



**THIS SPECIAL ISSUE** explores the intersection of artificial intelligence (AI) and software engineering (SE), that is, what can AI do for SE, and how can we as software engineers design and build better AI systems? As AI continues to disrupt many fields, from agriculture to zoology (and everything in between), it's important to explore the essential connections between AI and SE,<sup>1-3</sup> including the following:

- *SE for AI, or the creation of AI software:*<sup>3-6</sup> How do we architect, build, maintain, deploy, test,<sup>7</sup> and verify AI software?
- *SE by AI, which refers to the application of AI to SE:*<sup>8</sup> How can AI help software engineers do their jobs better and advance the state of the practice?
- *SE with AI, meaning SE in use:*<sup>9</sup> How have applications blended AI and SE so far?
- *SE in AI, which refers to the AI landscape and its effect on SE:* How do related topics, such as AI technology investment, ethics, data collection, and security, affect the work of software developers?

## Working in a Changing Landscape

The rapidly changing AI landscape is impacting many traditional SE methods and affecting how we plan, develop, acquire, deploy, test, and verify software. Broader changes may be on the way. Many large companies are opening their AI platforms to third-party developers. With everyone in the world having access to the same powerful AI tools, what effect will this have on software development as a whole? Also, as the level of intelligence in SE solutions increases, what will the role of humans be in guiding AI to create software

systems? Ethics, too, is a growing concern. Some AI software has shown disturbing discrimination, and the rising use of deep learning means machine learning (ML) models are less explainable, making it harder to assess their reliability and trustworthiness.

Entire conference series are now dedicated to ethical considerations. (These include the Fairware series; Association for Computing Machinery Fairness, Accountability, and Transparency Conference; and the IEEE Automated Software Engineering EXPLAIN workshop series.) For more on the current status of AI, ethics, and current industrial practices, see the article by Vakkuri in this special issue.

## How Can AI Help Develop Software Better, Faster, and Cheaper?

AI as an enabling technique is already extending our ability to design, develop, and deploy software in ways that are better, faster, and cheaper than ever before. Applying AI to various SE tasks has already been shown to increase effectiveness and efficiency (see “Using AI in the SE Lifecycle”). For example, much of SE is a process of picking what task to do next, such as when agile teams reflect on a Scrum backlog to decide which tasks are needed for the next sprint. Prior to the age of AI, that process of picking the next best action was a mostly manual process. However, the more we automate SE, the more we generate software artifacts that can be explored and executed automatically using AI tools.

## The AI Infrastructure

In 2020, what's exciting about AI and SE is that not only are the tools discussed in “Using AI in the SE Lifecycle” (see Figure S1) possible, but

those tools are supported by large-scale infrastructure. Some of that infrastructure is human based. For example, Kim et al.<sup>10</sup> report that more than a thousand people at Microsoft are involved in data science activities (see also her article in this special issue). Furthermore, data science techniques are widely taught at tens of thousands of universities around the world.

Other parts of this infrastructure exist in software and hardware systems, many of which can be accessed on demand. This on-demand access allows an organization to easily scale up or down its use of AI tools. For example, readily available tools, such as R, MATLAB, scikit-learn, Weka, Keras, and jMetal, make it very easy for newcomers to quickly apply state-of-the-art techniques. Many vendors, such as Amazon Web Services, also offer easily accessible and affordable large-scale, cloud-based CPU and data storage facilities. Other vendors offer extensive support for the software tools required to field complex AI tools in industrial environments. For example, Red Hat supports its customers in the creation of containers and data processing pipelines. Such containers can be readily moved, paused, duplicated, and restarted across the cloud. For more information on this approach to AI, see the article by Benton in this special issue.

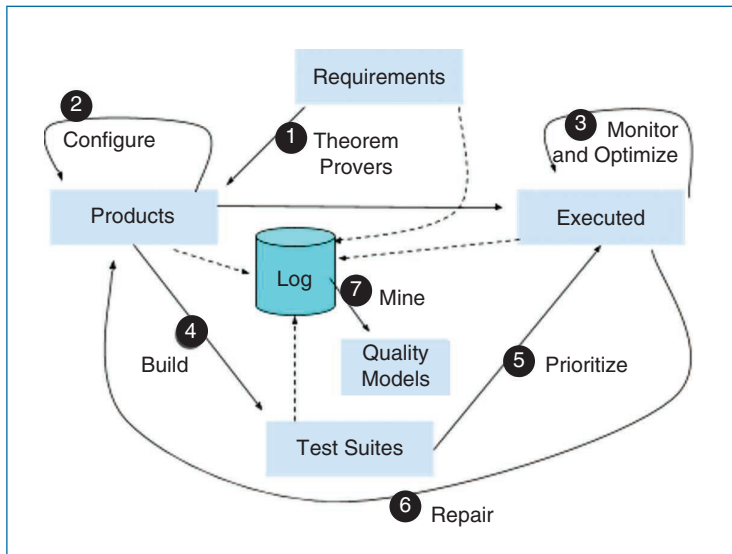
## How Can We Prepare for an AI Future?

