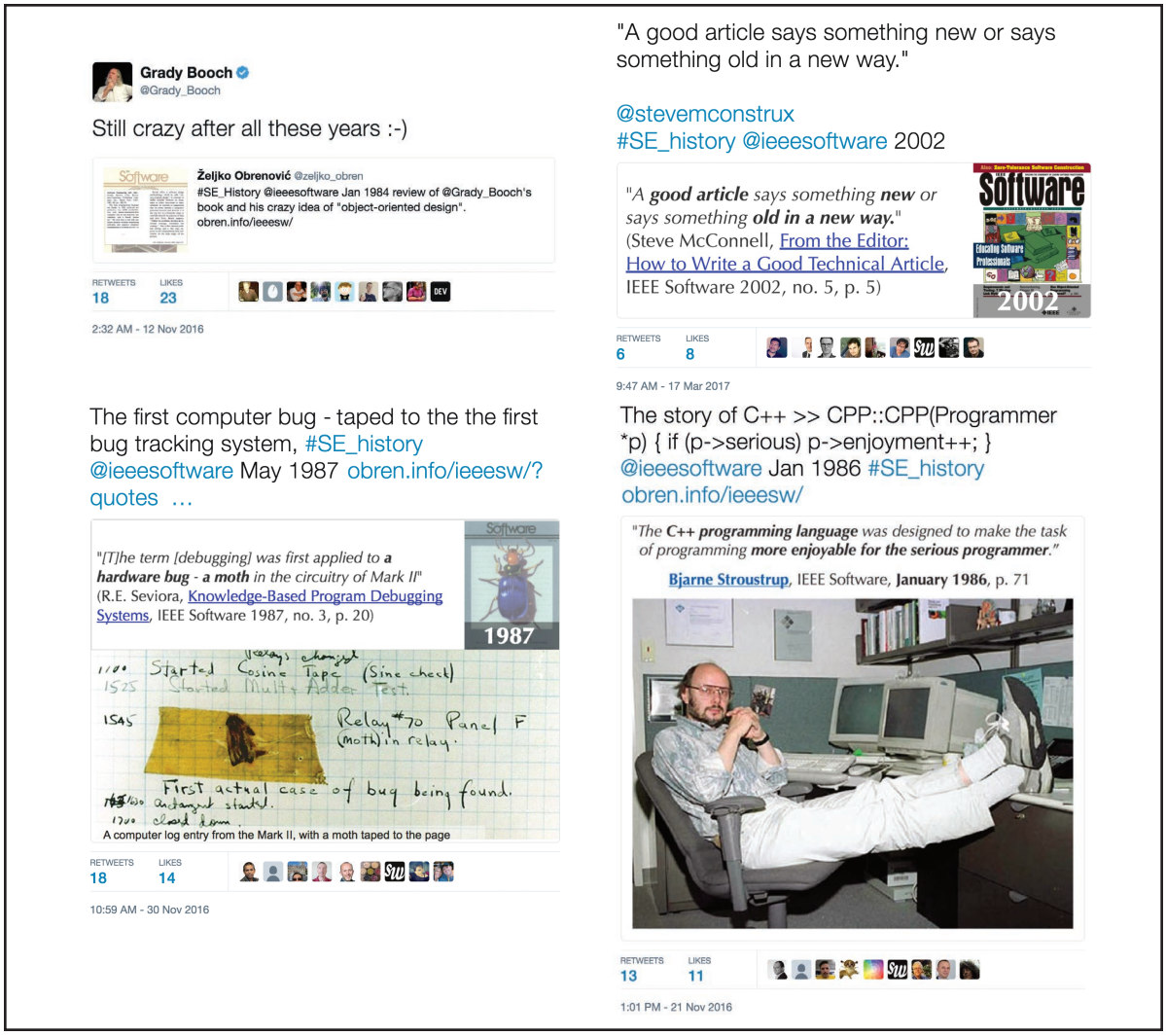

FIGURE C. Some interesting tweets from the *IEEE Software* history website.

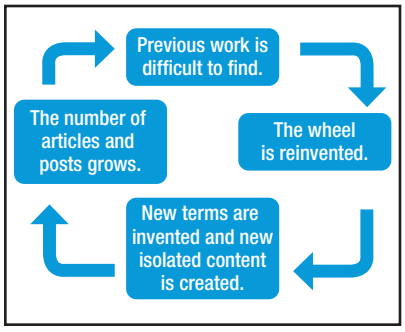

FIGURE 3. The vicious cycle of inflation of software engineering terms and knowledge. New cycles of such reinventions are inevitable.

- 2000, no. 3. Requirements Engineering: Getting the Details Right.
- 2003, no. 1. RE 02: A Major Step toward a Mature Requirements Engineering Community.
- 2004, no. 2. Practical Requirements Engineering Solutions.
- 2005, no. 1. Innovation in Requirements Engineering.
- 2006, no. 3. RE 05: Engineering Successful Products.
- 2007, no. 2. Stakeholders in Requirements Engineering.
- 2008, no. 2. Quality Requirements.

