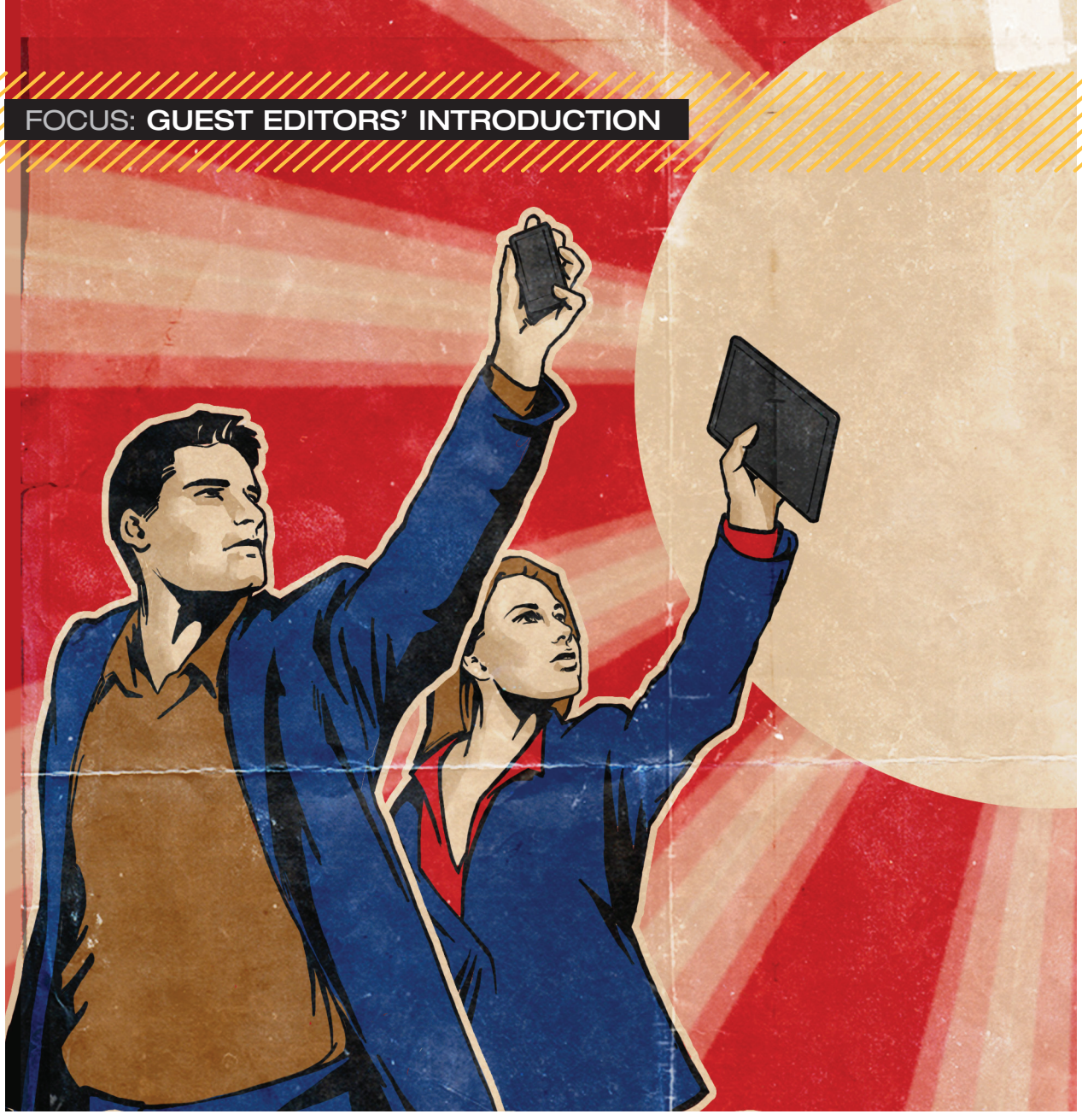

The Future of Software Engineering

Forrest Shull and Anita Carleton, Software Engineering Institute

Jeromy Carriere, Google

Rafael Prikladnicki, Pontificia Universidade Catolica do Rio Grande do Sul

Dongmei Zhang, Microsoft Research

THIS YEAR MARKS the 50th anniversary of the Turing Award, which was first given to Alan Perlis, an oft-quoted mathematician who described the relationship between humans and computers as having "a vitality like a gangly youth growing out of his clothes within an endless puberty."[1] Now that our dependence on software permeates nearly every aspect of our lives, it's time to ask ourselves where this relationship is headed and, even though software engineering is still a relatively new discipline, how much we've matured.

In many ways, we seem ready to take a seat at the grown-up table as discussions in our community increasingly revolve around the

0740-7459/16/$33.00 © 2016 IEEE

immediate, concrete concerns of affordability, dependability, privacy, and security. And there's plenty to talk about: recent headlines trumpet software flaws as causing airspace closures, credit card data theft, car and plane malfunctions, unreachable 911 centers, and billions of dollars of wasted taxpayer money. Furthermore, as the *New York Times* editorial "Volkswagen and the Era of Cheating Software" reminded us, accidental functionality shouldn't be the only concern keeping us up at night.[2] "Smart" objects can use software to cheat and lie, and objects are getting smarter all the time.

