AI-Assisted Software Engineering

Background

In this NESSI[1] paper we discuss the opportunities and challenges of AI-Assisted Software Engineering (AISE), which is the use of AI and machine learning to support or automate software engineering tasks. Generative AI, large language models (LLMs) and AI chatbots have the potential to radically transform software engineering, delivering potential benefits such as higher productivity and quality. AISE "*has the potential to change the software profession more than any other recent technology*"[2]. However, AISE also poses significant technical and non-technical challenges.

In each section that follows, we recommend key research and innovation (R&I) directions for AISE. They address gaps identified in the Horizon Europe Strategic Plan 2025-2027 Analysis[3].

These recommendations clearly demonstrate that AISE is more than just assisting programmers in generating code. AISE will have a profound impact on the whole software development lifecycle, from requirements engineering to operations and maintenance.

Generative AI, LLMs and AI chatbots are new tools that can help humans with various tasks. These tools will also radically change how we do software engineering in the future. AISE will let software developers focus more on the problem and the quality, and less on the details and the programmatic syntax. By further raising the level of abstraction, this will make software engineering more automated and productive. AISE will help the European software industry to compete globally and deliver high-quality software.

AI-Assisted Requirements Engineering

Requirements engineering is the process of defining and refining the needs and expectations of different stakeholders for a software system. Requirements engineering helps to establish a common understanding and agreement on the requirements among all the parties involved in the software engineering process. Requirements engineering uses various methods and techniques to support the whole requirements lifecycle, from gathering and analysing requirements to validating and managing them[4].

Opportunities

AI combined with natural language processing can assist requirements engineering activities, as many requirements are expressed in natural language[5]. AI Chatbots based on LLMs can deliver new automation opportunities. AI chatbots can improve the communication and understanding among people from different backgrounds and skill levels, and may help to overcome language barriers e.g. by allowing stakeholders to converse and provide input and feedback in their preferred language.

Challenges

AI chatbots are fascinating because they can answer natural language questions and generate rich text, but their use may also pose problems for requirements engineering. The language model behind an AI chatbot might 'hallucinate', i.e. create nonsensical text that does not match the input. This means that AI chatbots might have low fidelity. AI chatbots might give different answers to the same question, which means that they have low stability. Low fidelity and low stability can reduce the quality of the responses and affect the

[1] NESSI (Networked European Software and Services Initiative), the European association promoting research, development and innovation in the field of software, data and digital services; https://nessi.eu/

[2] Christof Ebert, Panos Louridas: Generative AI for Software Practitioners. IEEE Softw. 40(4): 30-38, 2023

[3] European Commission, Directorate-General for Research and Innovation, Horizon Europe strategic plan 2025-2027 analysis, Publications Office of the European Union, 2023, https://data.europa.eu/doi/10.2777/637816

[4] Klaus Pohl. Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, 2010

[5] Walid Maalej: From RSSE to BotSE: Potentials and Challenges Revisited after 15 Years. arXiv preprint, 2023; https://arxiv.org/abs/2304.09308

requirements engineering outcomes, leading to poor requirements. A key challenge is how to make sure that AI chatbots produce high-fidelity and high-stability responses. This might involve using specific prompt engineering that considers the requirements engineering context, as well as fine-tuning the parameters of the underlying LLMs.

Recommendations

R&I should explore how AI chatbots and LLMs can support requirements engineering activities, thereby delivering new automation opportunities and improving the communication and understanding among stakeholders from different backgrounds and skill levels, including non-technical users and customers.

AI-Assisted Code Creation and Maintenance

Code creation and maintenance are difficult because they involve designing effective algorithms and programmatic solutions, ensuring compatibility with different platforms, and fixing errors or bugs that may arise. As software becomes more complex and sophisticated, it also becomes more challenging to ensure its quality, performance and flexibility.

Generative AI tools (such as Github Copilot and Code Llama) are transforming programming, offering profound improvements in code creation, maintenance, and significantly increasing developer productivity. Their adoption in software engineering is estimated to lead to a $1.5 trillion boost in global GDP by 2030[6].

Opportunities

Generative AI tools make code suggestions and autocompletions smarter, providing developers with code snippets that they may not have thought of. Programming is being supplemented by prompt engineering to generate code from natural language prompts. Generative AI tools are becoming increasingly interactive (exemplified by the GPT Engineer, which asks clarification questions to generate better code), and collaborative (such as the AutoGen framework, which provides a multi-agent conversation framework among LLMs and humans as a high-level abstraction). This will empower software developers to refine and improve their prompts and assemble increasingly powerful LLM applications to get the desired results, ultimately enhancing computational thinking and change how coding skills are learned.

