Editor in Chief: **Ipek Ozkaya**
Carnegie Mellon Software Engineering Institute,
ipek.ozkaya@computer.org

The Developer Nation

Ipek Ozkaya

DEVELOPING SOFTWARE SYSTEMS is a stakeholder-focused activity that needs to be cognizant of the priorities and constraints of the business, the users of the software, those who will need to maintain it, and the many involved in its timely, high-quality, and resource-conscious engineering. Software developers are among the key stakeholders in any software development project endeavor, yet they are also those whose needs are the most overlooked. I define a software developer as an individual who is knowledgeable about one or more programming languages and is capable of using the tools required to create a software functionality. Software developers may also be referred to as programmers, developers, and coders. In addition, all software engineers are expected to be trained in software development, while not all software developers are trained in software engineering or participate in engineering activities. Admittedly, this distinction is fuzzy at best. Software engineering skills include the ability to understand the requirements of the software to be developed, formulate the design and architecture of the software by understanding its tradeoffs, and understand the many activities that need to be executed for the successful delivery of the software. Software development and software engineering are often used interchangeably as well.

Here are some eye-opening numbers. According to a recent survey, there are 26.9 million software developers in 2021 around the world, with 4.3 million of them in North America—a number that is expected to grow to 28.7 million in 2024.[1] The U.S. Bureau of Labor Statistics[2] predicts that "Employment of software developers, quality assurance analysts, and testers is projected to grow 22% from 2019 to 2029, much faster than the average for all occupations." Furthermore, also according to the U.S. Bureau of Labor Statistics, as of December 2020, the global talent shortage in software engineering and development amounted to 40 million skilled workers worldwide, with an expectation to reach 85.2 million in a decade.[3] I suspect that these numbers include those trained as software engineers as well as others who have moved into software engineering-focused careers. To put these numbers related to software developers in a comparative context, the World Health Organization estimated about 18 million medical doctors and 39 million nurses in 2020 globally.[4] And the World Bank puts the number of teachers at 85 million worldwide: 9.4 million in preprimary, 30.3 million in primary, 18.1 in lower secondary, 14 in upper secondary, and 12.5 in tertiary education.[5]

If we interpret only by the numbers, the number of developers globally makes up a population that is larger than that of many countries, such as Romania, The Netherlands, Greece, and the Czech Republic. As the software engineering community, we continue to struggle to deliver better, faster, and cheaper software to serve businesses. However, we don't seem to reflect and ask frequently enough whether we are serving the developer nation and its needs to enable it to do its job more effectively and to develop better, faster, and cheaper software.

Of course, we need to interpret all these numbers with caution. The methods of data collection likely bias and skew the results. However, they still provide some comparative basis from which to draw observations. While we do have a significant

Digital Object Identifier 10.1109/MS.2021.3118481
Date of current version: 23 December 2021

CONTACT US

AUTHORS

For detailed information on submitting articles, visit the "Write for Us" section at www.computer.org/software

LETTERS TO THE EDITOR

Send letters to software@computer.org

ON THE WEB

www.computer.org/software

SUBSCRIBE

www.computer.org/subscribe

SUBSCRIPTION CHANGE OF ADDRESS

address.change@ieee.org
(please specify *IEEE Software*.)

MEMBERSHIP CHANGE OF ADDRESS

member.services@ieee.org

MISSING OR DAMAGED COPIES

contactcenter@ieee.org

REPRINT PERMISSION

IEEE utilizes Rightslink for permissions requests. For more information, visit www.ieee.org/publications/rights/rights-link.html

number of developers globally, the need for more qualified software engineering effort and developers is only growing. An interesting fact about developers compared to other professions is that all others are confined by the legal, geographical, and language barriers of their regions. However, all developers do "speak" programming languages and development tools, which creates a well-understood baseline of commonality and easier collaboration. While the challenges and needs of other professions are impacted significantly by their local legal and geographic context, some of the challenges developers face may be resolved at a global scale by targeted strategies.

Understanding the Gaps

Among the recent software engineering practices pioneered, we can refer to two fundamental shifts that contributed to improving the job of software developers and, consequently, the software development effort. The first one is the agile software development movement, which shined a light not only on delivered software and improved collaboration among all stakeholders, but also on the unreasonable expectations from developers. The collection of agile software development practices also emphasized the importance of a sustainable tempo of development and the 40-h week.[6] The second shift is the improved automation around continuous integration and deployment and the definition of DevOps practices, which enabled providing faster and more concrete feedback loops to developers.[7] The success of both agile software development and DevOps practices and tooling can be attributed to a number of common characteristics they have. Both agile software development and DevOps practices aim to provide a solution to the wasted effort spent by developers. In addition, by recommending alternative approaches to conducting development work, agile and DevOps approaches intended to not only focus on improved software quality, but also on improved developer workflows.

