Check for updates

OPEN LETTER

AI4SWEng manifest: Empowering the future of software engineering with advanced trustworthy artificial intelligence

[version 1; peer review: 2 approved with reservations, 1 not approved]

Ali Serdar Atalay[1], Ömer Öztin[1], Alper Kanak[2], İbrahim Arif[2], Atta Badii[3], Naga Sai Dasari[3], Salih Ergün[4], S. Halit Ergün[4], Oğuzhan Herkiloğlu[5], Grégory Mermoud[6], Paolo Papotti[7], Marco Tiemann[8], Pablo Gallegos[9], Marteyn van Gasteren[10], Dimitrios I. Fotiadis[11], Orkun Hasekioglu[12], Vasilis Pezoulas[11], David Wannier[6], Giuseppe Di Fatta[13], Andrea Rosani[13], Anastasija Collen[14], Niels Alexander Nijdam[14]

[1]AI4SEC OÖ, Talinn, Estonia
[2]Ergünler R&D Co. Ltd. - ERARGE, Isparta, Turkey
[3]University of Reading Department of Computer Science, Reading, England, UK
[4]ERGTECH Research GmbH, Zurich, Switzerland
[5]BITNET, Istanbul, Turkey
[6]HES-SO Valais Wallis School of Engineering Sciences, Sion, Valais, Switzerland
[7]EURECOM, Sophia Antipolis, Provence-Alpes-Côte d'Azur, France
[8]Innova Integra Limited, Reading, England, UK
[9]IDENER Research & Development A.I.E, Seville, Spain
[10]ITCL Technology Centre, Burgos, Spain
[11]Foundation for Research & Technology – Hellas, Heraklion, Greece
[12]ATPharma, Istanbul, Turkey
[13]Free University of Bozen-Bolzano, Bolzano, Italy
[14]University of Geneva, Genève, Switzerland

v1 **First published:** 19 Nov 2025, **5**:352
https://doi.org/10.12688/openreseurope.21192.1
Latest published: 19 Nov 2025, **5**:352
https://doi.org/10.12688/openreseurope.21192.1

Abstract

AI4SWEng, a Horizon Europe project that will be active from 2025 until 2028, unites 15 leading partners across the European Union, Switzerland and Turkey, combining experts in Model-Driven Software Engineering and trustworthy AI, with a special focus on applying Large Language Models. The project addresses complex challenges in industries such as healthcare, cyber-physical systems, and electric vehicles, focusing on multi-architectural and resource-constrained systems. Our mission is to transform agile software development by leveraging AI to boost efficiency, reliability, and security while ensuring ethical and regulatory compliance. The goal is to deliver scalable, sustainable, and socially responsible solutions that accelerate time-to-market without compromising quality. The

Open Peer Review

Approval Status ? ? ✗

	1	2	3
version 1 19 Nov 2025	? view	? view	✗ view

1. **Anh Nguyen-Duc**, Norwegian University of Science and Technology, Trondheim, Norway
2. **Filippo Lanubile**, University of Bari, Bari, Italy
3. **Jay Lofstead**, Sandia National Laboratories, Albuquerque, USA

AI4SWEng project will deliver an AI-powered software engineering suite providing end-to-end support for the software lifecycle, from code generation and advanced debugging to security, energy efficiency, and project management. By reducing pain points and enhancing productivity, the suite/platform aims to reduce developer stress, foster creativity, and improve job satisfaction. With a commitment to user-centred design and advanced prompt engineering, we empower developers to harness the full potential of AI. In this paper the AI4SWEng project consortium presents a novel and strategic approach proposed by the project consortium that will shape the future of software engineering, driving innovation and paving the way for more agile, intelligent, and sustainable software development.

Plain language summary
AI4SWEng is a European research project running from 2025 to 2028. It brings together 15 partners from the EU, Switzerland, and Turkey who are experts in software engineering and trustworthy artificial intelligence, especially Large Language Models (LLMs).

The project focuses on industries like healthcare, cyber-physical systems, and electric vehicles, where software often needs to work across different system architectures and with limited resources. Its main aim is to make agile software development faster, more reliable, and more secure—while following ethical and legal rules.

AI4SWEng will create an AI-powered toolkit that supports the entire software development process—from writing and debugging code to improving security, energy efficiency, and project management. The platform will help developers work more efficiently, reduce stress, and encourage creativity.

By combining user-focused design with advanced AI techniques, the project aims to shape the future of software development, making it more agile, intelligent, and sustainable.

Keywords
Software Engineering, Artificial Intelligence, Trustworthy AI, Agile Development, Large Language Models, Cyber Resilience

This article is included in the Horizon Europe gateway.

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Ali Serdar Atalay (ali.atalay@ai4sec.eu)

Author roles: Atalay AS: Conceptualization, Investigation, Methodology, Project Administration, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Öztin Ö**: Conceptualization, Investigation, Methodology, Project Administration, Software, Supervision, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Kanak A**: Conceptualization, Investigation, Methodology, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Arif İ**: Investigation, Software, Validation, Writing – Review & Editing; **Badii A**: Conceptualization, Investigation, Methodology, Project Administration, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Dasari NS**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Ergün S**: Conceptualization, Investigation, Methodology, Validation, Writing – Original Draft Preparation; **Ergün SH**: Conceptualization, Validation, Writing – Review & Editing; **Herkiloğlu O**: Conceptualization, Investigation, Methodology, Software, Writing – Original Draft Preparation; **Mermoud G**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Papotti P**: Conceptualization, Investigation, Methodology, Software, Writing – Original Draft Preparation; **Tiemann M**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Gallegos P**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **van Gasteren M**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Fotiadis DI**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Hasekioglu O**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Pezoulas V**: Conceptualization, Investigation, Methodology, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Wannier D**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Di Fatta G**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Rosani A**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Collen A**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation; **Nijdam NA**: Conceptualization, Investigation, Methodology, Writing – Original Draft Preparation

Competing interests: No competing interests were disclosed.

Grant information: This paper is supported by the AI4SWEng project that has received funding from the European Union Horizon Europe Programme under the HORIZON-CL4-2024-DIGITAL-EMERGING-01-22 call with grant ID: 101189908. Effort contributions of the Swiss partners (HESSO, ETCH, UNIGE) are financed by the State Secretariat for Education, Research and Innovation (SERI) of Switzerland. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or Swiss Government. Neither the European Union and Swiss Government nor the granting authorities can be held responsible for them.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2025 Atalay AS *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License , which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Atalay AS, Öztin Ö, Kanak A *et al.* **AI4SWEng manifest: Empowering the future of software engineering with advanced trustworthy artificial intelligence [version 1; peer review: 2 approved with reservations, 1 not approved]** Open Research Europe 2025, **5**:352 https://doi.org/10.12688/openreseurope.21192.1

First published: 19 Nov 2025, **5**:352 https://doi.org/10.12688/openreseurope.21192.1

List of abbreviations

Abbreviation	Description
AI	Artificial Intelligence
LLM	Large Language Model
CI/CD	Continuous Integration/Continuous Development
SO	Specific Objective
EO	Expected Outcome
KIO	Key Innovation Outcome
GUI	Graphical User Interface
NLP	Natural Language Processing
GAN	Generative Adversarial Network
IoT	Internet of Things
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
RFT	Random Forest Trees
LSTM	Long Short-Term Memory
CTC	Connectionist Temporal Classification
VGG	Visual Geometry Group
RAG	Retrieval-Augmented Generation
HLS	High Level Synthesis
FPGA	Field Programmable Gate Array
EHR	Electronic Health Record
GDPR	General Data Protection Regulation
API	Application Programming Interface
XSS	Cross-Site Scripting
MitM	Man-in-the-Middle
DoS	Denial-of-Service
AFL++	American Fuzzy Lop Plus
OSS-Fuzz	Open-Source Software Fuzzing Infrastructure
KLEE	Symbolic Virtual Machine for C programs
SDLC	Software Development Lifecycle
SaaS	Software as a Service
GNN	Graph Neural Network
V&V	Verification & Validation
TAT	Test Automation Toolkit
DNN	Deep Neural Network
VAE	Variational Autoencoder
AART	AI-Assisted Red-Teaming
QAIF	Quality AI framework
CPS	Cyber-physical systems
EVs	Electrical Vehicles
RUL	Remaining Useful Life
GIS	Geographical Information Systems
SDG	Sustainable Development Goals

Disclaimer

The views expressed in this article are those of the author(s). Publication in Open Research Europe does not imply endorsement of the European Commission.