If you and your organization are thinking about how best to combine SE and AI, we offer the following tenets for your consideration.

### Just Say “No”? (Consider Avoiding the AI Hype)

Ensure that you have a problem that both can and should be solved by AI.<sup>11</sup>

## USING AI IN THE SE LIFECYCLE



**FIGURE S1.** After describing a space of options,<sup>S1</sup> theorem provers can explore requirements to extract ① the possible products (i.e., systems with mutually compatible requirements that can coexist together).<sup>S2</sup> Then, once a system is running, developers face numerous tuning options. At that time, tools, such as multiobjective genetic algorithms, can find and configure ② that software.<sup>S3</sup> Once configured and executed, software can be monitored and optimized ③ by artificial intelligence (AI) tools that reduce the cloud resources needed for that software.<sup>S3</sup> (For more of these kinds of AI configuration tools, see the article by Kaltenecker in this special issue.) Also, when sharing software, teams find it useful to build ④ test suites that check to see if anyone's changes have hurt the system. AI tools can learn test suites that can reach all parts of a software system.<sup>S4</sup> For large test suites, it can be slow and expensive to run all of those tests each night in a cloud environment. To reduce that cost and give developers faster feedback on problematic tools, AI tools can learn how to prioritize ⑤ tests that are most likely to fail.<sup>S5</sup> (For a specific example of AI-enabled testing, see the article by Zhang et al. in this special issue.) Once systems are running and test case results are available, then AI can offer much support to the software development process. For example, AI tools can automatically find and repair ⑥ buggy code.<sup>S6</sup> Finally, if we log all of these activities, AI tools can learn ⑦ quality models that predict features about these systems. For example, AI tools can study code repository systems, such as GitHub (and their issue tracking systems), to learn quality models that predict software development time, bug locations,<sup>S7</sup> how long issues will take to resolve, antipatterns in software development,<sup>S8</sup> and much more.

### References

- S1. I. J. Jureta, A. Borgida, N. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proc. 18th IEEE Int. Requirements Engineering Conf.*, 2010, pp. 115–124. doi: 10.1109/RE.2010.24.
- S2. M. Mendonca, A. Wąsowski, and K. Czarnecki, "SAT-based analysis of feature models is easy," in *Proc. Int. Software Product Line Conf.*, 2009, pp. 231–240. doi: 10.1145/1753235.1753267.
- S3. M. Linares-Vásquez, G. Bavota, C. Cárdenas, R. Oliveto, M. Di Penta, and D. Shybyanyk, "Optimizing energy consumption of GUIs in Android apps: A multi-objective approach," in *Proc. 2015 10th Joint Meeting Foundations of Software Engineering*, 2015, pp. 143–154. doi: 10.1145/2786805.2786847.
- S4. R. Dutra, K. Laeuffer, J. Bachrach, and K. Sen, "Efficient sampling of SAT solutions for testing," in *Proc. 40th Int. Conf. Software Engineering*, 2018, pp. 549–559. doi: 10.1145/3180155.3180248.
- S5. Z. Yu, F. Fahid, T. Menzies, G. Rothermel, K. Patrick, and S. Cherian, "TERMINATOR: Better automated UI test case prioritization," in *Proc. 27th ACM Joint Meeting European Software Engineering Conf. Symp. Foundations of Software Engineering*, 2019, pp. 883–894. doi: 10.1145/3338906.3340448.
- S6. C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A generic method for automatic software repair," *IEEE Trans. Softw. Eng.*, vol. 38, no. 1, pp. 54–72, Jan.–Feb. 2012. doi: 10.1109/TSE.2011.104.
- S7. C. Bird, P. Rigby, E. Barr, D. Hamilton, D. German, and P. Devanbu, "The promises and perils of mining git," in *Proc. 6th IEEE Int. Working Conf. Mining Software Repositories*, 2009. doi: 10.1109/MSR.2009.5069475.
- S8. F. Arcelli Fontana, V. Mika, Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empirical Softw. Eng.*, vol. 21, no. 3, pp. 1143–1191, June 2016. doi: 10.1007/s10664-015-9378-4.