Generative AI can make code refactoring and restructuring easier and faster, enhancing performance, readability and modularity. Generative AI can also quickly generate patches and fixes for vulnerabilities and bugs, reducing the time to address critical issues. Engineers can use conversational interfaces to fine-tune and tailor AI-generated adaptations to their specific context, rather than accepting generic suggestions.

Generative AI can also help with legacy code maintenance, which is becoming more challenging as the expertise of legacy programming languages and environments fades away. Generative AI tools can analyse legacy code, find quality problems, and suggest refactoring or rewriting options (such as IBM's watsonX Code Assistant, which transforms COBOL code into Java).

Challenges

AISE offers novel possibilities for code generation and software engineering, but it also faces important difficulties. For example, LLMs can produce outputs that are not consistent or logical ('hallucinations', as mentioned above), which can affect the quality and traceability of the code. Moreover, security is a major concern, as tools such as GitHub Copilot may generate code with vulnerabilities in about 40% of the cases[7]. Also, intellectual property issues can also emerge, as some generated code may unintentionally copy licensed programs (discussed further below, under "Non-Technical Concerns in AISE"). While Generative AI can produce high-quality general-purpose code, generating domain-specific code may still be challenging.

[6] Thomas Dohmke, Marco Iansiti, and Greg Richards. "Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle." arXiv preprint, 2023; https://arxiv.org/abs/2306.15033

[7] Pearce, Hammond, et al. "Asleep at the keyboard? assessing the security of github copilot's code contributions" IEEE Symposium on Security and Privacy, 2022

Retrieval-augmented generation can enhance code generation and summarization by seamlessly pulling relevant coding patterns from relevant databases, and aid maintenance by ensuring consistent and up-to-date coding practices.

Recommendations

R&I should explore techniques including prompt engineering and retrieval-augmented generation to improve the quality and interoperability of AI-generated code, investigate how to translate, refactor and reverse engineer code using natural language prompts, and exploit generative AI models to maintain and evolve legacy systems.

AI-Assisted Quality Assurance and Testing

As our dependency on software continues to grow, we see an increasing pressure on software engineering to quickly deliver software while assuring adequate quality. The total cost of poor software quality in the US alone amounted to $2 trillion in 2020 [8]. Given this central and crucial role of quality assurance, the use of AI for quality assurance has been widely studied. Even before the launch of ChatGPT in November 2022, there were over 800 publications with a compound annual growth rate of 38% between 2018 and 2022 [9].

Opportunities

Generative AI, LLMs and AI Chatbots will help evaluate code quality and make the quality process more efficient. Some examples are grouping and ranking tests and finding parts of the code that are more likely to have bugs. AISE will offer novel possibilities for quality assurance, such as generating better test assertions, more precisely fixing programs, and more accurately pinpointing bugs introduced during software changes.

Challenges

When using Generative AI, the generated tests may contain bugs in the test inputs or the expected outcomes. Generating the expected outcomes for the tests is difficult (this is known as the test oracle problem), as it may require information that is beyond the current capabilities of AI. LLMs may produce unrealistic or erroneous outputs, such as calling non-existent functions or passing unsupported parameters to existing functions. Tests that do not have any compile or runtime errors still need to be checked for correctness. When tests fail, it is hard to determine whether the fault lies in the software itself or in the test. This calls for a systematic approach to the incorporation of AI-generated tests into the quality assurance process.

AI can help predict faults, failures and development effort in novel ways. These predictions can assist practitioners in making timely decisions. However, AI models may sometimes produce inaccurate predictions, which can result in wasting resources on false positives or overlooking important actions due to false negatives. Providing AI models with ways to express and guarantee the confidence in their outcomes would greatly improve the quality of AI-supported decision making.

Recommendations

R&I should explore how AI-generated testing and AI-supported quality assurance can be effectively integrated into the software development process, including means to verify the correctness, confidence and adequacy of AI-generated results.

[8] Herb Krasner. The cost of poor software quality in the US: a 2020 report. Consortium for Information and Software Quality (CISQ), 2021

[9] Andreas Metzger; Xhulja Shahini; Johannes Haerkötter, Klaus Pohl. A Systematic Literature Review of Machine Learning for Uncovering Software Faults and Failures, 2023; https://doi.org/10.5281/zenodo.7615631

AI-Assisted Integration and Deployment

To deliver software changes faster and more frequently, modern software development practices such as DevOps use automated Continuous Integration and Deployment (CI/CD) pipelines. This facilitates the rapid release and operation of new software versions.

