



# What Should a Software Engineer Know?

Ipek Ozkaya

**A SOFTWARE ENGINEER** applies the principles of engineering to the design, development, maintenance, testing, and evaluation of a software-enabled system. While this fundamental understanding of what a software engineer does is commonly shared, the journey to understand what a software engineer should know evolves, mostly as a consequence of the rapid pace of technological changes.

Similar to all other engineering disciplines, the societal implications of software-enabled systems influence the software-engineering body of knowledge. One way we observe this influence is through the incredible pace at which advances in artificial intelligence (AI) and, in particular, machine learning (ML) are impacting software-enabled systems.

Should we be teaching budding software engineers skills that we are not already providing to ensure that they become competent in the brave new world of developing socially responsible software systems that are increasingly likely to include AI/ML components?

## Fundamental Knowledge Areas

*The Guide to the Software Engineering Body of Knowledge (SWEBOK)* describes 15 areas that are fundamental to the profession.<sup>1</sup> The knowledge areas should not surprise any practicing software engineer. They span software requirements, design, construction, testing, maintenance, configuration management, software-engineering management, software-engineering processes, models and methods (including formal and agile methods), software quality, professional practices, software-engineering economics, computing foundations, mathematical foundations, and engineering foundations.

There are natural overlaps between these areas; however, collectively they provide a reasonable coverage of the knowledge and skills that a software engineer should develop during the course of her career. On one hand the *SWEBOK* knowledge areas cover the essential competencies in software engineering. On the other hand, software-enabled systems are increasingly ubiquitous, developed and sustained by interdisciplinary teams, and integrated into all aspects of so-

ciety. Consequently, it is important to consider emerging areas of knowledge that are likely to be increasingly critical for software engineers. I argue that, given the pervasive nature of software in society and the promise of AI, software engineers, at a minimum, should have an understanding of three additional areas: data science, computing hardware, and socially responsible engineering.

## Data Science

Data science combines theories and techniques from mathematics, statistics, computer science, and information science to understand and analyze the phenomenon about data. For example, extracting knowledge and insight from patterns in information is the fundamental aspect of ML in data science.

Software engineers need to understand the basics of data science for several reasons. Increasingly, software-enabled systems include ML models and data. Developing software systems with these elements presents different software-engineering challenges. For instance, data and ML models erode at different rates than other software components. This requires software

Digital Object Identifier 10.1109/MS.2019.2946668  
Date of current version: 24 December 2019

# CONTACT US

## AUTHORS

For detailed information on submitting articles, visit the “Write for Us” section at [www.computer.org/software](http://www.computer.org/software)

## LETTERS TO THE EDITOR

Send letters to [software@computer.org](mailto:software@computer.org)

## ON THE WEB

[www.computer.org/software](http://www.computer.org/software)

## SUBSCRIBE

[www.computer.org/subscribe](http://www.computer.org/subscribe)

## SUBSCRIPTION CHANGE OF ADDRESS

[address.change@ieee.org](mailto:address.change@ieee.org)  
(please specify *IEEE Software*.)

## MEMBERSHIP CHANGE OF ADDRESS

[member.services@ieee.org](mailto:member.services@ieee.org)

## MISSING OR DAMAGED COPIES

[help@computer.org](mailto:help@computer.org)

## REPRINT PERMISSION

IEEE utilizes Rightslink for permissions requests. For more information, visit [www.ieee.org/publications/rights/rights-link.html](http://www.ieee.org/publications/rights/rights-link.html)

engineers to design systems that create data-agnostic pipelines to enable data and model changes to be incorporated with ease. In addition, software engineers need to understand how to verify and validate such systems as well as how to continuously test for them. Understanding data-science methods provides opportunities for software engineers to question the verification and validation of the outcomes appropriately and raise flags when uncertainty is high. Software engineering for ML systems needs to incorporate the discovery, management, versioning, and overall handling of data; model customization and reuse; and incorporation of general data-intensive AI components, all of which are new skill sets.<sup>2</sup>

The ability to engineer systems that address these challenges requires software engineers to have an understanding of how data may evolve. In addition, interacting with the tools and techniques that data scientists use becomes part of the responsibilities of software engineers. Increasingly, software-engineering teams integrate data scientists. A study done with Microsoft engineers observed two leading kinds of data scientists. *Polymaths* conduct all of the necessary work related to preparing and analyzing data, including some software-engineering tasks, but come from a data-science background. Conversely, *moonlighters* are software engineers, but to address an immediate need in the team, they often perform data-scientist tasks.<sup>3</sup> The blurring of the boundaries between roles and responsibilities is likely to increase, further necessitating software engineers to possess some of the fundamental data-science knowledge.