### Put Users Before Algorithms

Understanding algorithms is great, but understanding your users and their data needs is even greater.

- Do you spend more time talking to users than fiddling with your AI software?
- Do you understand what decisions users need to make and what data they own?
- Have you included highly integrated subject matter experts, data scientists, and data architects in your SE teams?<sup>10</sup>

### Fret About Ethics

The pace of change in AI tools seems to have left behind legal and political institutions. Ethics is now a first-class design consideration in AI systems. In 2020, it is up to us (the creators of those AI tools) to take responsibility for our creations. Therefore, we ask the following questions.

- Do you consider the ethical consequences of capturing software users' data and have actions for removing and securing these data?
- Do you treat ethics as both a software design consideration and a policy concern?
- Can you generate explanations from your system such that other people can understand how the conclusions are reached and how they might change those conclusions?
- For the people mentioned in the data you use,
  - did they give permission for you to use their data?
  - are those data safe (not exposed to public scrutiny)?
  - are the data reliable (minimal errors or subjective judgment)?

- are the identities of those people anonymized?
- do those people have the right to be forgotten (i.e., the records for individuals can be found and removed from your data sets)?
- Do you check for inappropriate bias in the conclusions about subgroups within your population? Do you apply any kind of bias mitigation strategies, where possible, to reduce that bias? Such strategies could be implemented as manual processes (e.g., a bias review board), automatic processes,<sup>12</sup> or both.

### Plan for Scale

In any industrial application, the data mining method is repeated multiple times to either answer an extra user question, make some enhancement or bug fix to the method, or deploy it to a different set of users.

- Are you using a large data mining toolkit that supports many kinds of learning algorithms? Successful AI engineers routinely try multiple AI methods.
- Data ingestion, cleansing, protection, monitoring, and validation are necessary for engineering a successful AI system. Are you double- and triple-checking that you are using the right data?
- Have you scripted your entire workflow? Developing AI apps is an agile process that requires experimentation and discovery, mistakes and repairs, folly—and then insight.
- Have you escaped from pretty graphical user interfaces (GUIs)? GUI tools are great for very rapid prototyping, but to support iteration and repeated

experimentations, move to scripting tools as soon as possible that let you automatically repeat your analysis again and again and again.

- When deploying AI for real-time use in your software, do you have plans for testing and managing your bots to ensure that they 1) can adapt for changes and 2) cannot be manipulated with false data to behave differently?

### Make It Better, Better, Better

There are so many options when building an AI system. Once you've got it working, you know there are ways to make it work better.<sup>13</sup>

- Did you incorporate user experience and interaction to constantly validate and evolve models and architecture?
- Do you define checkpoints to account for the potential needs of recovery, traceability, and decision justification?
- Did you try, several times, to make it better, better, better? Once version 1 worked better than some simplistic baseline version 0, did you then seek version 2 that is better than version 1? (Why, you might ask, should you seek the version 2 that is better, better, better than the baseline version 0? Well, there are many ways to configure a learner, but after making a learner better twice, there are exponentially fewer ways<sup>14</sup> to make that code better, better, better.)
- Can you tell when some learner is better than another? Important outcomes are riding on your conclusions. Make sure that you check and validate them. Have you statistically tested your methods? If you think one

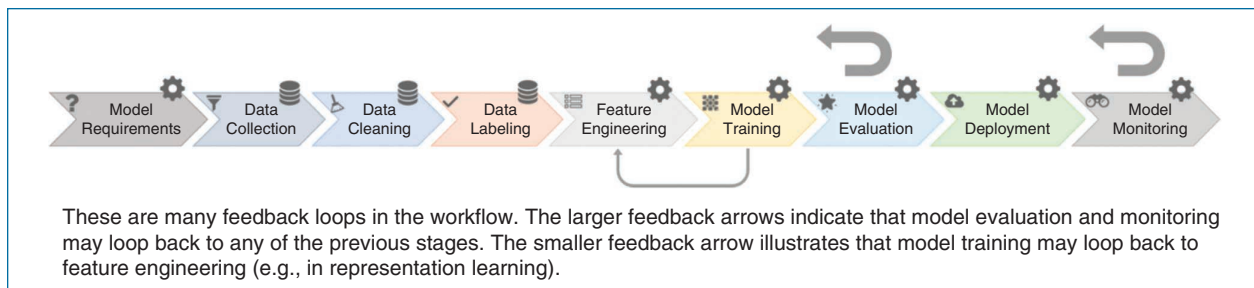


FIGURE 1. The nine stages of the ML workflow.<sup>5</sup>

learner is better than the other, then check it across several data sets (or one data set sampled 10 times, 90% at random) using statistical tests.

