The Golden Age of Software Engineering

Ipek Ozkaya

SOFTWARE HAS BECOME more pervasive than we in the software engineering community ever imagined. As the amount of software in the world increases, so do the challenges of ensuring that software is cost effective and good enough to meet users’ needs. At the same time, the economic and social impact of software is also increasing. Despite many challenges, noteworthy achievements in software are being made at an accelerating pace, and software continues to be a dynamic business field.

Organizations traditionally known as manufacturing organizations are looking for ways to rapidly reorganize to increase their investments in software. Manufacturing organizations have long used software from IT systems, plant and process automation to embedded systems. They have recognized that surviving in our new fast-paced software-enabled economy requires competencies not only to develop their own software when needed but also to innovate in software. These transformations are not trivial. Look at the example of General Electric (GE). GE established its software business, GE Digital, in late 2015 with services in cloud-based software and analytics, field-service management software, cybersecurity, and digital twin and industrial machine-learning technology. After a brief period of three years, GE decided to sell this business. There are, of course, many factors, business decisions, and challenges that go into such transformations, but critical among them is the task of hiring software-engineering talent attuned to current trends. Such business transformations will likely continue to enhance the importance of people with software-engineering skills.

Meanwhile, organizations traditionally known as software organizations are investing in hardware and other commodity domains to not miss out on opportunities to reach more users. Such promising domains include phones, vehicles, wearable computing devices, smart homes, and food and household products. Brick-and-mortar businesses are being transformed into digital businesses. Amazon’s acquisition of Whole Foods is one example. I must admit that visiting the Amazon Go store in Seattle was an experience like watching The Jetsons and imagining that we are approaching the reality of flying cars and talking housemaid robots. Such business shifts are pushing the boundaries of creativity in a software-enabled world.

These are business examples, decisions made in boardrooms and driven by money, but they can be implemented only one source line of code at a time—sometimes with bugs; poor design choices; insufficient testing and quality; privacy; security; and ethics implications. Now, to address these implications, let us look at an example of how the way in which we write software has also been changing. Facebook earlier in 2018 announced the use of a tool called Sapienz, an intelligent automated testing system that introduces test cases into the software-production pipeline. Facebook reports that Sapienz has cut the time to diagnose and fix issues from days to hours and minutes. Sapienz relies on search-based software-engineering approaches to sample the space of all possible tests and select relevant ones. Search-based software engineering applies search techniques such as genetic algorithms, probabilistic

Digital Object Identifier 10.1109/MS.2018.2877032
Date of publication: 8 January 2019
search, and tabu search to software-engineering problems by stating them as optimization problems. Search-based algorithms help find “near-optimal” or “good-enough” solutions and improve the heuristics over time. The application of Sapienz in Facebook grew out of many years of research in search-based software engineering, and we are starting to see its applications in industry-scale software. We are now likely to see a rapid increase in the availability of better tools that assist in automating the routine tasks of software engineers, thus giving them more time to focus on designing their systems.

My analysis of the evolution in the field would be incomplete if I did not acknowledge the excitement, advances, and opportunities—and challenges—we are seeing in machine learning and artificial intelligence (AI). It is common knowledge that, in the past few years, exponential increases in computational power and machine learning have resulted in huge investments in AI. We can talk about the application of machine-learning algorithms to models of billions of parameters and to terabytes and petabytes of data! Decision making has a whole new meaning when driven by these algorithms! On the other hand, irrational enthusiasm and confusion are nothing new to our industry. We went through a similar experience when service-oriented architectures became possible thanks to better middleware. We continue to see similar excitement about everything in the cloud thanks to the availability of high-capacity networks, low-cost computers and storage devices, and hardware virtualization. In time, the disparity between hype and reality diminishes. We now know that we must treat data as an essential component of our systems.

The opportunities and excitement over AI-driven software systems also require different kinds of experts to contribute to the creation of software. We have seen the birth and expansion of the data-science field within the last decade. The use of the term *data science* is also relatively new. It is attributed to William S. Cleveland who, in 2001, wrote “Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics.” Today we see data science and machine learning established as undergraduate or graduate programs in many universities to help meet the need for such expertise. Data scientists write software. They are likely to make increasingly critical software contributions as machine learning and AI become more prominent parts of software-enabled systems.

Those of you who have been active in research will also be aware of the increasing emphasis on open science. Software-engineering researchers have been talking about the long-overdue and welcome initiative that 11 European funding organizations have taken by declaring that they will require the articles that they fund to be freely available from the moment of publication. The Organization for Economic Cooperation and Development states that open science must “make the primary outputs of publicly funded research results—publications and the research data—publicly accessible in digital format with no or minimal restriction.” Open science has the potential to democratize accessibility of fundamental research and accelerate scientific progress.

