

# Artificial Intelligence and Software Engineering: Are We Ready?

**Atif Mashkoor**, Johannes Kepler University

**Tim Menzies**, North Carolina State University

**Alexander Egyed**, Johannes Kepler University

**Rudolf Ramler**, Software Competence Center Hagenberg





Artificial intelligence and software engineering complement each other in various ways. This special issue highlights how this relationship is developing over time to address the challenges faced in modern-day computing.

Artificial intelligence (AI) has disrupted all walks of life. According to a recent Ipsos survey<sup>16</sup> for the World Economic Forum, on average, six out of 10 adults expect that products and services using AI will profoundly change their daily life in the next three to five years. This change is due to the incorporation of AI into products and services such as recommender systems; incorporation into workflows and processes such as the rise of bots for automated tasks; and advanced analytical capabilities, including identifying optimal resources for tasks.

According to the latest report published by McKinsey, “The State of AI in 2021,”<sup>17</sup> the top use cases of AI include service operation optimization, product enhancement, supply chain management, manufacturing, and so on. One of the most notable benefits of using AI in all those areas is a significant cost decrease—up to 51% in some cases. However, one of the prime reasons for such high gains is adherence to the core and advanced best practices already rooted in software engineering (SE), for example, the use of design thinking when developing AI tools; internal testing, verification, and validation before deployment; a framework for AI model development; a lifecycle approach; and reusability. While SE is already helping AI prosper, the same is true in reverse. AI and SE can interact across iterative and agile software process lifecycles. For example, Shafiq et al.<sup>10</sup> provide an overview of how AI is disrupting various lifecycle stages of software development. Another view of the connection of AI

and SE adapted from Carleton et al.<sup>3</sup> is presented in Figure 1.

For the planning stage, AI can help find the most critical issues that need to be fixed on a priority basis<sup>11</sup> and help agile project management, including identifying and refining backlog items for sprint planning and managing risks.<sup>4</sup> For the requirements stage, AI can assist with recommendation approaches for task allocation.<sup>12</sup> This is helpful for finding the most suitable resource person for a task,<sup>1</sup> evenly distributing knowledge among team members, and finding solutions that satisfy the different needs of various stakeholders.<sup>7</sup> For the design stage, patterns play an essential role. Using AI, we can identify and recognize design patterns in software through source code and user interface (UI)

layouts. For example, see the Washizaki et al. article in this issue as well as the article by Nguyen et al.,<sup>13</sup> who propose an approach to semiautomate design tasks by learning from previous UI design patterns. Another area where AI helps in designing software is development effort estimation.<sup>6,14</sup>

Developers face numerous tuning options once a system design is translated into a product. At that time, tools such as multiobjective genetic algorithms could help configure software. Once configured and executed, the software can be monitored and optimized by AI tools that reduce the cloud resources that are needed.<sup>15</sup> For more on AI tools that support better configuration, see the Apel et al. article in this issue. Also, when sharing software, teams find it helpful to build test