Introduction

The integration of AI into software engineering is transforming the development pipeline, enhancing efficiency, productivity, and quality across various stages. AI techniques are being increasingly applied to various phases of the software development lifecycle, including requirements analysis, design, development, testing, and maintenance. These applications aim to improve processes and address challenges in software engineering[1]. AI-driven coding assistants have attracted significant investment, with nearly $1 billion in funding since early 2023[2]. These tools are becoming integral to software engineering, automating tasks such as code generation, debugging, and testing, thereby enhancing developer productivity and efficiency. Additionally, AI assists in project management and DevOps by automating routine tasks, improving time estimates, and optimising continuous integration/continuous deployment (CI/CD) pipelines. This integration enhances efficiency and reliability in the development process[3]. The role of AI in software engineering is expected to expand, with AI agents anticipated to handle a significant portion of software engineering tasks soon. This evolution underscores the importance of integrating AI into development workflows to maintain competitiveness and efficiency.

In particular, generative AI and LLMs are transforming software development by automating key processes such as requirement gathering, code generation, testing, and debugging, while working in collaboration with human developers[4]. AI tools streamline development by interpreting natural language inputs to produce code, generate user stories, and create test cases, enhancing productivity, reducing errors, and enabling developers to focus on complex, creative tasks. With capabilities such as autocompletion, code synthesis, and adaptive testing, AI accelerates the development cycle, improves software quality, and optimises debugging by identifying and fixing vulnerabilities and inefficiencies, thereby reshaping how software is designed, built, and maintained.

AI-supported software development offers transformative potential but also poses several challenges that need to be addressed[5,6]. Achieving efficient and agile development through AI (Specific Objective 1 (SO), see Section 2) is complex due to the integration of advanced AI models into traditional workflows, often requiring significant adaptation of existing tools and practices. Integration complexity (SO1) adds to this challenge, as diverse systems, frameworks, and architectures must seamlessly interact. Resource optimisation (SO3) becomes critical in leveraging AI efficiently while minimising computational and energy costs, especially in building dynamic and resilient multi-architecture software. Ethical and regulatory considerations (SO5), including compliance with frameworks such as the AI Act, demand careful attention to ensure fairness, privacy, and transparency. Advanced code analysis and fault prediction (SO1) highlight the need for robust AI models

capable of identifying vulnerabilities and inefficiencies, while scalability and resource constraints (SO3) pose difficulties in scaling AI solutions across large, resource-limited projects. Accurate evaluation and performance metrics (SO4) are essential for measuring the impact of AI tools on productivity and quality. Moreover, ensuring quality assurance (SO2) throughout the lifecycle remains challenging due to the dynamic nature of AI systems. Despite promises of efficiency and speed, many AI-enabled processes remain time-consuming (SO1), requiring refinement to match industry needs. Lastly, the integration of responsible software engineering practices (SO6) is vital to align AI-driven development with ethical, sustainable, and user-centric goals. Addressing these challenges holistically is essential to unlock the full potential of AI in software development.

The AI4sweng initiative[7], a research and innovation action funded under the Horizon Europe Programme, focuses on delivering AI-powered support to software developers. Its goal is to reduce stress in agile development processes and enhance productivity, job satisfaction, and creativity. The mission of the AI4SWEng project is to deliver transformative AI-powered support for high-efficiency system design and development with resilience, reliability, security-privacy protection, energy efficiency, maintainability, and re-usability, as well as ethical and regulatory compliance, such as AI-ACT[8]. The focus is on using AI models, techniques, and tools to support the system development and evaluation pipeline by:

- starting from data provenance checking, data extraction-generation-balancing, data cleaning, requirements engineering, data (re)-engineering, architecture specification, security and privacy protection and automated testing and
- at the deployment stage, using AI for continuous evaluation, fine-tuning, refinement and re-optimisation of the solution.

Thus, the project aims to advance software engineering methods, tools, techniques, and best practices by leveraging novel AI Engineering and Data technologies. The focus is on developing and maintaining software for multi-architecture systems, efficient and agile modelling, reliability, and security-by-design, including vulnerability assessment and mitigation[9].

Accordingly, AI4SWEng will deliver a software development platform capability and a TAT for AI-enabled software engineering to support all phases of the development lifecycle from requirements specification to (automated) testing of the most common building blocks of DevOps-MLOps solution stacks. This tremendous effort involves the design and development of an advanced framework for code analysis-generation, code-architecture, cross-compilation, validation, and semi-automatic creation of pre-conditions, post-conditions, and invariants for software modules. In this way, the project aims to support the automatic derivation of requirements and the development of conceptual and architectural models, architecture-to-code-to-architecture derivation, reverse execution, and dynamic code-slicing and debugging, code analysis for proactive code optimisation, fault prediction and location, as well as self-repair capabilities for reliability, resilience, and security in multi-architecture systems. Furthermore, recognising the worldwide shortage of AI-trained software engineering talent and the need for advanced training for the new generation of software developers with excellent skills in using such AI-powered tools to support agile development pipelines, AI4SWEng also provides user training particularly focused on crucial skills such as Prompt Engineering to support data analysis and quality improvement, code understanding, architecture specification, execution and dynamic-slicing.

This paper aims to announce the AI4SWEng mission as well as to inform the software development community about its promising strategy. This introduction is needed to reach a wider audience and raise awareness and discussions about the project scope. The presented solution can be helpful for other researchers to extend their vision towards the AI4SWEng perspective and foster joint studies and further exchange of knowledge and experiences. This overarching goal of the paper is presented in the following sections. Section 2 summarises the AI4SWEng objectives and the baseline background that motivates the project Consortium to propose the key innovations and solutions. Section 3 describes the high-level solution architecture and the underlying key innovations offered by the Consortium partners. Section 4 presents the use cases and the realistic strategy to prove the concept of the AI4SWEng idea. Section 5 summarises the expected impact in the medium and long term. Finally, Section 6 concludes the paper.

AI4SWEng background and objectives

AI4SWEng envisages advancing the state-of-the-art in AI-powered software engineering, emphasising a systematic and rigorous approach to designing, testing, validating, and maintaining software systems following established principles and best practice. A central goal of this endeavour is to address the challenge of AI bootstrapping for lifecycle support in AI-enabled software engineering. This involves leveraging LLMs to map natural language expressions of system requirements into formal specifications, while also facilitating data engineering, architectural design, code development, and testing. By harnessing AI, the project aims to enable task-specific "X-by-design" approaches, optimising the performance, reliability, and predictability of software systems through AI-driven development tools, such as the AI-SysDev and the TAT. Furthermore, the initiative aims to support the development of advanced systems characterised by parallelism, multiprocessing, and multi-architecture network-centric paradigms, aligned with the demands of modern system-of-systems engineering.

To enable the addressed advancement, AI4SWEng aims to revolutionise software engineering by leveraging responsible AI and data technologies, focusing on accelerating development and maintenance, particularly for multi-architecture systems. We address specific challenges in efficient and agile modelling, verification & validation (V&V), and vulnerability assessment & mitigation, all under the framework of

responsible AI principles. Dealing with the above-mentioned scope, AI4SWEng has six Specific Objectives (SOs) listed below:

1. Improving the development-time analysis and operationalisation of architecture design, including monitoring software performance, AI-driven log data analysis and bug fix generation, dynamic slicing and reverse execution, pro-active fault prediction and identification, and recommendations for corrective actions.