- Have you tamed the wriggle? If the variance of your conclusion is high, you will never be able to distinguish good from better learning methods. Have you taken steps to tame the wriggle? For example, have you applied discretization to reject trivially small differences between numbers? Have you used feature selection to prune spurious attributes? Do you use instance selection to remove repeated data (e.g., cluster and return only a small number of samples per cluster)?
- Have you applied any automatic hyperparameter optimization tools to any part of your workflow?<sup>15</sup>

### Be Kind to Your Software

AI software is still software. Do you apply best practices to your AI software? Consider these specific questions.

- On your project teams, for every one inductive/knowledge engineer, are there at least two software/test/systems engineers?<sup>10</sup>
- Is all your software in a version control tool? When you make

your software, is that process automated? If your system has configuration parameters, are those under version control?

- Do you have automated unit and system tests? Do you make and run several times per day? Per week? Do you have any test case prioritization scheme?
- Do you estimate your time to complete tasks before you do them? Do you track your time as you do tasks? Do you reflect on tracked versus estimated time in order to improve future estimates?
- Do you know the tools everyone else says are the best? Do you use them?

### Keep It Agile—Really

Intelligence means adaption. Complex AI applications are an exploration of possibilities, not a walk down a familiar path. In terms of traditional SE lifecycle steps, finding out what models, possibilities, and constraints exist in a domain is more like requirements engineering than a waterfall process.<sup>16</sup> Ask yourself the following.

- Are you set up to get early feedback on your initial approach? Continuous and early feedback from users allows needed changes to be made as soon as

possible (e.g., when assumptions don't match the users' perception) and without wasting heavy up-front investment.

- With that feedback, are you allowed to change the goals of the project?
- Have you avoided the agile fall trap (in public, you say you are agile, but, in private, you work to fixed time and deliverables)?

### Apply AI to Something You Know

The best way to learn AI is to apply it to something you know very well—like your own work!

- Are you defining and following your own AI software process? Writing software is more than just writing code, and writing AI software is more than just running a learner. For example, software analytics researchers at Microsoft offer a nine-step process for their own AI work (see Figure 1). Some stages are data oriented (e.g., collection, cleaning, and labeling), and others are model oriented (e.g., model requirements, feature engineering, training, evaluation, deployment, and monitoring).
- What is your process? Are you tracking your activity within that process?



- Do you apply data-driven decision making to your own processes? Specifically, do you apply any logging/ML/data analytics techniques to find issues and opportunities with your own process?
- Are you AI enabling (all of) your software and tools (e.g., making all data available in real time in reachable formats)?

### Train for AI

In such a fast-moving field, are you being left behind?

- Do you train your engineers in AI?
- Do you teach your own R&D staff sufficiently to utilize these new technologies to improve their own work?
- Do you have any sort of reading group at work (formal or informal) where you and your colleagues maintain a watch over the latest changes in AI?

### Organize for AI

There are different models—such as centralized, distributed, and hybrid—but, ultimately, a hybrid model emphasizing cross-functional collaboration may make the most sense. Consider the following.

- Figure out how humans and computers can build off each other's strengths.
- Understand the importance of data, training, and algorithms. AI algorithms are not natively intelligent. AI starts with naked algorithms that become intelligent only upon being trained on large amounts of data and, for most business applications, large amounts of company-specific data.

### SE Principles Apply to AI Engineering

An AI system is a software-intensive system, and the established principles of designing and deploying quality software systems that meet their mission goals on time still apply. Teams should follow modern SE and systems engineering practices as well as guidelines, such as those in the Defense Innovation Board's "Software Is Never Done" study.<sup>17</sup> Teams should strive to deliver functionality on time and with quality, design for architecturally significant requirements (such as security, usability, reliability, performance, and scalability), and plan for sustaining the system for its entire lifetime. Also, include highly integrated subject matter experts, data scientists, and data architects in your SE teams.

### In This Issue

To better understand AI and SE, we invited contributions for this special issue that document the state of the art and reflect on open issues and challenges. From numerous submissions, we selected the following most compelling articles, which cover a broad range of important issues about SE and AI. In "Software Engineering for Data Analytics," Kim talks about industrial applications of AI and how SE practices (such as debugging) need to change to better support that kind of analysis. She notes that it is inherently challenging to define what should be a correct behavior for heuristics-based, probabilistic, and predictive analytics. Hence, we need better, easy-to-extend, easy-to-use specification techniques to facilitate debugging and testing.