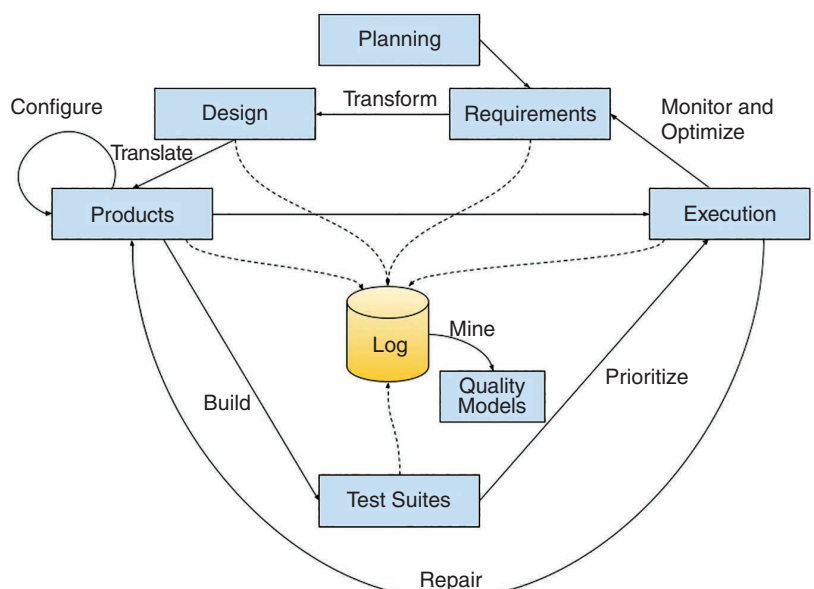


FIGURE 1. The AI-inspired SE lifecycle.

suites<sup>5</sup> that check to see whether anyone's changes have hurt the system. AI tools can learn test suites to reach all parts of a software system. For more on testing, see the article by Khaliq et al. Large test suites can be slow and expensive to run nightly in a cloud environment. To reduce that cost and give developers faster feedback, AI tools<sup>2</sup> can learn how to prioritize tests that are most likely to fail.

Once systems are running and test case results are available, AI can support the software development process again. For example, AI tools<sup>9</sup> can automatically find and repair buggy code. Finally, suppose we log all these activities.

**INCREASINGLY, RISK MANAGEMENT HAS BEEN PLAYING AN ESSENTIAL ROLE IN ENGINEERING PROCESSES TAILORED TO ROBOTIC SOFTWARE AND SYSTEMS.**

In that case, AI tools can learn from quality models that predict features about these systems. For example, AI tools can study code repository systems, such as GitHub (<https://github.com/>), and their issue tracking systems to learn quality models that predict software development time, bug locations, how long issues will take to resolve, antipatterns in software development, and much more.<sup>8</sup> For an example of that antipattern kind of analysis, see the article by Adigun et al.

With so many technical options available for AI, it is reasonable to ask whether AI technologies are mature enough to use. Hence, in this special issue, we asked: "AI and SE: Are we


ready?" Based on our submissions, the answer is a resounding yes. We posted an open call to the SE community. Overall, nine articles featuring 39 authors from seven countries were submitted and rigorously screened by at least three reviewers (in multiple rounds). In the end, after several revisions, five articles were accepted. Taken together, they illustrate what is now possible with the SE of AI systems. AI is no longer a black box, incomprehensible, or unimprovable. Instead, SE knowledge of AI has increased to the point where AI tools are now open workbenches where SE can adjust and improve the software.

For example, in "Software-Engineering Design Patterns for Machine Learning Applications," H. Washizaki and colleagues explore repeated patterns in how developers use machine learning (ML) algorithms. To illustrate the approach, one such pattern ("distinguish business logic from ML model") tells future developers that it is crucial to isolate failures between the business logic and ML learning layer (to help debug ML application systems). Results like this help bridge the gap between traditional software and ML application systems (concerning design). In another work, S.J. Warrnett and U. Zdun discuss core issues in the generation of ML applications. In "Architectural Design Decisions

for the Machine Learning Workflow," they argue that bringing ML models to production is challenging, partially due to the disparity between SE and ML practices but also because of knowledge gaps at the level of the individual practitioner. Their novel architectural design decision model supports all the choices needed to bring an ML model to production.

Increasingly, risk management has been playing an essential role in engineering processes tailored to robotic software and systems. Hence, we include the article by J.B. Adigun and colleagues, "Collaborative Artificial Intelligence Needs Stronger Assurances Driven by Risks." These researchers apply a risk representation method based on the RiskML language to International Organization for Standardization requirements for industrial AI robots. This approach offers many advantages, including identifying preconditions of unsafe behavior (by exploring the worst case branches on decision trees). Another challenge is the daunting task of how to test adaptive AI systems. Z. Khaliq and colleagues address that in "Transformers for GUI Testing: A Plausible Solution to Automated Test Case Generation and Flaky Tests." In their learning-based approach, tests are generated directly from a GUI (instead of evaluations in a tedious manual and error-prone manner). Using GPT-2 as a few-shot learner, their method fine-tunes test flows generated from GUI descriptions.