2. Enabling more effective generative AI-driven requirements engineering, architecture specification and code analysis-generation for common functionalities, automated testing and optimisation, including architecture generation from code and automatically generating function test suites for a given architecture/code base.

3. Realising a generative AI-driven code adaptation framework for compiler code generation, cross-compilation, and test-case creation; software scalability-reusability through dynamic reconfiguration with CodeOps-DevOps support.

4. Strengthening auditable use of generative AI in software development, including source identification and data-provenance-aware training of generative AI systems. Analysis of the impact of AI-enabled software deployment for full functional compliance responsive to key performance metrics, e.g. resilience, reliability, security-privacy, energy efficiency and maintainability.

5. Adopting auditable security-privacy protection by design, socio-ethical acceptability (AI-ACT Compliance) and user acceptance.

6. Establishing a framework for software developer training to best exploit the resulting AI-powered tools to accelerate agile, efficient software engineering, using AI-generated insights to optimise software development decisions, resource allocation, and road-mapping the extension of the certified common code base. -including open-source availability of the resulting AI4SWEng tools and project -policy insights.

To address the recent challenges of software development, including hardware-based design and embedded programming, there is a critical need for research and development initiatives that advance the State-of-the-Art in responsible software engineering methods, tools, and best practices. AI4SWEng focuses on smart intelligent system specification, agile system and code development, advanced code analysis, fault prediction and location, and self-repair mechanisms enabled by emerging AI and data technologies. In line with the SOs mentioned above, the following major outcomes are expected (Expected Outcome – EO):

1. A suite of AI-powered tools and techniques for improved lifecycle software development and maintenance efficiency.

2. Higher developer job satisfaction and productivity through AI-powered support, particularly for accelerated resolution of the more stressful and time-consuming "pain-point" stages of the software development pipeline to achieve demonstrably reduced time-to-market for new software releases and faster turnaround times for resolving software issues.

3. Enhanced software development process quality with minimised bugs and proactive maintenance approaches.

4. Data-driven insights for informed design decision-making and strategic software development planning.

Aligned with these SOs and EOs, the AI4SWEng project envisages addressing the challenges that arise from the implementation of an AI-powered software development environment through specific paradigmatic responses designed to enhance efficiency, security, scalability, and ethical compliance. Energy efficiency is achieved using smaller, task-specific LLMs, particularly beneficial for deployment in resource-constrained environments such as Smart Edge nodes. The approach ensures that the system is inherently secure, with all processing and fine-tuning performed locally, eliminating the need for sensitive data to leave the premises of the client organisation. The system also features an auditable pipeline by leveraging techniques such as Retrieval-Augmented Generation (RAG), enabling the traceability of outputs to their original datasets. Scalability is ensured with task-specific LLMs with RAG, capable of handling extensive data sets and contexts exceeding the inherent capacity of the model. Moreover, the system is flexible, adapting and evolving based on organisation-specific proprietary data, thereby offering tailored solutions. Importantly, the project prioritises auditable ethical quality, ensuring that AI solutions align with societal values, stakeholder rights, and regulatory norms. This comprehensive strategy is supported by an intense solution framework and targeted key performance indicators, ensuring measurable impact across energy efficiency, security, scalability, and ethical compliance dimensions.

Policy background

The AI4SWEng project is set to make substantial contributions to the EU policy framework, with tangible impacts across key European strategic dimensions. By delivering technical innovations such as the AI-SysDev Tool, the Test Automation Toolkit (TAT), LLM-powered engineering components, and energy-efficient AI deployment, the project bridges advanced research with EU policy ambitions.

- Strengthening the EU Industrial Strategy[10]: The AI4SWEng project approach to flexible, secure, decentralised, and AI-enabled software development aligns with the goals of the EU Industrial Strategy. Tools such as the AI-SysDev Tool and TAT empower SMEs to adopt automated, efficient, and standards-compliant development workflows, reducing costs and time-to-market. By enabling modular, scalable, and reusable code components, the project supports digital innovation ecosystems and contributes to a resilient and competitive European industry.

- Advancing the European Green Deal[11]: Through the integration of TinyML, task-specific LLMs, and energy-aware automation tools, AI4SWEng supports the Green Deal objective to create a resource-efficient and sustainable economy. The platform enables low-energy AI inference at the edge, promotes efficient testing practices, and facilitates greener software development pipelines—contributing to lower emissions and reduced energy consumption in digital operations.

- Supporting the Digital Single Market[12]: AI4SWEng promotes secure, standardised, and interoperable digital solutions. The use of Retrieval-Augmented Generation (RAG), data provenance tracking, and privacy-preserving LLM fine-tuning ensures compliance with data sovereignty and transparency requirements under the Digital Single Market framework. These features support the free flow of data, cross-border collaboration, and trustworthy AI adoption throughout the EU.

These contributions reflect how the AI4SWEng project technical outputs serve as enablers for major EU policy objectives, highlighting the potential of the project to drive sustainable, secure, and industrial-grade digital transformation. By embedding regulatory alignment, ethical safeguards, and energy efficiency at the core of its tools, AI4SWEng acts as a strategic force for European digital and green innovation.

Scientific background

AI-enabled code analysis and understanding support code generation, testing, debugging, documentation, and maintenance through natural language processing, machine learning, and deep learning to analyse software structure, semantics, and behaviour[13]. This includes addressing agile project management challenges through a theoretical pain point model, with generative AI proposed as a remedy for practical issues. Current advancements focus on code representation, synthesis, completion, summarisation, search, recommendation, testing, debugging, refactoring, and security. Notable tools include CodeBERT[14], which encodes natural and programming languages into a unified semantic space; DeepCode[15], which synthesises an executable code from test cases across multiple domains; and Code2Vec[16], which encodes abstract syntax trees into fixed-length vectors for tasks such as summarisation, classification, and similarity detection. Automated software testing encompasses test execution, adequacy, prioritisation, suite reduction, fault localisation, program repair, and verification, offering a comprehensive framework for improving software reliability[17].

AI-assisted code generation has gained significant traction with tools such as CoPilot[18], which automates application development using AI models and natural language understanding. AlphaCode[19] employs reinforcement learning to tackle complex programming tasks, while TransCoder[20] by Facebook AI enables code translation across programming languages, facilitating legacy code migration and multilingual development. CodeNet[21], an open-source dataset with 14 million samples in 50 languages, supports diverse programming tasks. Similarly, OpenAI Codex[22] excels in code generation, translation, refactoring, and bug fixing. Despite these advancements, challenges persist, including limitations in understanding language semantics and achieving sophisticated code comprehension. Providers of integrated software development environments, such as JetBrains, are offering AI assistant functionalities that combine or switch between different commercial models for different applications[23].

AI-enabled simulating reverse execution logic enables programs or systems to run backwards, reversing forward computations to aid in error correction, state restoration, and scenario exploration. Applications include reversible execution for robustness in embodied AI and industrial robots[24], which integrates reversible computation with AI planning for robotic assembly tasks. The RAMSES framework[25] uses reversibility and speculation to enable parallel execution in agent-based discrete event models. Additionally, ReVA[26] serves as an AI assistant for reverse engineering, functioning as a disassembler-agnostic tool for reverse engineering tasks.

AI-enabled generation of pseudocode for reverse execution leverages AI to generate human-readable pseudocode from executable code, facilitating reverse engineering, debugging, and testing. By simulating reverse execution, this approach efficiently reconstructs prior execution states to identify and correct errors or explore alternative scenarios. Authors of a recent study introduced a novel methodology utilising large language models for static analysis through pseudocode simulation[27]. Additionally, a retrieval-based transformer model was proposed that combines code retrieval and generation techniques to produce pseudocode from executable code, further advancing the capabilities of AI in software analysis[28].

AI-enabled reverse execution can be achieved by various methods, such as recording and replaying the program state, performing backward taint analysis, or using symbolic execution. Authors of a recent paper presented two systems that used reverse execution to triage and debug software failures in production: i) RETracer[29] to triage software failures based on program semantics reconstructed from memory dumps; ii) REPT[30] reconstructed the execution history with high fidelity by combining online lightweight hardware tracing of the control flow with offline binary analysis of the data flow.