In the 2010s, the focus shifted toward architecture:

- 2010, no. 2. Agility and Architecture.
- 2013, no. 2. Twin Peaks of Requirements and Architecture.
- 2013, no. 6. Architecture Sustainability.
- 2015, no. 5. Software Architecture.
- 2016, no. 6. The Role of the Software Architect.

Overall, *IEEE Software* has published 167 articles with "Architecture" or "Architect" in the title. Of those articles, 97 (58 percent) have been published since 2010.

Preventing Knowledge Inflation

> *There's a lot of forgetting, and a lot of "never knew that" in our field today.*[1] *—Robert Glass*

> *Those who cannot remember the past are condemned to repeat it. —George Santayana*

Knowing history can help us avoid repeating errors and building on each other's work. As Robert Glass noted, we keep forgetting early contributions and often reinvent the wheel. In my experience, much current software engineering work, especially practitioner's books and posts, aren't well connected to previous work. Similarly, Martin Fowler talked about *semantic diffusion*, which occurs when a definition gets spread through the wider community in a way that weakens it.[2] Often this weakening is a consequence of lack of awareness of the original work related to the definition.

I call this problem *the inflation of software engineering terms and knowledge* (see Figure 3). The difficulty of finding previous work often leads to reinvention of concepts and solutions. The reinvented solutions get documented and published, leading to the invention of new terms and creation of isolated content (unconnected to previous work). These new terms and content increase the already significant number of articles and posts, which makes finding previous work even more difficult. New cycles of such reinventions are inevitable.

Unfortunately, *IEEE Software* is partly to blame for this cycle. Just by looking at the covers, you can see that many themes repeat. For instance, there have been four "business of software" issues:

- 2002, no. 6. The Business of Software Engineering.
- 2004, no. 5. The Business of Software Engineering.
- 2011, no. 4. Software as a Business.
- 2016, no. 5. The Business of Software.

Looking at the introductions of the later theme issues, you can see that none of them connects to any of the previous ones. Nor do they relate to the brilliant but largely forgotten 1984 issue (no. 3) on Capital-Intensive Software Technology.

IEEE Software provides content that can help root new contributions in previous solid peer-reviewed research and practices. Many *IEEE Software* authors have been the originators of nowadays mainstream concepts and ideas. For examples, 10 of the 17 authors of the "Manifesto for Agile Software Development"[3] have written for *IEEE Software*: Kent Beck, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Andrew Hunt, Ron Jeffries, Robert Martin, Steve Mellor, and Dave Thomas.

Detailed, easily accessible historical data can help slow down the inflation of knowledge by making it easier to find and connect to previous work and ideas.

Being a Source of Inspiration

Another value of easily accessible historical information is in its relevant and inspirational content. I was surprised to discover that many articles from the 1980s and 1990s are still relevant. For example, con-

sider this quote from Bruce Shriver's introduction of the first *IEEE Software* issue in 1984:

> *Many of the challenges facing the software industry today are a direct result of our insatiable appetite for new computer-based systems applications. Others confront us simply because we have not managed to successfully solve a large number of problems that we ourselves created many years ago.*
>
> *Specifically, we still, by and large, lack the necessary methods to increase our ability to design and implement high-quality systems.*[4]

This quote still accurately summarizes current software engineering challenges.

I've come across quite a few such insightful pieces. For my daily work, I've found early work on software architecture, quality, and maintenance still insightful and inspiring. Here are three of my favorite quotes:

> *Architecture is not so much about the software, but about the people who write the software. The core principles of architecture, such as coupling and cohesion, aren't about the code. The code doesn't "care" about how cohesive or decoupled it is; if anything, tightly coupled software lacks some of the performance snags found in more modular systems. But people do care about their coupling to other team members.*[5]
>
> *We are so used to the notion that quality must take a back seat to productivity that we continue to put up with practices that we know will produce software of lesser quality.*[6]
>
> *The greater speed of technical change means that capital investment must be recovered more quickly and that enhancement and evolution consume proportionately more resources than in a slowly changing technology. This contributes to the fact that maintenance and enhancement are the dominant costs in the software life cycle today.*[7]

I particularly like the clarity of definitions and research questions in the early articles, which often have defined a new field. Knowing such early work also helps you have authority in the field.