Agile software development and improved deployment processes and tool chains were both welcomed changes; however, they both had a shared unexpected consequence of overlooking design and how design effort relates to development effort.[8,9] The code executes and carries the design of the software; however, the developers do not have the tools to bridge the gap between the design and the code. Today a developer's main task is still seen as fundamentally developing and delivering the functionality, rather than developing and delivering the functionality by also staying within the bounds of the best design and architecture that supports it. Functionality and error-free code are clearly emphasized expectations from developers, a design that enables both is unfortunately not. The recognition that developers need to have design and architecture skills and that their task is not simply focused on the implementation tasks is not new;[10] however, our tools and processes have yet to catch up. The consequence of this design and architecture skill gap on software systems is that, despite an improved focus on enabling developers to conduct their tasks better, systems are still deployed abundant with low-quality and unintentional technical debt. This, in return, hinders developers from avoiding the time crunch. They continue to struggle to deliver to the quality desired with a sustainable work-week tempo.

Prioritizing the Needs

Given such a large population of software developers, coupled with an overwhelming claim for an increasing

demand for more, simply training more developers and encouraging them in this career path will neither solve the demand nor the timely quality software development problems. After all, the software engineering community knows too well that adding manpower to a late project makes it later.[11] We are already late to addressing the software development and engineering skill set gap challenges. We need a different mindset and a better understanding of how to support software developers. There are significant disparities in our level of appreciation of how developers can more easily conduct their day-to-day tasks. We need research that focuses on how to improve developer workflows.[12] There are disparities in software engineering and software development education and tooling to better equip developers to recognize design conformance gaps, unintentional technical debt, and quality issues they are injecting into systems as they are implementing them, not significantly after the fact. Software development processes that remove wasted effort have not caught up with the quality and system sustainability gaps.

The demand for more software that is driving the demand for more skilled workers to service its creation is not likely to decrease. Our best bet is to focus on supporting developers in the improved creation of software to reduce rework, support timely delivery, and manage unintentional complexity. The increased complexity of software also drives an increase in the demand for more software developer skill sets. While automated tool support is critical, researchers don't often ask: What kind of automation is most needed and likely to make a difference? We already know that a disconnect from the design, structure, and behavior of the system as a whole just exacerbates both the challenges that developers face as well as the timely delivery of high-quality software systems. We need to better understand previous successes of improved software delivery paradigms and their positive and unintended negative consequences.

Existing research is abundant with findings where developers opt to not use tools to support implementation because of the mistakes they introduce or the sheer fact that the workflows they assume do not support seamless integration with developers' tasks. Software development efficiency is overdue for another conceptual shift, similar to what happened with the introduction of agile software development processes and DevOps tool chains, to better support developers. Studying development activities with a focus on bridging design-to-code activities, understanding where workflows break, and mapping these points of failure to how they impact the accumulating technical debt, overall quality, and the release tempo of the software produced are essential research steps to take. Of course, without any doubt, initiatives to encourage more individuals to take on software development and engineering as career paths are critical. However, we cannot neglect how to better support those who are already in the trenches servicing the development and sustainment of the ever-increasing global software need. The good news is that the findings and improvements implemented are likely to have a faster global impact because of the shared commonalities in developer tools, programming languages, and processes on which the developer nation relies.

Digital Object Identifier 10.1109/MS.2021.3118480

Digital Object Identifier 10.1109/MS.2021.3130456

References

1. "EDC worldwide developer population and demographic study 2021 v1," Evans Data Corporation. https://evansdata.com/reports/viewRelease.php?reportID=9 (accessed Oct. 2021).
2. "Occupational outlook handbook software developers, quality assurance analysts, and testers," U.S. Bureau of Labor Statistics, Washington, DC, USA. Accessed: Oct. 2021. [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm
3. T. Sloyan, "Is there a developer shortage? Yes, but the problem is more complicated than it looks," Forbes Technology Council, June 8, 2021. https://www.forbes.com/sites/forbestechcouncil/2021/06/08/is-there-a-developer-shortage-yes-but-the-problem-is-more-complicated-than-it-looks/?sh=7c2fed63b8e0
4. "Global Health Workforce statistics database," World Health Organization, Geneva, Switzerland. Accessed: Oct. 2021. [Online]. Available: https://www.who.int/data/gho/data/themes/topics/health-workforce
5. "Teachers," The World Bank. Accessed: Oct. 2021. [Online]. Available: https://www.worldbank.org/en/topic/teachers
6. K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
7. H. Jez and F. David, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. London, U.K.: Pearson Education, 2011.
8. P. Abrahamsson, M. A. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Softw.*, vol. 27, no. 2, pp. 16–22, 2010, doi: 10.1109/MS.2010.36.
9. G. Fairbanks, "Why is it getting harder to apply software architecture?" *IEEE Softw.*, vol. 38, no. 4, pp. 126–129, 2021, doi: 10.1109/MS.2021.3071520.
10. M. Keeling, *Design It!: From Programmer to Software Architect*. The Pragmatic Bookshelf, 2017.
11. F. P. Brooks, *The Mythical Man-Month*. Reading, MA, USA: Addison-Wesley, 1995.
12. 12. A. Carleton et al., *Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development*, Software Engineering Institute, Carnegie Mellon Univ., 2021. [Online]. Available: https://resources.sei.cmu.edu/asset_files/Book/2021_014_001_741195.pdf