Finally, in "Green Configuration: Can Artificial Intelligence Help Reduce Energy Consumption of Configurable Software Systems?," N. Siegmund and colleagues comment that largely untapped potential arises from the configuration options

a software system provides by adapting to the application scenario, workload, and underlying hardware. While humans may be daunted by the configuration options available in a system, automatic AI agents can find valuable arrangements. For example, in the article, the authors explore how AI configuration tools can explore the energy consumption of software. The editors of this special issue would like to thank the production team of *Computer* for supporting the creation of this issue. Special mention is also due to our reviewers, who processed all our submissions. Many thanks! 

## REFERENCES

1. J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 3, pp. 10:1–10:35, 2011, doi: 10.1145/2000791.2000794.
2. A. Bajaj and O. P. Sangwan, "Test case prioritization using bat algorithm," *Recent Adv. Comput. Sci. Commun. (Formerly: Recent Patents Comput. Sci.)*, vol. 14, no. 2, pp. 593–598, 2021, doi: 10.2174/2213275912666190226154344.
3. A. D. Carleton, E. Harper, T. Menzies, T. Xie, S. Eldh, and M. R. Lyu, "The AI effect: Working at the intersection of AI and SE," *IEEE Softw.*, vol. 37, no. 4, pp. 26–35, 2020, doi: 10.1109/MS.2020.2987666.
4. H. K. Dam, T. Tran, J. Grundy, A. Ghose, and Y. Kamei, "Towards effective AI-powered agile project management," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. New Ideas Emerg. Results (ICSE-NIER)*, 2019, pp. 41–44, doi: 10.1109/ICSE-NIER.2019.00019.
5. R. Dutra, K. Laeuffer, J. Bachrach, and K. Sen, "Efficient sampling of sat solutions for testing," in

## ABOUT THE AUTHORS

**ATIF MASHKOOR** is a senior research scientist at Johannes Kepler University, Austria, and the founding managing director of the Sino-Pak Center for AI at Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, 22621, Pakistan. His research interests include requirements engineering, formal methods, and artificial intelligence-inspired software engineering. Mashkoo received a Ph.D. in computer science from the University of Lorraine, France. Contact him at [atif.mashkoo@jku.at](mailto:atif.mashkoo@jku.at).

**TIM MENZIES** is a full professor of computer science at North Carolina State University, Raleigh, North Carolina, 27695, USA, where he is the director of the Real World AI for SE lab. His research interests include software engineering, data mining, artificial intelligence, and open access science. Menzies received a Ph.D. from the University of New South Wales, Australia, in 1995. He is a Fellow of IEEE. Contact him at [timm@ieee.org](mailto:timm@ieee.org).

**ALEXANDER EGYED** is a professor of software-intensive systems and heads the Institute for Software Systems Engineering at Johannes Kepler University, Linz, 4040, Austria. His research interests include software engineering, requirements engineering, and consistency checking. Egyed received a Ph.D. from the University of Southern California, Los Angeles. He is a Senior Member of IEEE. Contact him at [alexander.egyed@jku.at](mailto:alexander.egyed@jku.at).

**RUDOLF RAMLER** is a research manager at Software Competence Center Hagenberg, Hagenberg, 4232, Austria. His research interests include software engineering and testing, software analytics, and application lifecycle management. Ramler received an M.Sc. from Johannes Kepler University. Contact him at [rudolf.ramler@scch.at](mailto:rudolf.ramler@scch.at).