A Cause for Optimism

Yet in the face of a sometimes grim litany of software concerns, we're happy to report that our field retains at least a bit of the starry-eyed optimism of youth. This issue's call for papers generated widespread interest and resulted in twice the submissions as for any previous special issue. We believe the community's enthusiasm is as well-founded as its angst. Software already prevents car collisions, helps fight human trafficking and predict genocide, enables electronic prescriptions and health records that save thousands of lives, and has changed how doctors perform life-saving surgeries. Although software makes the biggest splash in the news when things go wrong, it's important to consider not just software's problems but also its promise and the capabilities it will provide in the future. In addition, we need to consider how our jobs as software engineers are transforming along with the services that software makes possible.

In many ways, this special issue is the product of a robust, engaged community, with record numbers of people taking the time to provide content or review submissions. We regret we could publish only a fraction of the submissions, because they all help illuminate the path forward. They touched on almost every aspect of our profession, ranging from ideas for rethinking education and training new software developers to new programming paradigms that might change how developers think about the functionalities they're implementing. With the help of the peer reviewers, we've tried to select the best of the best for you.

Perhaps most interesting was the number of submissions focusing on requirements, two of which we've included. Although requirements might seem slightly unexpected to be listed as one of the areas that will most impact our field's future, it reminds us that humans still drive this relationship. As Perlis said, "The most important computer is the one that rages in our skulls and ever seeks that satisfactory external emulator."[1] Despite the plethora of fascinating technological areas in which breakthroughs might radically change our day-to-day practice, the common theme remains that humans are still the focus. That includes both the human users of these technological breakthroughs and the human users of the new software systems that become possible. As ever, effectively managing these relationships requires diverse views so that our systems don't emulate the minds of only a few.

In such a forward-looking field, it's not surprising that this issue is one of many views on where our field is headed. We recommend placing the visions on these pages against those provided in other forums for an interesting opportunity to compare and contrast. For example, the Future of Software Engineering track at the 2014 International Conference on Software Engineering (http://2014.icse-conferences.org/fose) featured roadmaps predicting the path forward for areas that have already proven their worth. It also looked at technologies being researched and developed that might have outsize importance in the future, such as automated deduction, probabilistic programming, and the use of massive open online courses to help software education scale. *IEEE Spectrum* compiled its own thoughtful set of roadmaps regarding technologies from all fields that will change the future;[3] it's instructive to see in how many cases software provides key capabilities. And for an accessible view of the future, informed by decades of experience, it's hard to beat *IEEE Software's* department editor Grady Booch.[4] Grady's talks are always helpful reminders that we, as software engineering professionals, play a fundamental role in the advancement of the human spirit, encompassing war, commerce, the arts, science, society, and faith.

We hope this special issue, by focusing on the intersection between software engineering research and practice, adds to the dialog and provides thought-provoking and perhaps inspirational ideas about where we're headed. Although we work in a fast-moving field and often have to keep focused on the day-to-day necessities, it's important to look at today's trends and imagine where they might lead. If you feel we missed the mark, maybe it's not so much that we as a community aren't that good at predicting the future but that we haven't yet agreed on how we want it to look. Remember, as Perlis said, "In software systems it is often the early bird that makes the worm."[1] We need to ask ourselves not only what we're able to make and do but also whether

ABOUT THE AUTHORS

FORREST SHULL is the assistant director for empirical research at Carnegie Mellon University's Software Engineering Institute, where he leads work to advance the use of empirically grounded information in software engineering, cybersecurity, and emerging technologies. Shull received a PhD in computer science from the University of Maryland. He's editor in chief emeritus of *IEEE Software*. Contact him at fjshull@sei.cmu.edu.

ANITA CARLETON is the deputy director of the Software Engineering Institute's Software Solutions Division. She provides leadership for the research, development, and transition of methods and technologies that encapsulate best practices for software engineering, management, and measurement at the nation's only federally funded R&D center focused on tackling the most challenging software-related issues. She is an IEEE Senior Member and serves on the *IEEE Software* advisory board. Contact her at adc@sei.cmu.edu.

JEROMY CARRIERE is an engineering director at Google. His research interests are large-scale distributed systems, software architecture, and software engineering management. Carriere earned his bachelor of mathematics in computer science from the University of Waterloo. Contact him at jcarriere@google.com.

RAFAEL PRIKLADNICKI is an associate professor in the Computer Science School at Pontificia Universidade Catolica do Rio Grande do Sul (PUCRS) and the director of the university's Science and Technology Park (Tecnopuc). His research interests include distributed and agile software development. Prikladnicki received a PhD in computer science from PUCRS. He's on the *IEEE Software* editorial board. Contact him at rafaelp@pucrs.br.