The challenges and opportunities of using AI techniques to automate the reverse engineering of protected software were presented in 31. This study proposed some open research directions, highlighting the deployment of machine learning and symbolic execution.

AI-enabled automated design patterns and code suggestions enhance the efficiency, reliability, and maintainability of software development by providing automated guidance and feedback, supporting the creation of high-quality, consistent, and reusable code. A recent study demonstrated how design patterns can structure automation solutions in a modular and

reusable manner, with examples highlighting the capabilities of UiPath, a robotic process automation platform, in implementing these patterns[32]. Other work has introduced an experimental approach for the automatic detection of design patterns in source code through machine learning, achieving high accuracy and scalability compared to traditional manual methods[33]. These advancements underline the potential of AI-driven solutions in streamlining software development processes and promoting best practices.

AI-enabled dynamic program slicing focuses on techniques such as targeting backward execution, recording and replaying program states, backward taint analysis, and symbolic execution to analyse and debug software. In a recent study, a novel hybrid dynamic-static slicing approach was introduced that balances accuracy and runtime cost. Evaluated on 21 failures across six widely deployed applications, the technique successfully recovered 94% of program statements in the dynamic slice while incurring only a 5% runtime overhead, demonstrating its effectiveness for debugging complex systems[34]. Similarly, AI-enabled automated trace generation leverages AI to generate and analyse execution traces—sequences of events or states representing software behaviour—for debugging, testing, optimisation, and reverse engineering[35]. Additionally, a recent review paper explored the role of AI in enhancing test automation through the automated generation of test scripts, test data, execution, maintenance, and root cause analysis, showcasing its potential to streamline software testing processes[36]. AI-enabled automated testing is to support the quality, efficiency, and reliability of software testing. AI-enabled automated testing tools can generate and prioritise test cases, analyse test results, identify defects, and provide feedback and recommendations. For instance, TestLab is an intelligent automated software testing framework, presented as a framework for the integration of various AI-powered (continuous) testing methods (vulnerability detection, security analysis, and test-case generation)[37]. A very recent study presented user reviews and ratings of various AI-augmented software testing tools in machine learning, self-healing heuristics, or computer vision, to enhance the capabilities of automated testing[38].

White-box Testing of Deep Neural Networks (DNNs) examines the internal structure, mechanisms, and implementation details of models to evaluate correctness, robustness, and performance. Unlike black-box testing, white-box approaches require access to the model architecture, parameters, and training data. Key techniques include architecture analysis, parameter inspection, gradient and activation checking, coverage analysis, and adversarial or fuzz testing. Fuzz testing, exemplified by tools such as DeepHunter[39], assesses model resilience against unexpected inputs, aiding in robustness improvement, failure mode identification, and security testing. Symbolic execution complements fuzzing by generating input candidates abstractly, covering multiple execution paths, and avoiding false warnings.

The AI4SWEng innovation mission aims to advance the capabilities of AI-powered tools for supporting software engineering pipelines by addressing both the performance of AI models and developers' ability to effectively model and target problem subspaces. To go beyond the state-of-the-art, the project will enhance foundational support for tools such as the AI-SysDev Tool and TAT by leveraging open-source LLMs, trained locally on proprietary data to learn from past requirements and optimise tasks such as requirements engineering, architectural planning, code generation, and automated testing. For large-scale data/code exceeding LLM context limits, RAG strategies will enable scalability, transparency, and auditable lineage tracing. The innovation emphasises scalable LLM modelling, data acquisition and preparation, and seamless transitions from requirements engineering to architecture planning, code development, and testing, thereby optimising the end-to-end exploratory data analysis and modelling pipeline for software development.

AI4SWEng innovation landscape

The integrated solution stack is formed of Key Innovations and Outcomes (KIOs), each of which is planned to be developed as software-only or hardware-supported services. All underlying KIOs are integrated over the AI4SWEng AI Engineering Suite (Suite, hereinafter), i.e., KIO1, which is an online framework integrating other KIOs that are to be offered as complementary KIOs for joint deployment. KIO1 stands out as a distinctive tool that integrates KIOs as compliant with a typical DevOps-based software development lifecycle. As seen in Figure 1, KIO1 has a Graphical User Interface (GUI) enabling code generation, Rew/Fwd and execution functions at the front end. At the backend, each KIO is integrated as a service, governed by containerisation, so that the engineering suite has a flexible and modular architecture. KIO1 relies on a typical software development life cycle, starting with requirement engineering and planning (KIO3, KIO4) by enabling the NLP-to-Formal Requirements-to-Code planning. The second step is data management, which incorporates data generation, preparation, validation and quality checking (KIO5, KIO6). Coding is the third phase where design-by-contract code generation takes place (KIO9). Refinement/improvement is a crucial step that supports the coding by enabling bug Locate & Fix, reverse execution and dynamic slicing (KIO2) as well as energy-efficient model-driven code enhancement/generation (KIO10). The fourth phase is parallel with the refinements and improvements as it incorporates AI tools for cross-compilation and reconfiguration using the DevOps methodology (KIO8) and AI-powered test automation (KIO11) within the agile development cycle. The AI-SysDev Tool (KIO7) encapsulates all these four phases of the life cycle as LLM is used horizontally in other KIOs. Multi-level cybersecurity (KIO12) and developer training tools (KIO13) are other complementary tools intended to improve the resilience of the Suite against privacy and security issues and enhance the extrapolation and adoption of the KIO1 by stakeholders. The General Data Protection Rule (GDPR) and ethics, model reliability and integrity checking, awareness activities, and strategic alignment with mainstream software development products and standards will help the Consortium fulfil the usability, acceptability, interoperability, scalability, fairness and trustworthiness requirements of the KIO1. Brief descriptions of the KIO2-KIO13 are as follows.