*IEEE Software*’s September/October 2018 issue reflected on the 50th anniversary of the coining of the term *software engineering*. In that issue, Guest Editors Hakan Erdogmus, Nenad Medivovic, and Frances
FROM THE EDITOR

EDITORIAL STAFF

Managing Editor: Jessica Welsh
Senior Managing Editor: Geraldine Krolin-Taylor
Publications Coordinator: sofware@computer.org
Multimedia Editor: Erica Hardison
Cover Artist: Andrew Baker
Publisher: Robin Baldwin
Publications Portfolio Manager: Carrie Clark
Senior Advertising Coordinator: Debbie Sims, dsims@computer.org

CS PUBLICATIONS BOARD
Greg Byrd (VP for Publications), Erik Altman, Ayse Basar Bener, Alfredo Benso, Robert Dupuis, David S. Ebert, Davide Falessi, Vladimir Getov, Avi Mendelson, Dimitrios Serpanos, Forrest Shul, George K. Thiruvathukal

CS MAGAZINE OPERATIONS COMMITTEE
George K. Thiruvathukal (Chair), Gul Agha, Irena Bojanova, Jim X. Chen, Shu-Ching Chen,Gerardo Con Diaz, David Alan Grier, Lizy Kurian John, Marc Langheinrich, Torsten Müller, David Nicol, Ipek Ozkaya, George Pelli, VS Subrahmanian

Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author’s or firm’s opinion. Inclusion in IEEE Software does not necessarily constitute endorsement by IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society’s Web-based system, ScholarOne, at http://mc.manuscriptcentral.com/sw-cs. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 4,700 words including figures and tables, which count for 200 words each.


Paulisch, in reaction to recent developments in big data, powerful commodity platforms (the cloud), the Internet of Things, cyberphysical systems, AI, and machine or deep learning, reflected that “once the hype that has accompanied each of these inevitably dies down, the opportunities for real science and engineering will still remain.” I could not agree more. All of these changes and the excitement they cause are, in the long run, good for software engineers and the systems they create: they push us to ask the hard questions and improve how we conduct our profession and how we develop our systems.

We have arrived at the golden age of software engineering! A golden age is a period of great achievements and pervasive acceptance of these achievements as they are not speculative anymore. The examples I picked here were meant to demonstrate possibilities when boundaries between different camps start diminishing to propel creativity and achievement: between software and hardware/manufacturing businesses, between manual and automated software-engineering practices, between data and software that run on that data, between science and practice. Consequently, society at large is making us increasingly aware of our collective responsibility in creating software and living in a software-enabled society. The road is not all straight, but we should watch carefully the next decade or two!

Looking Ahead
What does all of this mean for the IEEE Software community—whom we at the magazine refer to as the reflective software practitioners who comprise our target audience? Reflective practice involves reviewing one’s actions to enable continuous learning. It requires combining practical values as well as theories to inform our practice of software engineering. The rationale behind reflective practice is that experience alone does not necessarily lead to learning and progress; deliberate reflection on experience is essential. Reflective practitioners are those who are willing to step back and share their lessons learned with their peers, learn from others, and try out techniques that may seem, at first, not to have been thoroughly tested but that push the boundaries of practice. The reflective practitioner embraces change as an inevitable consequence of reflection and is passionate about finding ways to improve.

As the IEEE Software magazine community, which includes our incredible volunteer network as well as the dedicated staff that work with us at the IEEE Computer Society and IEEE Publications, we strive to be a platform for content that helps software engineers understand and reflect on recent technology changes as well as enduring fundamentals. We feature peer-reviewed content that includes examples of work that translated theory into practice, reflections on experiences, and case studies, to name a few areas of interest. And our departments, all of which are run by renowned and experienced individuals in their fields, illuminate different aspects of our industry. In addition to our digital articles, we also host Software Engineering Radio, which features discussions with industry experts on various aspects relevant to software professionals. Our software history site gives an excellent overview of our past content. All of our outlets are meant to be lasting educational resources.

At IEEE Software, we embrace change. In fact, our operating procedures are built around enabling change! With this issue of the magazine you are seeing another change, as I take the helm of the magazine from Diomidis
Spinellis, the former editor in chief. IEEE Software is a platform that offers opportunities for software engineers from all corners of our profession to contribute their time and ideas. To enable a healthy volunteer network, our system works on a rotation basis. Every two to four years, a new editor in chief is elected through an open call. All of our other volunteer roles also work on a rotation basis. We issue open calls about opportunities for volunteers to contribute as well as other opportunities, such as leading theme issues.

I will be serving in this role for the next three years beginning with this January/February 2019 issue. Diomidis Spinellis has led the magazine with incredible success since 2015 and is leaving the magazine in a great state. Needless to say, I am honored and excited to be given this opportunity. I am also aware of the challenges ahead, but I hope that if you have read this far you have already begun to understand my perspective: that challenges create opportunities to reflect and improve. My top priority will be to increase our engagement with our readers around our top-notch technical content. For

IEEE Software welcomes the following new members to our team—in particular, a number of associate editors whom we were fortunate enough to gain through our open call in October 2018.

ASSOCIATE EDITORS

Silvia Abrahão (Software Quality) is an associate professor in the Department of Computer Science at Universitat Politècnica de Valencia (UPV). Abrahão received a Ph.D. in computer science from UPV. She is a Member of the IEEE and leads the Spanish Network of Excellence on Software Quality and Sustainability. Contact her at sabrahao@dsic.upv.es.