Opportunities

Writing scripts for CI/CD pipelines can be complex and tedious for human developers, and thus presents itself as a promising target for Generative AI. This can be combined with novel virtualization techniques to hide the low-level details of computing, storage and networking resources, such as Function as a Service (FaaS) or Serverless Architecture, where developers only specify the resource requirements for their software without worrying about the deployment targets.

As the computing infrastructure shifts from centralized cloud to decentralized Cloud-Edge-IoT continuum, AI becomes an essential tool for managing and supporting the deployment of software on the available resources. To handle the ever-expanding computing continuum for continuous delivery, advanced learning and reasoning capabilities (such as multi-agent reinforcement learning, deep reinforcement learning, and meta reinforcement learning) offer novel opportunities to address dynamicity, complexity and uncertainty.

Challenges

AI can help automate the integration and deployment of software, but it faces many challenges, including how to: plan deployment with multiple objectives, preferences and constraints that may vary depending on the situation; capture the high-level intents of developers instead of low-level actions that may not reflect their goals; keep developers in the loop and let them review and refine deployment scripts over time; coordinate deployment agents across different locations, domains or providers to achieve a global optimal solution; and build trust between developers and AI by explaining deployment decisions and outcomes.

Recommendations

R&I should explore cutting edge AI approaches such as decentralised and federated AI, Human-AI interaction, and Deep Reinforcement Learning to facilitate continuous software integration and deployment in the decentralized Cloud-Edge-IoT continuum.

AI-Assisted Adaptation

A (self-)adaptive software system can change its own structure and behaviour at runtime. Examples include cloud systems that can scale up or down, IoT systems that can learn and act intelligently, and process management systems that can anticipate and adjust to changes.

A crucial component of an adaptive system is its adaptation logic, which specifies when and how the system should adapt itself. When developing the adaptation logic, developers must deal with design time uncertainty, which means they have incomplete information about when and how the system should adapt[10].

Opportunities

Using AI in an online fashion (e.g. online deep learning) is a new way to deal with design time uncertainty for adaptive systems. For example, by using deep reinforcement learning at run time, the system can learn from data that is only available when it is running, and thus cope with uncertainty better. Moreover, the system can benefit from predictive monitoring, which is a technique that estimates the likelihood of failures in the near future by leveraging deep learning models and complex operational data.

[10] Andreas Metzger, Clement Quinton, Zoltan Mann, Luciano Baresi, Klaus Pohl, “Realizing Self-Adaptive Systems via Online Reinforcement Learning and Feature-Model-guided Exploration”, Springer Computing, March, 2022, pp. 1–22, 2022; https://doi.org/10.1007/s00607-022-01052-x

Challenges

Online reinforcement learning has been used successfully for adaptive systems, but important challenges remain. Modern deep learning algorithms are stochastic, which means that their performance can significantly vary. Thus even if deep learning models perform well in the lab they may not perform well in a production environment. While factors that cause variance may be controlled (e.g. non-deterministic neural network layers or random weight initialization), this typically leads to performance degradation. A promising approach is to use meta-learning techniques on top of the deep learning models, such as ensemble learning, which aggregates and thereby enhances the results of multiple deep learning models.

The successful application of online reinforcement learning depends on how well the learning problem, and in particular the reward function, is defined. Software engineers need to explicitly define a reward function, which quantifies the feedback to the reinforcement learning algorithm. Ensuring that this accurately reflects the trade-off among different goal dimensions and achieves the overall learning goal is a challenge.

In addition to leveraging AI chatbots to provide natural-language and interactive explanations, and thereby support debugging of adaptive systems[11], a novel direction may be to use LLMs to generate suitable reward functions from a sufficiently concise description of the problem context and overall learning goal.

Recommendations

R&I should explore how to leverage meta reinforcement learning techniques and LLMs to facilitate the reliability and resilience of software adaptation.

AI-Assisted Security

The software supply chain consists of many elements, such as code, tools, people and processes. These elements interact with each other to create, test, deploy and maintain software products. However, this also exposes the software supply chain to various threats, as discussed in the NESSI paper on software security[12]. To protect the software supply chain from cyber-attacks, which are becoming more frequent and advanced, AI can be a useful tool; the US has launched the AI Cyber Challenge to Protect America's Critical Software[13].