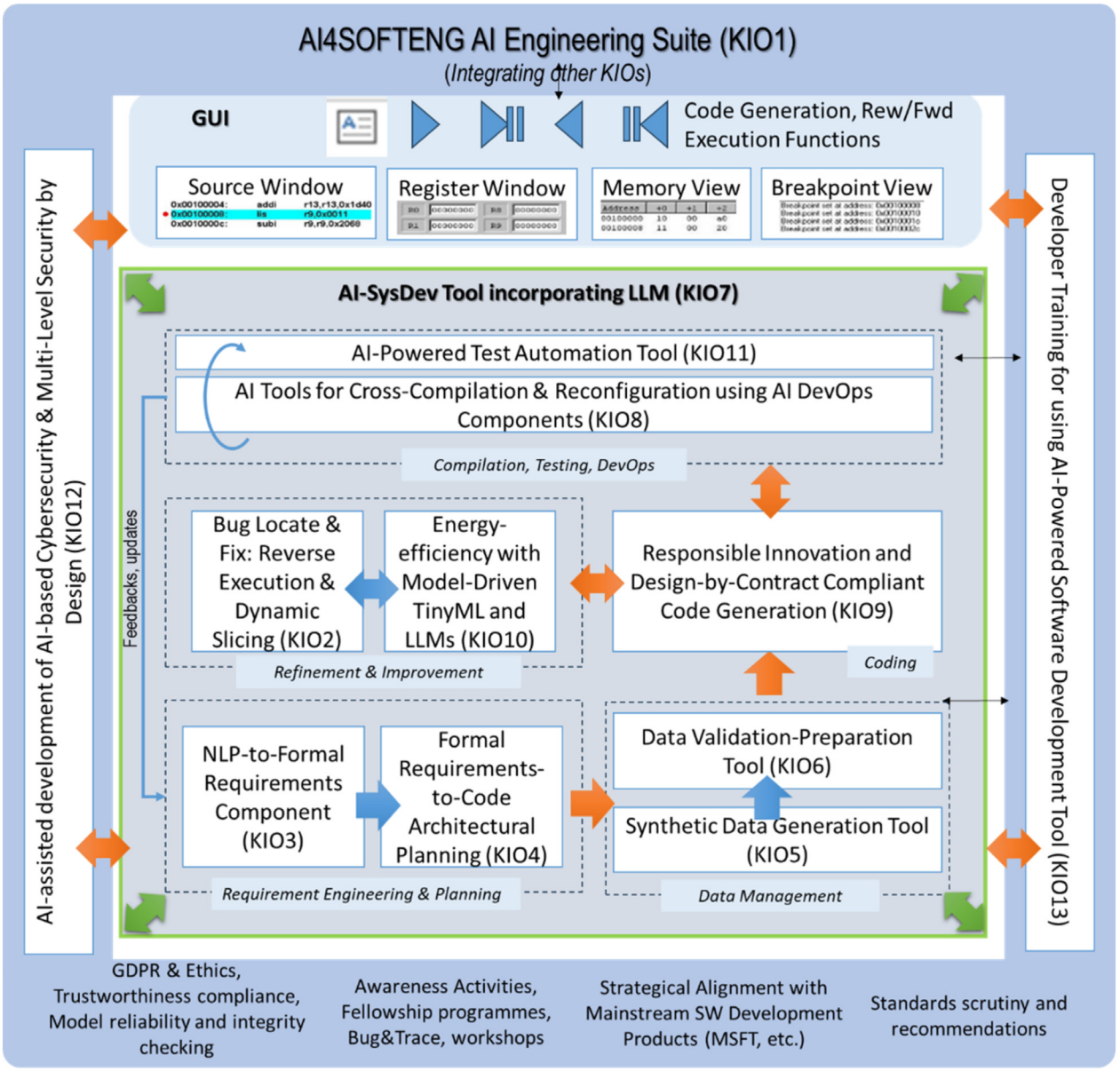


Figure 1. AI4SWEng AI Engineering Suite (KIO1) and underlying KIOs.

The effective integration of the KIOs into a cohesive and extensible framework is central to the success of the AI4SWEng Engineering Suite (KIO1). Each KIO represents a standalone innovation that addresses a specific challenge in the software development lifecycle—ranging from NLP-based requirement formalisation and code generation to automated testing, energy-efficient deployment, and ethical compliance. To support user interaction, KIO1 includes a graphical interface that enables code generation, reversible execution, and visualisation of development flows. To ensure that these components work in harmony, the proposed Suite implements a modular integration strategy based on standardised service interfaces, shared data formats, and orchestrated execution environments. This integration strategy adopts containerisation and microservice architecture to enable plug-and-play interoperability across KIOs. The platform enables flexible configuration and deployment, supporting diverse use cases and evolving software workflows by exposing each component as a service. A central integration layer manages the flow of data and control signals between KIOs, enabling iterative refinement loops, for example, linking the outcomes of automated requirement extraction and planning (KIO3, KIO4) with code synthesis (KIO9), and feeding runtime bug reports (KIO2) back into testing automation (KIO11) and reconfiguration modules (KIO8). Moreover, a unifying layer ensures semantic consistency and traceability throughout the development process. This includes

the integration of Retrieval-Augmented Generation (RAG) pipelines for cross-KIO knowledge management, ensuring that generative AI outputs are both contextually grounded and auditable. KIO7 serves as an orchestration hub that coordinates LLM-based intelligence across the entire suite. Additional capabilities include energy-aware model-driven code enhancement (KIO10), AI-based cross-compilation and reconfiguration, and automated test generation. Complementary tools for multi-level cybersecurity (KIO12) and developer training (KIO13) further strengthen the resilience and adoption of the Suite. Finally, alignment with GDPR, ethical AI principles, and mainstream software development standards helps to ensure the usability, trustworthiness, and scalability of KIO1 across diverse stakeholders.

KIO2: bug locate & fix: reverse execution & dynamic slicing component

In complex multi-threaded and multi-processor environments, dynamic slicing with forward and reverse execution is essential for identifying and resolving intricate bugs. AI aids this process by analysing programs, isolating relevant segments, and simulating reverse execution logic to generate pseudo-code, offering insights into root causes. Developers can then rewind or replay execution scenarios to apply self-correcting algorithms. Microsoft Research BugLab[40] leverages a deep learning model trained without labelled data using a self-supervised "hide and seek" approach. This involves two models—a bug selector and a bug detector—collaborating to identify and fix issues, such as incorrect comparisons or variable misuse. This approach highlights the growing trend of using AI for automated bug detection and repair, significantly reducing manual debugging efforts and accelerating development. The BugLab methodology, reminiscent of generative adversarial networks (GANs), employs a dynamic interplay between generating and detecting bugs.

KIO3: Natural Language Processing (NLP) to formal requirements component

KIO3 establishes a structured process for translating natural language descriptions into formal software engineering requirements. It begins by converting descriptions into user stories formatted as "As a [user type], I want [goal] so that [reason]," ensuring clarity and relevance to user needs. Functional and non-functional requirements are then defined, outlining the software features, its performance criteria, security measures, and usability standards. Wireframes are created to visually represent the software layout and functionality, offering a concrete understanding of its intended design. Acceptance criteria are established to determine when the software is complete and ready for delivery. Additionally, assumptions and constraints are documented to account for any factors influencing the development process. This systematic approach bridges the gap between natural language descriptions and formal requirements, advancing the development of Data Preparation Tools. These tools transform raw data into training sets for LLMs, which are fine-tuned using examples from the prepared data. A RAG strategy is implemented to address cases where input data exceeds the LLM context size. The integrated system is validated to ensure its functionality and effective performance within the targeted domain.

KIO4: Formal Requirements-to-Code architectural planning component

Building on the framework for translating natural language descriptions into formal software engineering requirements, KIO4 extends the process to code architectural planning, ensuring a smooth transition from requirements to implementation. By leveraging the fine-tuned LLMs from the previous stage, the approach supports mapping formal requirements to architectural components. The process includes requirements analysis to identify architectural elements and establish their mapping to guide planning. It formalises component properties, interfaces, and relationships, enhancing transparency in design decisions and supporting future maintenance. Architecture modelling and visualisation further aid this process, enabling integration and alignment with the broader software development pipeline. A real-time architectural suggestion mechanism, powered by fine-tuned LLMs, provides immediate feedback and recommendations, streamlining the architectural planning phase.

KIO5: synthetic data generation tool

As part of the broader trend of leveraging AI to enhance data availability and quality, advancements in AI-powered data synthesis and generation have become essential for developing robust AI systems. These techniques address challenges such as scarce or sensitive real-world data and imbalanced datasets, particularly in domains involving natural systems where minority classes are common. Reliable model performance in these scenarios requires balanced datasets while preserving the integrity of feature distributions across classes. KIO5 will build on recent innovations in data generation, including

1. NLP techniques using transformer-based LLMs trained on extensive code datasets to support data generation and programming tasks[41];
2. Generative Models, such as GANs and Variational Autoencoders (VAEs), to synthesise data without requiring real-world collection[42];
3. AI-Assisted Red-Teaming (AART), which automates data generation and augmentation to integrate adversarial testing earlier in development[43]; and
4. Tools such as the MIT Synthetic Data Vault[44] provide open-source solutions for generating diverse data types.

These approaches aim to improve data quality and availability while maintaining pattern integrity across datasets.

KIO6: Data validation-preparation tool

The Data Validator ensures data quality by comparing properties against a generated schema and recommending updates with each data ingestion instance. It supports single-batch and inter-batch validation and verifies assumptions in the training code not reflected in the data. The Data Analyser computes

statistical properties, while the Model Unit Tester identifies errors in training code using schema-based synthetic data. To operationalise the Data Understanding and Preparation phases of the Quality AI Framework (QAIF), the project will deploy the Data Quality Wrapper tool for automated preprocessing and report generation[45]. This approach emphasises quantifying observations, aligning with the data-centric AI paradigm that prioritises data quality and validation. Additionally, a Data Preparation tool will be implemented to extract, clean, and format existing textual and structured data, transforming it into training sets suitable for fine-tuning LLMs.