Kim's article can be regarded as an application view of AI. For a systems view, we turn to "Machine Learning Systems and Intelligent Applications."

Here, William C. Benton from Red Hat discusses systems issues relating to the deployment of AI tools. He argues that tools, such as Kubernetes, are well suited to taming the complexity of such workflows since they support declarative deployments, improved observability, and a single management interface for all applications.

No discussion on the current industrial impact of AI is complete without a thorough examination of how to manage the ethical implications of this technology. Vakkuri et al. discuss AI and ethics in "The Current State of Industrial Practice in Artificial Intelligence Ethics." High-level guidelines and tools for managing AI ethics have been introduced to help industrial organizations. Vakkuri et al. survey how those guidelines have been adopted in industry. Based on a survey from 166 respondents in 136 software companies, they offer many conclusions, including a list of ethical antipatterns to avoid.

Ethics is a hard problem that we are struggling to understand, but not everything about using AI is a struggle. For example, in "The Interplay of Sampling and Machine Learning for Software Performance Prediction," Kaltenecker et al. discuss a recent experience in which the interplay of sampling and ML helped configure and predict the performance of seemingly complex systems. This is a success story par excellence at the intersection between AI and SE.

Finally, we turn to an area of much current industrial interest. In "Deep Learning-Based Mobile Application Isomorphic GUI Identification for Automated Robotic Testing," Zhang et al. show how deep learning methods can improve our ability to test robotic applications for mobile applications.

## AI Around the World

Attention to AI is growing across the entire world today, driven by increased computational power, advances in deep learning, and widespread access to powerful AI tools from the world's leading companies. That means it's also rapidly becoming part of the everyday work of software engineers. In this infographic, we paint a broad picture of how some top AI players are investing in the promise of AI.

### References

Jacques Bughin, Jeongmin Seong, James Manyika, Lari Hämäläinen, Eckart Windhagen, and Eric Hazan, Tackling Europe's Gap in Digital and AI, McKinsey Global Institute, February 2019.

China Software Industry Association, www.csia.org.cn.

Owen Churchill, China's AI Dreams, *Nature*, January 2018.

Daxx, How Many Software Developers Are in the US and the World in 2019? February 2020.

Daniel Faggella, The State of AI in Montreal, *Emerj*, May 2019.

Josh Kolm, How Canadian Companies Rank on AI Adoption, *Strategy Online* September 2018.

Creig Lamb and David Rubinger, Stacking Up: A Snapshot of Canada's Developer Talent, Brookfield Institute, December 2017.

Jeff Loucks, Tom Davenport, and David Schatsky, State of AI in the Enterprise, 2nd Edition, Deloitte Insights, 2018.

McKinsey Global Institute, Artificial Intelligence in the United Kingdom: Prospects and Challenges, June 2019.

Raymond Perrault, Yoav Shoham, Erik Brynjolfsson, Jack Clark, John Etchemendy, Barbara Grosz, Terah Lyons, James Manyika, Saurabh Mishra, and Juan Carlos Niebles, The AI Index 2019 Annual Report, AI Index Steering Committee, Human-Centered AI Institute, Stanford University, December 2019.

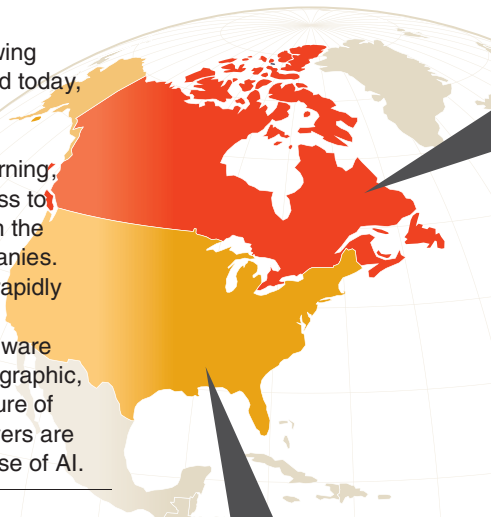
Christopher Reynolds, Canada Needs to Commercialize AI, *CBC*, February 2019.

Peter van der Schaft, Rounding up European AI Policies and Developments, *Robotics Business Review*, February 25, 2020.

Tortoise Media, Global AI Index, December 2019.