In my practical work as a consultant, I've discovered the value of historical content as an antidote to hype. Nothing cools down a heated sales pitch about a "revolutionary" new technology more than showing the presenter a 30-year-old article describing the same or a similar concept, sometimes with empirical studies, and asking how the "new" solution differs. I've used this tactic successfully a few times. For example, people presenting a new low-code platform are often proud of the platform's use of visual programming that supposedly implements a new programming paradigm. However, as Shi-Kuo Chang's 1987 survey on visual languages shows, many such visual-programming techniques are more than 30 years old.[8]

I also came across many inspirational but less known and unexpected pieces, such as great articles from Alan Kay and Christopher Alexander:

> *You could ... say that the main business of everyone on earth is to help everyone else—including ourselves—get enlightened because the technology is getting more and more dangerous.*[9]
>
> *What I am proposing ... is a view of programming as the natural, genetic infrastructure of a living world which you/we are capable of creating, managing, making available, and which could then have the result that a living structure in our towns, houses, work places, cities, becomes an attainable thing. That would be remarkable. It would turn the world around, and make living structure the norm once again, throughout society, and make the world worth living in again.*[10]

And this just scratches the surface. Please explore these quotes yourself, and use social media to let everyone know when you find some new inspirational pieces.

Another piece of inspiration is what I call "the art of *IEEE Software*." The covers, as well as the article illustrations, depict key software engineering concepts in an original and artistically pleasing way.

Having Intrinsic Historical Value

History has value in itself. People care about it. For example, Alison Gopnik explained that acknowledging the truth about the past, good or bad, individually or collectively, is deeply important to us as humans, even when it has no immediate effect on the present.[11] I think the same concept applies to the history of software engineering. Many of us software engineering professionals find it important to acknowledge the truth about the past for its own sake, even when it has no immediate effect on what we do now.

Gopnik also noted that many parents spend much energy trying to determine their children's future. However,

parents can't give their children a good future, but they can give them a good past. That also applies to us. Can we as potential authors determine software engineering's future? Who knows? We can try. But it's not completely in our hands. And history teaches us that we have, on quite a few occasions, been wrong. For instance, Fred Brooks, in an excerpt from his book *The Mythical Man-Month* that appeared in *IEEE Software*, said, "[David] Parnas was right, and I was wrong [about information hiding]."[12] But can we give software engineering a good history? This is definitely much more under our control.

Defining *IEEE Software*'s Future

> *Who controls the past … controls the future: who controls the present controls the past.*
> —*George Orwell*

Finally, one value of maintaining an accessible website about our history is being able to see how our past impacts *IEEE Software*'s future. Orwell's quote from *1984* (don't forget that *IEEE Software* started in 1984) in many ways reflects the magazine's situation. The most obvious example is the impact factor—the frequency with which the average article or paper in a publication has been cited in particular years. Although the impact factor is based on past data, it directly influences a publication's reputation and future. A high impact factor normally attracts more high-quality contributions. High-quality articles and papers are normally cited more, which might further increase the impact factor. And vice versa: publications with a low impact factor normally attract fewer high-quality contributions, which might start a vicious cycle of decreasing impact factors.

Up to now, the *IEEE Software* history website has been an experiment. It's still a prototype, and we're still experimenting with different ways of presentation, adding new content and releasing changes frequently.

You can help contribute to this history. For example, write great new articles for *IEEE Software*. Invest the effort to find previous research and connect your research to it. Or, promote historical content in any medium—for example, by using that content in education and as inspiration in daily work.

References

1. R.L. Glass, "'Silver Bullet' Milestones in Software History," *Comm. ACM*, vol. 48, no. 8, 2005, pp. 15–18.
2. M. Fowler, "Semantic Diffusion," 14 Dec. 2006; martinfowler.com/bliki/SemanticDiffusion.html.
3. K. Beck et al., "Manifesto for Agile Software Development," 2001; agilemanifesto.org.
4. B.D. Shriver, "From the Editor-in-Chief," *IEEE Software*, vol. 1, no. 1, 1984, pp. 4–5.
5. J.O. Coplien, "Reevaluating the Architectural Metaphor: Toward Piecemeal Growth," *IEEE Software*, vol. 16, no. 5, 1999, pp. 40–44.
6. P. Hsia, "Learning to Put Lessons into Practice," *IEEE Software*, vol. 10, no. 5, 1993, pp. 14–17.
7. P. Wegner, "Capital-Intensive Software Technology," *IEEE Software*, vol. 1, no. 3, 1984, pp. 7–45.
8. S.-K. Chang, "Visual Languages: A Tutorial and Survey," *IEEE Software*, vol. 4, no. 1, 1987, pp. 29–39.
9. "Inventing the Future" (interview with A. Kay), *IEEE Software*, vol. 15, no. 2, 1998, pp. 22–23.
10. C. Alexander, "The Origins of Pattern Theory: The Future of the Theory, and the Generation of a Living World," *IEEE Software*, vol. 16, no. 5, 1999, pp. 71–82.
11. A. Gopnik, *The Philosophical Baby: What Children's Minds Tell Us about Truth, Love, and the Meaning of Life*, Farrar, Straus and Giroux, 2009.
12. F. Brooks, "The Mythical Man-Month after 20 Years" (book excerpt), *IEEE Software*, vol. 12, no. 5, 1995, pp. 57–60.

ŽELJKO OBRENOVIĆ is a principal consultant at the Software Improvement Group. Contact him at obren@acm.org.