KIO7: AI-SysDev Tool incorporating LLM fine tuning and inferencing

LLM fine-tuning focuses on local training and customising open-source language models for tasks such as requirements engineering, architectural planning, code generation, and automated testing. Using examples from the Data Preparation Tool (KIO6), fine-tuned models adapt to project-specific needs. During inference, when input data exceeds the LLM context size, a RAG strategy is applied, including retrieval, re-ranking, and prompt compression[46]. This approach ensures better bug avoidance, standards compliance, and energy efficiency, aligning the fine-tuned LLMs with the needs of the software development process. The methodology extends to software maintenance and testing. In maintenance, LLMs assist with code refactoring, bug detection, and documentation by analysing the codebase and automating tasks. Fine-tuning models on domain-specific data enhances their relevance and integration into workflows. For testing, LLMs generate and analyse test cases, identify vulnerabilities, and simulate user interactions. Techniques such as Connectionist Temporal Classification (CTC) and encoder-decoder architectures are utilised for handling complex input-output sequences, enabling more effective and automated regression testing, and ultimately improving software quality and reliability. To further enhance user interaction insights, pre-trained CNNs (e.g., VGG, ResNet) and word embeddings (e.g., GloVe[47], FastText[48]) are employed to encode web page snapshots and user actions into dense vectors. These embeddings are fine-tuned during training to align with task-specific requirements. Transformers, such as TransCoder[49], support unsupervised machine translation for code adaptation between languages and webpage sampling improvements. By leveraging high-quality webpage samples, the system can recommend optimal module implementations and provide detailed feedback, overcoming limitations in traditional data logging and enabling more nuanced insights into user behaviour.

KIO8: AI Tools for cross-compilation & reconfiguration using AI DevOps components

Years of research in automata processing architectures and software engines have expanded the application domains of finite automata. However, fragmented software frameworks and restrictive licensing hinder cross-platform evaluation. To address this, MNCaRT[50], a centralised suite for automata processing, supports state machine compilation, execution, and optimisation across platforms. Advancements in containerisation, exemplified by Docker, ensure OS-independent deployment and portability despite challenges in implementation. KIO8 focuses on developing generative AI tools for cross-compilation and reconfiguration, leveraging AI/ML to automate code, configuration, and deployment script generation for various architectures. These tools analyse existing codebases, identify optimisation opportunities, and generate platform-compatible, optimised versions. By accelerating development cycles and improving scalability and resource utilisation, this approach enhances software adaptability across frameworks, platforms, and hardware configurations. Integrating AI DevOps components into the development pipeline facilitates seamless integration, continuous deployment, and real-time optimisation. This innovation transforms traditional software development by fostering portability, scalability, and agility, enabling organisations to drive efficiency and innovation in their software engineering practices.

KIO9: responsible innovation and design-by-contract compliant coding

This KIO aims to enhance the LLM's ability to integrate ethical guidelines into data handling, ensuring adherence to principles of fairness, accountability, transparency, and compliance with privacy laws. By embedding ethical norms, LLMs can generate and process data responsibly, addressing biases and upholding socially accepted standards. The approach includes an ontologically guided framework for context-specific compliance, enabling LLMs to suggest data processing protocols and safeguarding measures. It also focuses on facilitating abstraction transformations for seamless tool integration, encoding non-functional requirements such as security and reliability in AI-understandable formats, and enabling LLMs to adapt to evolving user business requirements, ensuring responsiveness to dynamic compliance needs throughout the software lifecycle.

KIO10: energy-efficiency with model-driven TinyML and LLMs

Since 2012, AI system energy consumption has surged, accounting for up to 9% of global electricity use and 2% of CO2 emissions[51]. To address this, the AI community is prioritising energy-efficient development, exemplified by the AI4SWEng project Model-Driven TinyML for IoT Edge devices[52]. This approach leverages domain-specific modelling to tackle design and validation challenges, improving service quality, ML performance, privacy, security, and sustainability. By enabling resource-constrained edge devices to perform ML tasks locally, TinyML minimises energy consumption by reducing data transmission to the cloud. Task-specific LLMs perform on-device analytics, further cutting energy use.

AI4SWEng employs multi-level abstraction in software modelling, facilitating platform-independent and platform-specific development and validation. The project advances smart IoT services deployable on ultra-low-power microcontrollers. Using TensorFlow Lite and its microcontroller variant, it integrates model-driven software engineering with AI to generate efficient code for diverse IoT platforms.

KIO11: AI-Powered TAT

This initiative focuses on developing a TAT tailored to the agile software development pipeline by integrating advanced tools and methodologies. The American Fuzzy Lop Plus (AFL++) fuzzer[53] will serve as a foundational component, providing robust fuzzing capabilities. The open-source OSS-Fuzz tool[54] will be incorporated, either partially or fully, to enhance the TAT testing functionality. Additionally, LLMs will be deployed to automate key aspects of the tool, while the dynamic symbolic execution engine KLEE[55] will complement the fuzzers by supporting symbolic execution tasks. To further strengthen the capabilities of the tool, integration with Angr will enable both static and dynamic ("concolic") binary symbolic execution. The tool will also provide user guidance on effectively utilising Angr[56], particularly in cases involving binary-format inputs. This cohesive approach ensures a state-of-the-art test automation solution, optimised for the demands of agile software development.

KIO12: AI-based cybersecurity & multi-level security by design

An AI-based cybersecurity solution is proposed as a service that promises Host and Network Intrusion Detection, particularly for identifying zero-day attacks that evade signature-based systems. However, challenges include a lack of representative datasets for training and evaluation and the complexity of configuring detection systems to address specific attack types, which influences feature selection and engineering. The AI4SWEng project addresses these issues by leveraging NLP to automate key aspects of cybersecurity: generating representative benign and malicious datasets from high-level communication scenario descriptions, using these datasets for training and validating AI-based detection systems, and assisting in the development of attack detection engines by automating technical decisions such as AI primitives and feature engineering. The approach extends beyond detection to response optimisation. The project also explores API misuse prevention by employing LLMs to derive and enforce technical requirements from contractual conditions, ensuring compliance with terms of use. Furthermore, LLMs are applied to detect real-world attack scenarios such as SQL Injection, Cross-Site Scripting (XSS), Man-in-the-Middle (MitM), Denial-of-Service (DoS), and phishing. Fine-tuned LLMs play a critical role in identifying vulnerabilities, generating secure code, conducting security checks, and recommending prioritised countermeasures, enabling robust security practices throughout the software development lifecycle.

KIO13: Developer training for using an AI-Powered software development tool

Advanced prompt engineering is essential to effectively leverage LLMs throughout the software development stages, including NLP-based requirements, architecture design, code generation, debugging, and optimisation. Addressing the global shortage of skilled software-AI engineers, this KIO focuses on delivering structured training programs with multimedia resources. These programs emphasise ontologically framed requirements, resource and constraint considerations, and compliance objectives to enhance prompt engineering and qualitative result evaluation. This initiative aims to transform the software engineering ecosystem by improving developer productivity, creativity, and job satisfaction, leading to faster time-to-market and increased market share for enterprises. KIO13 will offer advanced training in ontologically-rooted prompt engineering via a web service, incorporating tools for ontological engineering and visualisation to support requirements framing and prioritisation aligned with system design by contract.

Team and targeted use cases

AI4SWEng is a 42-month project, which commenced on December 1st, 2024, and funded as a Research and Innovation Action under the EU HORIZON-CL4-2024-DIGITAL-EMERGING-01-22 call with grant ID: 101189908[7]. AI4SWEng is implemented by a well-balanced and multi-disciplinary consortium, as listed in Table 1, comprised of 15 complementary partners, representing a total of eight countries worldwide, that is composed of 5 EU, and 3 Associated (UK, Türkiye and Switzerland) countries.