6. V.-S. Ionescu, "An approach to software development effort estimation using machine learning," in *Proc. 13th IEEE Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, 2017, pp. 197–203, doi: 10.1109/ICCP.2017.8117004.
7. G. Mathew, T. Menzies, N. A. Ernst, and J. Klein, "SHORT" er reasoning about larger requirements models," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, 2017, pp. 154–163, doi: 10.1109/RE.2017.31.
8. T. Menzies and T. Zimmermann, "Software analytics: What's next?" *IEEE Softw.*, vol. 35, no. 5, pp. 64–70, Sep. 2018, doi: 10.1109/MS.2018.29011035.
9. A. Mesbah, A. Rice, E. Johnston, N. Glorioso, and E. Aftandilian, "Deep-delta: Learning to repair compilation errors," in *Proc. 27th ACM Joint Meeting European Softw. Eng. Conf. Symp. Foundations of Softw. Eng., ESEC/*

- FSE 2019, 2019, pp. 925–936, doi: 10.1145/3338906.3340455.
10. S. Shafiq, A. Mashkoo, C. Mayr-Dorn, and A. Egyed, “A literature review of using machine learning in software development life cycle stages,” *IEEE Access*, vol. 9, pp. 140,896–140,920, Oct. 2021, doi: 10.1109/ACCESS.2021.3119746.
  11. S. Shafiq, A. Mashkoo, C. Mayr-Dorn, and A. Egyed, “NLP4IP: Natural language processing-based recommendation approach for issues prioritization,” in *Proc. 47th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA 2021)*, 2021, pp. 99–108, doi: 10.1109/SEAA53835.2021.00022.
  12. S. Shafiq, A. Mashkoo, C. Mayr-Dorn, and A. Egyed, “Taskallocator: A recommendation approach for role-based tasks allocation in agile software development,” in *Proc. 15th IEEE/ACM Joint Int. Conf. Softw. Syst. Process. 16th ACM/IEEE Int. Conf. Global Softw. Eng. ICSSP/ICGSE 2021*, 2021, pp. 39–49, doi: 10.1109/ICSSP-ICGSE52873.2021.00014.
  13. M. White, C. Vendome, M. Linares-Vasquez, and D. Poshyvanik, “Toward deep learning software repositories,” in *Proc. IEEE/ACM 12th Working Conf. Mining Softw. Repositories*, 2015, pp. 334–345.
  14. T. Xia, R. Shu, X. Shen, and T. Menzies, “Sequential model optimization for software effort estimation,” *IEEE Trans. Softw. Eng.*, early access, 2020, doi: 10.1109/TSE.2020.3047072.
  15. L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, “A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing,” *IEEE Access*, vol. 3, pp. 2687–2699, Dec. 2015, doi: 10.1109/ACCESS.2015.2508940.
  16. N. Boyon, “Opinions about AI vary depending on countries’ level of economic development.” Ipsos. <https://www.ipsos.com/en/global-opinions-about-ai-january-2022> (Accessed: Jan. 25, 2022).
  17. M. Chui, B. Hall, A. Singla, and A. Sukharevsky, “The state of AI in 2021.” McKinsey & Co. <https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2021> (Accessed: Jan. 25, 2022).

## Computing in Science & Engineering

The computational and data-centric problems faced by scientists and engineers transcend disciplines. There is a need to share knowledge of algorithms, software, and architectures, and to transmit lessons-learned to a broad scientific audience. *Computing in Science & Engineering (CiSE)* is a cross-disciplinary, international publication that meets this need by presenting contributions of high interest and educational value from a variety of fields, including physics, biology, chemistry, and astronomy. *CiSE* emphasizes innovative applications in cutting-edge techniques. *CiSE* publishes peer-reviewed research articles, as well as departments spanning news and analyses, topical reviews, tutorials, case studies, and more.

Read *CiSE* today! [www.computer.org/cise](http://www.computer.org/cise)



IEEE  
COMPUTER  
SOCIETY



Digital Object Identifier 10.1109/MC.2022.3150821