Opportunities

AI can enhance efficiency, productivity and quality in software engineering and security, as discussed in previous sections. AI can help programmers develop secure code by suggesting secure code snippets. AI can generate more comprehensive test cases which improve test coverage. AI can improve patch management by detecting vulnerabilities and by predicting the impact of a patch on system stability before deployment. Other promising opportunities include analysing large amounts of data for better insights and contextual awareness (e.g. anomaly detection), learning and adapting to changing conditions and predicting new threats, and drawing conclusions and generating action plans in real-time (e.g. incident response).

AI has the potential to improve security in all areas of the software supply chain and lifecycle via software verification and validation, software composition analysis, certification and conformity assessment, risk management for software-intensive complex systems, vulnerability detection, maintenance, reporting, and threat intelligence sharing.

Challenges

AI for secure software engineering still faces many shortcomings including generation of insecure code, uncertain prediction quality, inadequate reasoning, and low transparency. High-quality datasets are crucial for training secure AI models. Curated datasets may be needed to train and to validate AI models so that they

[11] Andreas Metzger, Jone Bartel, Jan Laufer, „An AI Chatbot for Explaining Deep Reinforcement Learning Decisions of Service-oriented Systems", 21st Int'l Conference on Service-Oriented Computing (ICSOC 2023), Springer; https://arxiv.org/abs/2309.14391
[12] https://nessi.eu/nessi-paper-on-software-security/
[13] https://www.whitehouse.gov/briefing-room/statements-releases/2023/08/09/biden-harris-administration-launches-artificial-intelligence-cyber-challenge-to-protect-americas-critical-software/

generate more secure code with fewer vulnerabilities. To ensure that software engineers can comprehend, modify and update the code, AI-assisted code generation should be transparent and explainable.

Unfortunately, AI tools for software security may also be misused by hackers and cyber-criminals. Software developers and security professionals should understand the features and impacts of AI-based cyber-attacks and how to stop them. The AI tools themselves may be vulnerable and need security measures to reduce their AI-specific security threats.

AI adoption for security is limited due to integration difficulties, a shortage of skilled professionals, and concerns about accuracy, dependability, and implementation costs. AI models, such as LLMs, pose security and privacy risks themselves, e.g. model inversion attacks revealing training data, bypassing safety features and using prompt injection for harmful outputs, data poisoning causing incorrect behaviour.

Recommendations

R&I should explore robust and transparent AI approaches to deliver and maintain secure software, together with strategies for collaboration and knowledge sharing about the ethical and reliable use of AI-powered security tools.

AI-Assisted Software Lifecycle

AISE will transform software engineering by enabling higher levels of automation for various tasks, from requirements engineering to operations and legacy systems maintenance. This will improve the efficiency and speed of the software lifecycle and supply chain. Many solutions have been proposed to automate specific software engineering tasks using AI techniques. A recent survey[14] indicates that the research output on using ML for automating individual software engineering tasks shows a compound annual growth rate of 33% from 2018 to 2020.

Opportunities

Automating individual software engineering tasks is beneficial, but there is also a great potential to exploit synergies among these tasks across the whole software lifecycle and supply chain. A single quality assurance task may not be enough to ensure the desired software quality; ideally, a suitable combination of different tasks would be used, such as dynamic testing and static code analysis. Exploiting the synergies between different tasks can increase the effectiveness of the individual tasks, e.g. a good estimate of the fault density of a software component obtained by using deep learning-based fault prediction methods could be used to optimise and prioritise testing effort and budget.

Challenges

One way to leverage synergies in AISE is to use the output of one AI-based software engineering task as input for another task, e.g. using the predicted fault-proneness of a component to prioritize test cases. However, this approach may not fully exploit the potential of AI. A more interesting direction is to leverage synergies that consider the specific characteristics of AI models. An AI model can provide an explanation that makes the output of deep learning models more understandable, and thus more trustworthy, when used as input for another activity.

Recommendations

R&I should cover the whole software lifecycle and supply chain, leading to a more effective exploitation of synergies among AISE tools, thereby increasing the productivity of software engineering tasks.

[14] Simin Wang et al. Machine/Deep Learning for Software Engineering: A Systematic Literature Review. IEEE Transactions on Software Engineering, 49(3): 1188-1231, 2023

Non-Technical Concerns in AISE

Besides the technical concerns discussed above, AISE offers novel non-technical opportunities but at the same time introduces major non-technical challenges.