The Consortium is a high-quality partnership fully capable of attaining the objectives of the project. Its complementarity is based on the diversified expertise, knowledge, and networks that our partners possess, substantially covering all skills, resources, and access to stakeholders required for successful implementation. In our current partner composition, we have partners for use cases in the intelligent software development sector. Our partners can be grouped into three main types:

i. R&D partners (RTOs) experienced in AI/ML, LLMs, and semantic web frameworks (AI4, EUR, CEA, UNIBZ, UREAD, IDE, ITCL, FORTH, HESSO, ETCH).

ii. Industrial partners and technology enablers that will develop the proposed system and deploy it to the facilities of use case organisations (EUR, ITCL, IDE, INNO, ERG, BIT, ATT, ETCH)

iii. Industrial partners which will provide systems, facilities or operational environments for use.

Targeted use cases

The AI4SWEng project aims to validate AI-powered tools for delivering efficient, reliable, and socio-ethically compliant (AI-ACT) software engineering solutions across diverse modern cyber-physical applications. These include smart compliance layers that support communication, standards, and regulatory compliance at both functional and non-functional levels. The project also focuses on adaptive Cyber-Physical Systems (CPS) capable of interacting with physical environments to provide intelligent and responsive services. In addition, it addresses the smart edge, a network layer comprising IoT devices that perform local data processing, analytics, and decision making to minimise latency, bandwidth demands, and privacy concerns associated with cloud computing. Furthermore, the project targets smart CPS, distributed systems that integrate

Table 1. List of Participants.

Nr.	Legal Name	Short Name	Country
1	AI4SEC OU	AI4	Estonia
2	EURECOM GIE	EUR	France
3	COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES	CEA	France
4	LIBERA UNIVERSITA DI BOLZANO	UNIBZ	Italy
5	FUNDACION INSTITUTO TECNOLOGICO DE CASTILLA Y LEON	ITCL	Spain
6	IDRYMA TECHNOLOGIAS KAI EREVNAS	FORTH	Greece
7	IDENER RESEARCH & DEVELOPMENT AGRUPACION DE INTERES ECONOMICO	IDE	Spain
8	THE UNIVERSITY OF READING	UREAD	United Kingdom
9	BITNET BILISIM HIZMETLERI LIMITED SIRKETI	BIT	Türkiye
10	INNOVA INTEGRA LIMITED	INN	United Kingdom
11	ATPHARMA ILAC SANAYI VE TICARET LIMITED SIRKETI	ATT	Türkiye
12	ERGUNLER INS. PET. UR. OTOMOTIV TEKS. MAD. SU UR. SAN. TIC. LTD. STI.	ERG	Türkiye
13	HAUTE ECOLE SPECIALISEE DE SUISSE OCCIDENTALE	HESSO	Switzerland
14	ERGTECH Research GmbH	ETCH	Switzerland
15	UNIVERSITE DE GENEVE	UNIGE	Switzerland

data intelligence with physical processes through computation, communication, and control, enabling seamless feedback loops between cyber and physical components. This validation underscores the transformative potential of AI tools in enhancing software quality, trustworthiness, and efficiency in a wide range of applications.

AI4SWEng will be validated in five use cases, which are detailed below:

Use case 1 (horizontal)
Optimising Software Development Lifecycle Using LLM: This use case demonstrates how LLMs can optimise the Software Development Lifecycle (SDLC) by automating key processes and improving efficiency, accuracy and sustainability. Software professionals, including developers, project managers, and testers, can leverage LLMs to reduce time and effort on tasks such as generating software requirements, drafting problems, and writing code. The process begins with the Project Manager (PM) providing task details to an LLM-based platform, including priority, programming language, and tags. The platform generates draft issues based on predefined templates, complete with tags such as priority, size, and story points, and may include automated tests and initial code solutions. PMs then review and refine the generated content to ensure completeness and accuracy before approving it for implementation. LLM tools enable developers to implement tasks more efficiently while adhering to energy-efficient coding practices. The completed code undergoes an automated compliance check to ensure it meets company standards and reduces the likelihood of bugs and human errors. The approach also includes mechanisms to address alternative flows, such as prompting developers to correct input inconsistencies or resolve compliance issues before finalising the code. The result is an SDLC workflow that is faster, more accurate, and consistent, with reduced manual effort and improved standardisation between projects. The benefits of this LLM-powered system include significant time savings, enhanced accuracy, and adherence to agile and coding standards. It promotes collaboration through standardised documentation and contributes to sustainability by encouraging energy-efficient practices. By integrating automated issue and code generation into the SDLC, software organisations can transform their workflows, achieve greater productivity and improved outcomes while aligning with modern standards of efficiency and sustainability.

Use case 2 (vertical)
Automating Medical Report Generation Using LLMs: This use case demonstrates how LLMs, such as GPT-4, can revolutionise medical report generation in the healthcare industry, improving efficiency, accuracy, and patient care. Healthcare professionals, including doctors, nurses, and medical assistants, can leverage LLMs to automate the creation of accurate and detailed medical reports, reducing the time and effort required for manual documentation. The process begins with inputting patient-specific data from electronic health records (EHR), including demographics, medical history, symptoms, diagnostic results, and treatment plans. The LLM analyses the data and generates a draft medical report based on predefined

templates, covering essential sections such as patient information, diagnosis, treatment plans, and follow-up recommendations. Healthcare professionals review and edit the draft to ensure completeness and accuracy before finalising and approving the report for storage in the EHR system. Automated compliance checks ensure the reports meet regulatory standards such as HIPAA and billing requirements. Once approved, the reports are securely stored and made accessible to authorised healthcare providers, with options for patient distribution. In cases of input data inconsistencies or compliance issues, the system prompts professionals for corrective action before proceeding. This approach significantly enhances efficiency by reducing manual workload, enabling healthcare professionals to focus more on patient care. It ensures the accuracy, consistency, and standardisation of medical reports while maintaining compliance with regulatory guidelines. By integrating LLM-based automation, healthcare organisations can streamline documentation practices, improving productivity and the overall quality of care.

Use case 3 (horizontal)
LLMs for Advanced Software Generation for FPGAs in C++ and High-Level Synthesis (HLS): This use case highlights the application of LLMs to streamline and enhance the FPGA software development lifecycle, addressing complex tasks such as deep learning implementations, cryptographic algorithm design, and video processing. The approach leverages fine-tuned LLMs trained specifically in C++ and HLS to generate and refine FPGA software iteratively. Through advanced prompting engineering techniques such as "Graph of Thoughts," the LLM engages in iterative reasoning, self-evaluation, and dynamic adjustments to produce code aligned with user-defined specifications.

Users provide textual input specifying program requirements, features, and functionalities. The LLM uses this input to iteratively generate HLS source code files, which are built into an FPGA project via automated scripts. Feedback from FPGA tooling, including timing and hardware usage reports, is integrated into the iterative process to refine the software further. This feedback-driven approach ensures the final output meets user requirements precisely, improving efficiency and customisation in FPGA software development. The use case demonstrates significant benefits, including enhanced efficiency, precision, and customisation in FPGA programming, enabling users to obtain tailored solutions for complex design requirements. By integrating advanced prompting engineering and fine-tuned LLMs, this approach showcases the transformative potential of AI in FPGA development, offering a robust framework for developing efficient and highly customised software solutions.

Use case 4 (vertical)
Next-Generation Battery Management in Electrical Vehicles (EVs) with AI-Driven TinyML Solutions: This use case employs Model-Driven TinyML to enhance battery Remaining Useful Life (RUL) prediction for Electric Vehicles (EVs) through a privacy-preserving, collaborative AI system. Combining graph neural networks (GNNs), large language models (LLMs), and federated learning, the system processes diverse data sources while maintaining privacy by avoiding raw data sharing. EV software engineers, TinyML developers, and data privacy specialists collaborate to develop a robust platform capable of real-time, accurate RUL predictions. The approach integrates advanced AI methodologies, including GNNs for sensor data analysis, LLMs for insights from unstructured data, CNNs for multi-dimensional sensor data, and Recursive Neural Networks (RNNs) for sequence predictions. Knowledge graphs provide contextualisation, while LSTMs, Autoencoders, and Random Forest Trees (RFTs) detect anomalies. Model pruning optimises efficiency, and transfer learning enhances robustness. Real-time predictions with self-repairing capabilities are achieved via federated learning, while self-adaptive algorithms and dynamic slicing with Bayesian Optimisation refine models and optimise resource allocation. The resulting federated TinyML model delivers accurate RUL predictions, maintains data privacy, and supports seamless deployment across platforms with reduced computational overhead.