## Computing Hardware

Not all software applications are computationally demanding. How-

ever, increasing demands for security, energy efficiency, high performance, and multimodal data processing including images, video, and acoustics, necessitate focusing on hardware that the software runs on more carefully. Developing software on field-programmable gate arrays (FPGAs), multicore processors, and graphics-processing units requires unique software-engineering approaches for parallelizing algorithms and computation. And, of course, with the much-anticipated possibility of making quantum computing a reality sooner than later, the question we should ask is: how might the way we develop software for these different hardware platforms change?

Software engineers need to understand the benefits and shortcomings of the hardware for which software is developed, to better take advantage of its capabilities. For example, the research and practitioner communities are increasingly focused on the implications of developing energy-aware software and how it operates.<sup>4</sup> The way software runs is directly related to how aware engineers are of the hardware platforms it will run on, how they optimize the development of the software for those platforms, and how efficient the computation can be.

Energy-aware software is only one of the emerging scenarios that necessitate software engineers' understanding of computing hardware. There are many other examples; for instance, the benefit of FPGAs is their user-reprogrammable hardware blocks that interconnect to enable application customization. The ability to change the internal operation of the hardware to accommodate changes to a design provides flexibility, making FPGAs suitable for applications in which low latency and high processing power are desired, such as

## LOOKING AHEAD



For 2020, we have an exciting lineup of theme issues. In this issue, guest edited by Rick Kazman and Liliana Pasquale, we focus on software engineering in society. We also introduce a new column, “SE for AI,” edited by Tim Menzies. Software systems are increasingly key AI-enabled systems, such as autonomous vehicles, or have significant AI components, including web search, recommendation systems, and home appliances. AI is built with software. Engineering AI-enabled systems presents different challenges that cannot be studied in isolation from software engineering. The goal of this column is to precisely focus on this relationship.

Our March/April 2020 issue will focus on design thinking, followed by the August/September 2020 issue that will be dedicated to the AI effect: working at the intersection of AI and software engineering. We will end the year by looking closer into blockchains, smart-contract engineering, and the psychology of software engineering. Details about the calls for papers for these theme issues can be found by visiting our website, <https://www.computer.org/publications/author-resources/calls-for-papers>, and selecting *IEEE Software*.

cybersecurity, autonomous systems, and wireless communications.

### Socially Responsible Engineering

The term *socially responsible engineering* means placing the public safety and interest ahead of all other considerations. As an engineering discipline, software engineering can borrow from the thinking that has been done to advance socially responsible engineering in other disciplines. Engineering ethics, for example, is treated under three distinct categories: individual, professional, and social.<sup>5</sup> The individual and professional responsibilities of software engineers are articulated through an existing code of ethics and under the software-engineering professional-practice knowledge area of *SWEBOK*. However, we have a long way to go in better incorporating socially responsible engineering know-how into software engineering. The many recent incidents of failure—the Boeing 737 crashes<sup>6</sup>

and Equifax data breach,<sup>7</sup> to name two high-profile ones—should be wake-up calls that we need to treat the social responsibilities of software engineers as top priority.

Teaching social responsibility requires teaching software engineers to account for the impact of their work on society. Software engineers should know how to discover all of the relevant facts concerning design, development, and deployment of their systems and how the possible outcomes of their available choices may positively or negatively affect/impact society. Thinking about software engineering outside the boundaries of social responsibility simply is not an option anymore. Luckily, we do not have to start from scratch and can build on decades of thinking about the societal impact of science and engineering.