Michael Atighetchi (Software Security) is a lead scientist in the intelligent software and systems business unit at Raytheon BBN Technologies. Atighetchi received an M.S. in informatics from the University of Stuttgart and an M.S. in computer science from the University of Massachusetts, Amherst. He is a Senior Member of the IEEE and ACM. Contact him at michael.atighetchi@raytheon.com.

Rami Bahsoon (Software Engineering Economics) is a senior lecturer in software engineering and a founding member of the Birmingham Software Engineering Research Group and the Cloud Software Engineering interest group at the University of Birmingham’s School of Computer Science. Bahsoon received a Ph.D. in software engineering from University College London. He is a Member of the IEEE and the ACM and an elected fellow of the Royal Society of Arts. Contact him at r.bahsoon@cs.bham.ac.uk.

Luciano Baresi (Distributed and Enterprise Software) is a full professor at Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria. Baresi received a Ph.D. in computer science from Politecnico di Milano and is on the editorial board of several publications. Contact him at luciano.baresi@polimi.it.
Ricardo Colomo-Palacios (Software Engineering Management) is a full professor in Østfold University College’s Computer Science Department. Colomo-Palacios received a Ph.D. in computer science from Universidad Politécnica de Madrid. Contact him at rcolomo@gmail.com.

Brian Fitzgerald (Software Engineering Process) is the director of Lero, the Irish Software Research Centre, and holds an endowed professorship at the University of Limerick. Fitzgerald received a Ph.D. from the University of London. Contact him at bf@ul.ie.

Alex Groce (Software Testing) is an associate professor at Northern Arizona University’s School for Informatics, Computing, and Cyber Systems. Groce received a Ph.D. in computer science from Carnegie Mellon University. He is a Member of the IEEE and a senior member of ACM. Contact him at agroce@gmail.com.

Foutse Khomh (Software Maintenance) is an associate professor at Polytechnique Montréal, where he heads the SWAT Lab on software analytics and cloud-engineering research. Khomh received a Ph.D. in software engineering from the University of Montréal. Contact him at foutse.khomh@polymtl.ca.

Ivano Malavolta (Software Design and Architecture) is an assistant professor at Vrije Universiteit Amsterdam. Malavolta received a Ph.D. in computer science from the University of L’Aquila. He is a Member of the IEEE, the Amsterdam Young Academy, Amsterdam Data Science, and the Dutch National Association for Software Engineering. Contact him at i.malavolta@vu.nl.

Miroslaw Staron (Software Engineering of Automotive Systems) is a professor of software engineering in the University of Gothenburg’s Department of Computer Science and Engineering. Staron received a Ph.D. from the Blekinge Institute of Technology. Contact him at miroslaw.staron@gu.se.
Mirko Viroli (Software Engineering for the Internet of Things) is a full professor in the University of Bologna’s Computer Science Department. Viroli received a Ph.D. in electronic and computer engineering from the University of Bologna. He is a member of the ACM and a Senior Member of the IEEE. Contact him at mirko.viroli@unibo.it.

Danny Weyns (Software Engineering Models and Methods) is a professor in Katholieke Universiteit (KU) Leuven’s Department of Computer Science. Weyns received a Ph.D. from KU Leuven for work on multi-agent systems and software architecture. Contact him at danny.weyns@kuleuven.be.

Stefan Wagner (Software Quality) is a full professor of software engineering and the executive director of the Institute of Software Technology at the University of Stuttgart. Wagner received a Ph.D. from the Technical University of Munich and is a Member of the IEEE, the ACM, and Gesellschaft für Informatik. Contact him at stefan.wagner@iste.uni-stuttgart.de.

ADVISORY BOARD MEMBERS

Judith Bishop is a professor of computer science at Stellenbosch University, and her research interests include programming languages, compilers, distributed systems, mobile computing, and gamification. Bishop received a Ph.D. in computer science from the University of Southampton. She received the IFIP Silver Core Award, is an ACM distinguished member, and a fellow of the British Computer Society and the Royal Society of South Africa. Contact her at jbishop@sun.ac.az.

Toshiaki Kurokawa is a standards advisor with the Japan Standards Association as well as an independent software engineer, author, translator, consultant, and design thinker. Kurokawa is on the RIETI committee on the impact of human-brain-style new AI. His relationship with AI goes back to the Fifth Generation Computer Systems project in the 1980s. He studied computational mathematics at the University of Tokyo. Contact him at toshiakikr@gmail.com.

NEW DEPARTMENT

With this issue, we introduce “The Pragmatic Designer” by George Fairbanks, a software engineer at Google. Fairbanks will be writing about topics in the intersection of implementation, design, and architecture.
me, engagement and debate are essential parts of creativity, of being a reflective practitioner: to discuss, give and receive feedback, and, consequently, improve. I look forward to hearing from you!

References

IEEE Computer Graphics and Applications bridges the theory and practice of computer graphics. Subscribe to CG&A and

- stay current on the latest tools and applications and gain invaluable practical and research knowledge,
- discover cutting-edge applications and learn more about the latest techniques, and
- benefit from CG&A’s active and connected editorial board.