Opportunities

Generative AI can be used for various purposes that can improve innovation and efficiency. It can teach developers new coding skills and languages by giving them interactive suggestions[15]. It can help developers brainstorm new ways of solving problems by generating code examples[16]. Moreover, it can help developers evaluate possible outcomes, such as the commercial ROI, how their software might affect different stakeholders, and other ethical issues. It can also help developers understand the legal aspects of software engineering, such as AI, data, and platform laws, to ensure compliance and avoid IPR violations.

AI chatbots based on LLMs can provide natural-language and interactive explanations for software systems that have AI and ML components. Explainability can help software and service providers comply with relevant laws and regulations, such as the GDPR and the AI Act in the EU. Furthermore, explainability can help software and service users trust software systems by understanding how they produce their results and whether they are acceptable or not.

Challenges

Intellectual Property

AI-generated content, such as synthetic code, poses legal questions for current IP regimes which are based on human creativity[17]. How to give fair credit and licenses for the products of proprietary models with different levels of licensing is a controversial issue, especially when the models are trained on user-generated data that is taken from the internet without compensating the users. It is not clear whether AI systems which create content independently without human input can be regarded as authors. As AISE becomes more common there will be a demand for more transparency and consistency in the ownership and licensing of AI-generated content. To safeguard the rights and interests of both AISE tool developers and users, new licensing schemes and attribution standards may need to be created and harmonised. Copyright law and liability models may need to be adjusted to deal with the legal challenges caused by AI-generated code which has not involved significant human input.

Transparency

LLMs are complex AI systems that do not reveal their inner workings easily, making it hard to apply traditional methods of technology governance. Some methods have been suggested to explain parts of LLMs, but they still face many shortcomings, such as ensuring the fidelity of explanations and the difficulty of evaluating and comparing explanations. To ensure quality and security, organisations could establish internal oversight boards which review AISE tools and the underlying LLMs before they are deployed. These boards could evaluate the potential risks of biases, security breaches, and misalignment with engineering objectives. Additionally, random audits of the code generated by LLMs could help verify its stability.

Sustainability

LLMs are powerful tools but also very costly in terms of energy and resources. They can produce thousands of tonnes of CO2 equivalent during training, which may increase as models get bigger[18]. Energy consumption during inference (i.e. when generating the outcomes) becomes a concern with increased use and adoption of these AI models. To reduce the environmental impact of LLMs, potential directions may be to use green

[15] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael J. Muller, Justin D. Weisz, The programmer's assistant: Conversational interaction with a large language model for software development. In Proceedings of the 28th International Conference on Intelligent User Interfaces, 2023

[16] David Noever, Kevin Williams, Chatbots as Fluent Polyglots: Revisiting Breakthrough Code Snippets. arXiv preprint, 2023; https://arxiv.org/abs/2301.03373

[17] Giorgio Franceschelli, Mirco Musolesi, Copyright in generative deep learning. Data & Policy, 4, e17, 2022

[18] https://aiindex.stanford.edu/report/

computing infrastructure (including processors) and to reuse and leverage existing models. We can also prune large models to make them smaller and faster without losing quality[19]. While LLMs may offer benefits for sustainability, such as automating quality assurance and reducing human effort, we need more research to understand the trade-offs and best practices for LLMs across their lifecycle.

Over-reliance

The code-generation capabilities of modern coding assistants are impressive. However, the quality of the underlying LLMs and thus their generated code may deteriorate over time, requiring proper tracking and testing. LLMs should help, not replace, human and collective programming knowledge (hence the term AI-assisted). Otherwise, the risks of wrong outputs and unethical design may increase due to less human supervision and expertise. To ensure that LLMs are aligned with ethical values and social norms, they should be developed and deployed with the participation of all relevant stakeholders. Moreover, LLMs should be subject to constitutional principles and expert oversight, as well as feedback mechanisms that can improve their performance and reduce their harms. Educating people about the benefits and risks of LLMs is essential for fostering trust and awareness and preventing overreliance on these powerful AI models.

Recommendations

R&I should pursue a multi-pronged and inter-disciplinary approach to address non-technical concerns related to AISE, including intellectual property (e.g. via new standardised licensing frameworks and attribution norms), transparency (e.g. via LLM audits and oversight boards), sustainability (e.g. by understanding the environmental cost-benefit of using LLMs), and over-reliance (e.g. via training and education that addresses the changes brought about by the use of LLMs).

Editor

Andreas Metzger, paluno, University of Duisburg-Essen.

The editorial process was assisted by ChatGPT and Edge Copilot / Bing Chat.

[19] Elias Frantar, Dan Alistarh: SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot. ICML 2023: 10323-10337