Remco Zwetsloot, James Dunham, Zachary Arnold, and Tina Huang, Keeping Top AI Talent in the United States, Center for Security and Emerging Technology, Georgetown University, December 2019.

Carnegie Mellon University Software Engineering Institute  
 Editor: Erin Harper  
 Designer: Kurt Hess



 **Canada**

**Pan-Canadian Artificial Intelligence (AI) Strategy**

The first national AI strategy to be released, issued with \$125 million in government investment in 2017. A primary focus: developing and attracting highly skilled AI talent, with \$86.5 earmarked for that purpose.

 **United States**

**American AI Initiative**

A high-level strategy created by executive order in 2019, but only one of many government AI initiatives. The Defense Advanced Research Projects Agency (DARPA) alone is investing \$2 billion in its "AI Next" campaign.

**United States**

**4.2 million** software developers

**\$36.5 billion** in private AI investment

**68%** of companies perceive the AI talent gap as moderate to extreme

**850,000** AI professionals, more than any other country

**75,000** graduate students in AI each year

**2/3** Graduate students are international students

**Canada**

**480,000** software developers

**\$1.9 billion** in private AI investment

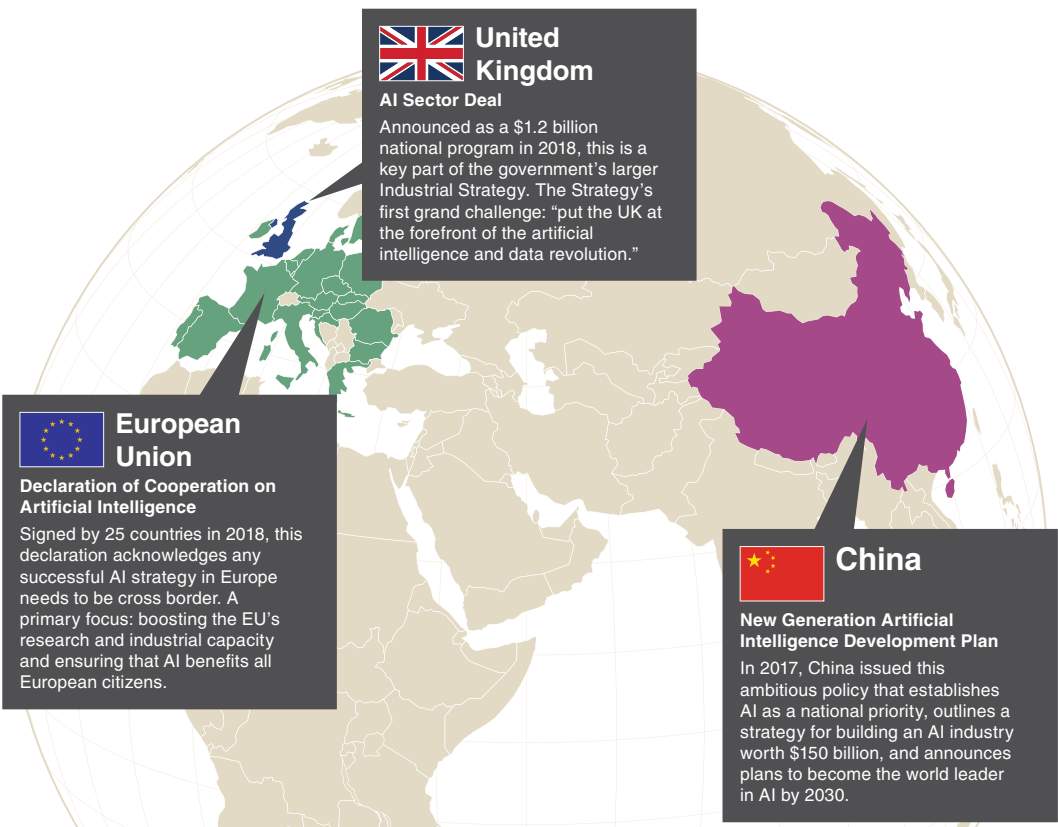
**73%** of businesses have ethics committees for AI

**50%** of ML patents in Canada have left Canadian hands from 2007–2017

**Montreal** has highest concentration of deep learning students and researchers worldwide

**51%** of executives believe AI will transform their company in next 3 years (lowest of countries surveyed)

FIGURE 2. AI around the world.



 **United Kingdom**  
**AI Sector Deal**  
 Announced as a \$1.2 billion national program in 2018, this is a key part of the government's larger Industrial Strategy. The Strategy's first grand challenge: "put the UK at the forefront of the artificial intelligence and data revolution."