DONGMEI ZHANG is a principal researcher and research manager at Microsoft. Her research interests are software analytics, machine learning, and information visualization. Zhang received a PhD in robotics from Carnegie Mellon University. Contact her at dongmeiz@microsoft.com.

what we're making is contributing to the future we want to see.

The Articles

In this issue you'll find a range of perspectives from professionals working around the world in diverse areas of software. The content ranges from detailed technical articles about the research areas behind today's trends to shorter essays and opinion pieces from folks working to sharpen the focus of their own visions of the future.

From the US, we offer thoughtful essays from key people who have helped build organizations that are playing important roles in building the software discipline: Andrew Moore (formerly at Google and now at Carnegie Mellon University), Tim O'Reilly (O'Reilly Media and Code for America), and Paul Nielsen and Kevin Fall (both at the Software Engineering Institute, the only US federally funded R&D center focusing on software).

Given software development's increasingly global nature, we've included contributions from outside the US. In the Perspectives—China department, four chief technology officers discuss the software industry's fast growth in China and their frontline experiences leading software development for massive systems and services in some of the fastest-growing Chinese companies. In the Perspectives—Brazil department, three executives from ThoughtWorks, a global company focusing on software design and delivery, give insights on the importance of the human side of software engineering and of transparency, emergence, and sustainable development for software engineering's future.

The six peer-reviewed articles explore innovative software practices and tools in use today and envision

how they might evolve. In "Toward Data-Driven Requirements Engineering," Walid Maalej and his colleagues consider the implications that the enormous volume of unstructured data generated through our digital interactions might have on requirements engineering. Users submit feedback in app stores, social media, and user groups, while software vendors gather massive amounts of implicit feedback through usage data. Can all this data lead to requirements engineering "by the masses, for the masses"?

In "Requirements: The Key to Sustainability," Christoph Becker and his colleagues approach requirements from another angle. As the line between software systems and socioeconomic and natural systems blurs, software systems influence aspects of our lives such as how we form relationships, how we travel, and what we buy. In this context, what does it mean to be responsible for the consequences of the software we design, and how can we establish sustainability as a major concern?

In "Reducing Friction in Software Development," Paris Avgeriou and his colleagues posit that we're producing software at such a rate that its growth hinders its sustainability. As technical debt becomes a dominant driver of progress, can we get ahead of the software quality and innovation curve by establishing technical-debt management as a core software engineering practice?

In "Crowdsourcing in Software Engineering: Models, Motivations, and Challenges," Thomas LaToza and André van der Hoek explore current crowdsourcing models and the challenges that must be met before those models can reach their full potential. Will crowdsourcing lead to fundamental, disruptive changes in how we develop software?

In "Speed, Data, and Ecosystems: The Future of Software Engineering," Jan Bosch also looks at trends in industry and society that have shaped software engineering recently. He focuses on three key trends—speed, data, and ecosystems—and their implications for software engineering's future.

Big data also plays a role in James Herbsleb and his colleagues' article, "Intelligently Transparent Software Ecosystems." As the software industry relies increasingly on open libraries, frameworks, and code fragments, can we create an infrastructure that achieves "intelligent transparency" by applying analytics to data from open source projects to bring stakeholders the information they need when they need it?

In this issue's Voice of Evidence column, Emily Hill, Philip Johnson, and Daniel Port share evidence about an "athletic" approach to software engineering education. They argue that, because of all these new technologies and new uses of software, we'll have to teach software engineering differently, and we should study which approaches can be effective.

Finally, in the Sounding Board department, George Hurlburt and Jeff Voas present an opinion piece on their vision of software engineering's future. Their idea of software engineering (whether an art or a science) morphing into "virtual philosophy" is sure to elicit reactions and generate discussion. We look forward to seeing what results.

The task we set for ourselves in this issue—presenting visions of our discipline's possible future—was challenging and exciting. We hope you'll be similarly inspired by what we've collected. Although we're sure that many of the details will be off the mark, we hope the visions in this issue inspire action—perhaps to learn more about a new technology that seems like a promising component for the hot careers of the future. Maybe even to build new capabilities or volunteer for an organization that uses software to build new capabilities, such as Code for America. But above all, to think and discuss the future we're building together.

Acknowledgments

We wish to express our extraordinary gratitude to Erin Harper, who made important contributions to several aspects of this special issue, including this introduction. Her work on the original proposal and the call for papers had a large impact on the final product.

References

1. A. Perlis, "Epigrams on Programming," *SIGPLAN Notices*, vol. 17, no. 9, 1982, pp. 7–13.
2. Z. Tufekci, "Volkswagen and the Era of Cheating Software," *New York Times*, 23 Sept. 2015; www.nytimes.com/2015/09/24/opinion/volkswagen-and-the-era-of-cheating-software.html?_r=0.
3. "The Future We Deserve, *IEEE Spectrum*, 2015; http://spectrum.ieee.org/static/the-future-we-deserve.
4. G. Booch, "ICSE 2015—Grady Booch Keynote" (video), 2015; www.youtube.com/watch?v=h1TGJJ-F-fE.

See www.computer.org/software-multimedia for multimedia content related to this article.