Software engineers work in interdisciplinary teams that include data scientists, hardware engineers, and domain and other experts as the problem at hand necessitates. There is a fine

## EDITORIAL STAFF

### IEEE SOFTWARE STAFF

Managing Editor: Jessica Welsh, [j.welsh@ieee.org](mailto:j.welsh@ieee.org)

Cover Design: Andrew Baker

Peer Review Administrator: [software@computer.org](mailto:software@computer.org)

Publications Portfolio Manager: Carrie Clark

Publisher: Robin Baldwin

Senior Advertising Coordinator: Debbie Sims

IEEE Computer Society Executive Director: Melissa Russell

### CS PUBLICATIONS BOARD

Fabrizio Lombardi (VP for Publications), Alfredo Benso, Cristiana Bolchini, Javier Bruguera, Carl K. Chang, Fred Douglass, Sumi Helal, Shi-Min Hu, Sy-Yen Kuo, Avi Mendelson, Stefano Zanero, Daniel Zeng

### CS MAGAZINE OPERATIONS COMMITTEE

Sumi Helal (Chair), Lorena Barba, Irena Bojanova, Shu-Ching Chen, Gerardo Con Diaz, Lizy K. John, Marc Langheinrich, Torsten Möller, David Nicol, Ipek Ozkaya, George Pallis, VS Subrahmanian, Jeffrey Voas

### IEEE PUBLICATIONS OPERATIONS

Senior Director, Publishing Operations:

Dawn M. Melley

Director, Editorial Services: Kevin Lisankie

Director, Production Services: Peter M. Tuohy

Associate Director, Information Conversion

and Editorial Support: Neelam Khinvasara

Senior Managing Editor: Geraldine Krolin-Taylor


Senior Art Director: Janet Dudar

**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.

IEEE prohibits discrimination, harassment and bullying. For more information, visit [www.ieee.org/web/aboutus/whatis/policies/p9-26.html](http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html).

Digital Object Identifier 10.1109/MS.2019.2946667

line that any professional walks between specializing her knowledge versus becoming a generalist. However, given the pervasive nature of data, advances in hardware, and the impact of software in society, we need to treat data science, computing hardware, and socially responsible engineering as key knowledge areas for software engineers and not bury them beneath other areas of expertise. 

## References

1. IEEE Computer Society, "Guide to the software engineering body of knowledge." Accessed on: Nov. 2019. [Online]. Available: <https://www.computer.org/education/>
2. S. Amershi et al., "Software engineering for machine learning: A case study," in *Proc. 41st Int. Conf. Software Engineering: Software Engineering in Practice (ICSE SEIP)*, 2019, pp. 291–300.
3. M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "Data scientists in software teams: State of the art and challenges," *IEEE Trans. Softw. Eng.*, vol. 44, no. 11, pp. 1024–1038, 2018.
4. A. Fonseca, R. Kazman, and P. Lago, "A manifesto for energy-aware software," *IEEE Softw.*, vol. 36, no. 6, pp. 79–82, 2019.
5. J. R. Herkert, "Ways of thinking about and teaching ethical problem solving: Microethics and macroethics in engineering," *Sci. Eng. Ethics*, vol. 11, no. 3, pp. 373–385, 2005.
6. L. Hatton and A.-F. Rutkowski, "'Lessons must be learned'—But are they?" *IEEE Softw.*, vol. 36, no. 4, pp. 91–95, 2019.
7. U.S. House of Representatives Committee on Oversight and Government Reform. 115th Congress. (Dec. 2018). *The Equifax Data Breach*. Accessed on: Nov. 2019. [Online]. Available: <https://republicans-oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf>



**Call for Articles**

**IEEE Pervasive Computing**

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

**Author guidelines:**  
[www.computer.org/mc/pervasive/author.htm](http://www.computer.org/mc/pervasive/author.htm)

**Further details:**  
[pervasive@computer.org](mailto:pervasive@computer.org)  
[www.computer.org/pervasive](http://www.computer.org/pervasive)

**IEEE pervasive COMPUTING**  
 MOBILE AND UBIQUITOUS SYSTEMS

Digital Object Identifier 10.1109/MS.2019.2959927