 **European Union**  
**Declaration of Cooperation on Artificial Intelligence**  
 Signed by 25 countries in 2018, this declaration acknowledges any successful AI strategy in Europe needs to be cross border. A primary focus: boosting the EU's research and industrial capacity and ensuring that AI benefits all European citizens.

 **China**  
**New Generation Artificial Intelligence Development Plan**  
 In 2017, China issued this ambitious policy that establishes AI as a national priority, outlines a strategy for building an AI industry worth \$150 billion, and announces plans to become the world leader in AI by 2030.

**European Union**  
**4.7 million** software developers  
**\$6 billion** in private AI investment  
**25%** of the world's AI startups  
**2** countries in the worldwide digital top 30  
**74%** of citizens think new technologies will do more harm than good  
**53%** of countries list "regulatory requirements" as their top business risk

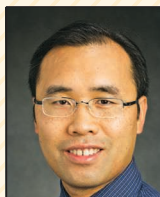
**United Kingdom**  
**814,000** software developers  
**\$2.8 billion** in private AI investment  
**#1** in Europe for density of medical AI startups  
**1,330** days to issue a patent, 3x longer than China  
**#3** in the world for raising AI investment  
**\$310 million** devoted to using AI for National Health Service

**China**  
**5.8 million** software developers  
**\$25 billion** in private AI investment  
**46%** of organizations have comprehensive strategies for AI (the most of all countries surveyed)  
**840 million** Internet users generate the world's largest supply of data  
 Received **85%** of facial recognition patents worldwide  
**190%** growth in patents over 5 years





**ANITA D. CARLETON** leads the Software Solutions Division at Carnegie Mellon University's Software Engineering Institute. She provides leadership for designing, building, and deploying advanced software technology solutions for national defense and security. Carleton received her bachelor's degree in applied mathematics from Carnegie Mellon University and her M.B.A. from the Massachusetts Institute of Technology and serves on the IEEE Software Advisory Board. Contact her at [adc@sei.cmu.edu](mailto:adc@sei.cmu.edu).



**TAO XIE** is a chair professor in the Department of Computer Science and Technology at Peking University, China. His research interests include software testing, program analysis, software analytics, software security, and intelligent software engineering. He is a Fellow of the IEEE and American Society for the Advancement of Science and an Association of Computing Machinery distinguished scientist. Contact him at [taoxie@pku.edu.cn](mailto:taoxie@pku.edu.cn).



**ERIN HARPER** is a communications strategist with more than 20 years of experience working in software engineering research. Her primary focus is on writing about technical and research results in a way that allows everyone to understand the potential effects of technical progression on their lives. Harper received her master's degree in technical communication from Carnegie Mellon University and has published work on topics such as system complexity, software measurement, data assurance, and machine learning. Contact her at [eah@sei.cmu.edu](mailto:eah@sei.cmu.edu).



**SIGRID ELDH** leads research in software test technology, quality, and debugging for Ericsson AB in Stockholm, Sweden, and she is also an adjunct professor at Carleton University and a senior lecturer at Mälardalen University. Her research interests include developing technologies for quality systems, with a focus on self-healing and autonomous systems. Eldh received her Ph.D. in test design from Mälardalen University. She has more than 35 years of experience in the software and telecom industry and in government agencies. Contact her at [sigrid.eldh@ericsson.com](mailto:sigrid.eldh@ericsson.com).




**TIM MENZIES** is a full professor in computer science at North Carolina State University, where he teaches software engineering (SE), automated SE, and foundations of software science. He is the director of the RAISE lab (Real World AI for SE) that explores SE, data mining, artificial intelligence (AI), search-based SE, and open access science. Menzies received his Ph.D. from the University of New South Wales in 1995. He is a Fellow of the IEEE. Contact him at [timm@ieee.org](mailto:timm@ieee.org) or <http://menzies.us>.



**MICHAEL R. LYU** is a professor and chair of the Computer Science and Engineering Department in The Chinese University of Hong Kong. He is a Fellow of the IEEE, American Society for the Advancement of Science, and Association for Computing Machinery. He was named among the AI 2000 Most Influential Scholars in 2020. Contact him at [lyu@cse.cuhk.edu.hk](mailto:lyu@cse.cuhk.edu.hk).



We also prepared a special feature with interviews from internationally recognized AI experts in academia and industry to gather their perspectives about important focus areas for AI, the ways AI will affect the way we work as software developers, and their hopes for AI in the future; see the article on p. 87 in this issue. Finally, as we prepared this special issue, it became clear just how much AI funding and interest is spreading across the globe. To put the articles and interviews in this issue into context, an infographic (Figure 2) is included that paints a broad picture of how some top AI players are investing in the promise of AI. 

### Acknowledgments

This special issue would not have been possible without the many contributing authors, support of our reviewers, and *IEEE Software* editorial staff. We thank them for their hard work in bringing the best of today's SE world to print.

### References

1. M. Harman, "The role of artificial intelligence in software engineering," in *Proc. IEEE 2012 First Int. Workshop Realizing AI Synergies in Software Engineering (RAISE)*, Zurich, Switzerland, 2012, pp. 1–6.
2. T. Xie, "Intelligent software engineering: Synergy between AI and software engineering," in *Proc. Symp. Dependable Software Engineering Theories, Tools, and Applications (SETTA 2018)*, Beijing, China, Sept. 2018, pp. 3–7.
3. T. Menzies, "The five laws of SE for AI," *IEEE Softw.*, vol. 37, no. 1, pp. 81–85, Jan.–Feb. 2020. doi: 10.1109/MS.2019.2954841.
4. G. Hulten, *Building Intelligent Systems: A Guide to Machine Learning Engineering*. New York: Apress, 2018.
5. S. Amershi et al., "Software engineering for machine learning: A case study," in *Proc. IEEE/ACM 41st Int. Conf. Software Engineering: Software Engineering in Practice (ICSE-SEIP 2019)*, May 2019, pp. 291–300. doi: 10.1109/ICSE-SEIP.2019.00042.
6. J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Trans. Softw. Eng.*, to be published. [Online]. Available: <https://arxiv.org/pdf/1906.10742.pdf>
7. T. Zhang, C. Gao, L. Ma, M. R. Lyu, and M. Kim, "An empirical study of common challenges in developing deep learning applications," in *Proc. 30th IEEE Int. Symp. Software Reliability Engineering (ISSRE)*, 2019, pp. 104–115.
8. M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A survey of machine learning for big code and naturalness," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 81:1–37, 2018. doi: 10.1145/3212695.
9. Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?" *Proc. IEEE Trans. Softw. Eng.*, to be published.
10. 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, Nov. 2018. doi: 10.1109/TSE.2017.2754374.
11. A. Horneman, A. Mellinger, and I. Ozkaya, *AI Engineering: 11 Foundational Practices*. Pittsburgh: Carnegie Mellon Univ. Software Engineering Institute, Sept. 2019.
12. R. K. E. Bellamy et al. "Think your artificial intelligence software is fair? Think again," *IEEE Softw.*, vol. 36, no. 4, pp. 76–80, July–Aug. 2019. doi: 10.1109/MS.2019.2908514.
13. D. H. Wolpert, "Ubiquity symposium: Evolutionary computation and the processes of life: What the no free lunch theorems really mean: How to improve search algorithms," *Ubiquity*, vol. 2013, pp. 1–15, Dec. 2013. doi: 10.1145/2555235.2555237. [Online] Available at: <https://ubiquity.acm.org/article.cfm?id=2555237>
14. G. D. Montanez, "Bounding the number of favorable functions in stochastic search," in *Proc. IEEE Congr. Evolutionary Computation*, Cancún, México, 2013, pp. 3019–3026. doi: 10.1109/CEC.2013.6557937
15. C. Tantithamthavorn, S. McIntosh, A. E. Hassan, K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *Proc. 38th Int. Conf. Software Engineering (ICSE '16)*, New York, 2016, pp. 321–332. doi: 10.1145/2884781.2884857.
16. C. Kaestner, "Machine learning is requirements engineering—On the role of bugs, verification, and validation in machine learning," March 3, 2020. [Online]. Available: <https://medium.com/@ckaestne/machine-learning-is-requirements-engineering-8957aee55ef4>
17. M. McQuade, R. M. Murray, G. Louie, M. Medin, J. Pahlka, and T. Stephens, "Software is never done: Refactoring the acquisition code for competitive advantage," Defense Innovation Board, U.S. Department of Defense, version 3.3, Mar. 12, 2019. [Online]. Available: [https://media.defense.gov/2019/Mar/14/2002101480/-1/-1/0/DIB-SWAP\\_STUDY\\_REPORT\[DRAFT\]\\_LAST%20MODIFIED\\_13MAR2019.PDF](https://media.defense.gov/2019/Mar/14/2002101480/-1/-1/0/DIB-SWAP_STUDY_REPORT[DRAFT]_LAST%20MODIFIED_13MAR2019.PDF)