Use case 5 (vertical)
AI-Powered GIS Integration for Smart Energy Management: This use case leverages the AI-SysDev platform to enhance the efficiency and reliability of Geographical Information Systems (GIS) for smart energy management. By integrating GIS data with energy demand and supply information, along with geographical and meteorological insights, the platform enables optimised energy distribution and seamless integration of renewable energy sources. Key actors include GIS software engineers, AI developers, energy management specialists, and data scientists.

The solution employs AI-driven techniques to generate user stories and functional tests using LLMs, integrate diverse datasets, and optimise energy distribution networks through machine learning. Predictive modelling forecasts energy demand and supply fluctuations, while renewable energy analysis evaluates output and its grid impact. Dynamic resource allocation and optimisation ensure real-time adjustments, with unit tests and simulations validating system performance. Deployment includes monitoring and feedback mechanisms for continuous improvement. The resulting AI-powered GIS platform minimises energy losses, enhances network reliability, and facilitates renewable energy integration. With self-adaptive mechanisms to address anomalies and optimise operations, this use case demonstrates advanced methodologies for sustainable, efficient, and reliable energy management systems.

Expected impact

AI4SWEng will have a positive impact on boosting responsible software engineering methods, tools, and best practices, leveraging novel AI and data technologies. Specifically, it seeks to enhance software development and maintenance for multi-architecture systems through efficient and agile modelling, verification, validation, vulnerability assessment, and mitigation. The AI4SWEng initiative will deliver a transformative toolset comprising the AISysDev Tool and the

TAT integrated over the Suite (KIO1). These tools aim to revolutionise the agile software development pipeline by achieving significant improvements in efficiency and quality across multiple metrics, including reliability, resilience, and security of software solutions, as well as energy efficiency aligned with the Green Deal objectives and United Nations (UN) Sustainable Development Goals (SDG)[57]. The project is expected to reduce errors, bugs, and developer stress while boosting productivity, creativity, and code reuse. Additional benefits include faster time-to-market, enhanced profitability, and improved job satisfaction, ultimately fostering a higher quality work experience in software engineering.

The AI4SWEng project positions Europe as a global leader in sustainable, AI-driven software engineering. By aligning with the Green Deal and UN SDG goals and fostering strategic autonomy, it ensures competitive, environmentally responsible innovation across digital domains. The expected impacts of the project can be summarised in seven dimensions as listed below:

i. Shortened Software Development Time and Reduced Energy Consumption

ii. Efficient and Reliable Software Development Ecosystem

iii. Commercialisation of AI4SWEng Tools as Next-Generation AI-Powered IDEs

iv. Economic Impact and Cost Savings

v. Strategic Advantages in AI and Data for Europe's Open Strategic Autonomy

vi. Reinforced European Industry Leadership Across the Digital Supply Chain

vii. Robust Industrial and Technology Presence in Diverse Applications

Ambition

Aligned with the impact dimensions of AI4SWEng, the consortium has a strong ambition to create a wider impact on the economic, technological, social, and environmental levels.

Economic impact

The AI4SWEng project is poised to drive a transformative economic impact within the software development sector, fostering innovation, sustainability, and industrial competitiveness. By empowering SMEs, lowering market entry barriers, and advancing green software development practices it stimulates economic activity and aligns with the EU objectives for a sustainable economy. AI4SWEng enhances the EU leadership in sustainable software development through industrial competitiveness, alignment with the principles of Industry 5.0 and contributions to the digital and green transition. Its innovations promise significant cost reductions, operational efficiency, and productivity gains, making the EU a more attractive destination for investment while driving R&D momentum and enabling global market penetration. This positions AI4SWEng as a catalyst for economic growth, technological advancement, and sustainable mobility in the evolving global economy.

Technological impact

The AI4SWEng project is set to revolutionise the software development landscape for EU research on automotive, mobility, energy, pharmacy and medical sectors, aligning with sustainable economy trends and policy directives. By enhancing AI-based intelligent software development, the project lowers market entry barriers for SMEs and enables dynamic responses to market demands, fostering greater accessibility and innovation. Through its strong emphasis on intelligent software testing technologies, AI4SWEng ensures optimal software performance and provides a competitive edge in rapidly evolving areas, such as generative and trustworthy AI. Demonstrating these advancements through targeted use cases, the project highlights measurable improvements in competitiveness and sustainability, showcasing its transformative potential for AI-driven software development practices across the automotive, health, energy and broader industrial sectors.

Societal impact

The AI4SWEng project delivers a multifaceted societal impact by transforming the targeted sectors with a strong emphasis on social sustainability and workforce development. By integrating AI-based intelligent software development technologies, the project enhances accessibility and cost-effectiveness, particularly benefiting SMEs. Its commitment to sustainable practices aligns with societal demands for environmentally responsible production, driving efficiency and innovation. Additionally, AI4SWEng investigates social indicators and frameworks to assess the broader social performance implications of AI-driven technologies, ensuring a balanced approach to technological advancement that addresses both economic and societal needs.

Environmental impact

Sustainability has become a critical concern for the 21st century, with the software industry, a key driver of EU economic growth, now at the forefront of integrating sustainable practices. AI4SWEng addresses this imperative by advancing sustainable software development and providing Software-as-a-Service (SaaS) solutions that minimise environmental footprints. By embedding environmental considerations into core operations, the project reflects the industry's commitment to ecological sustainability and sets a precedent for innovation. AI4SWEng promotes resource efficiency and reduced environmental impact, offering tangible pathways to align software development with broader sustainability goals, thereby redefining the relationship between technology, mobility, and environmental responsibility.

Conclusion

The AI4SWEng project represents a significant leap forward in software engineering, uniting leading experts across

Europe to address critical challenges in multi-architectural, resource-constrained systems. By integrating trustworthy AI and advanced LLMs, the project aims to revolutionise the software development lifecycle with a focus on efficiency, reliability, security, and sustainability. The AI4SWEng transformative platform, including a comprehensive development suite, will empower developers with end-to-end support for agile workflows, reducing pain points, fostering creativity, and improving productivity. Through its innovative approach to integrating AI-driven solutions into software engineering, AI4SWEng aligns with societal and environmental goals, contributing to sustainable mobility and advancing European leadership in digital transformation. This initiative sets a precedent for the next era of intelligent, ethical, and sustainable software development, ensuring a lasting impact on industry, research, and society. In upcoming years, the AI4SWEng Consortium will delve into the KIOs and deepen research on these topics by advancing research on novel generative AI, testing and automation techniques.

Ethics and consent

Ethical approval and consent were not required.

Data availability

No data are associated with this article.

Acknowledgments

The authors thank to researchers who had taken a central role in preparing the AI4SWEng proposal. The authors are also grateful to all partners in the consortium who supported the writing process and shared crucial information about the project activities.

References

1. Batarseh FA, Mohod R, Kumar A, *et al.*: **The application of artificial intelligence in software engineering: a review challenging conventional wisdom.** In: *Data Democracy*. 2020; 179–232.
Publisher Full Text
2. Financial Times: **AI-powered coding pulls in almost $1bn of funding to claim, 'killer app' status**. 2023; Accessed: 05.01.2025.
Reference Source
3. IBM: **AI in software development**. 2025; Accessed: 05.01.2025.
Reference Source
4. Hou X, Zhao Y, Liu Y, *et al.*: **Large Language Models for software engineering: a systematic literature review.** *ACM Trans Softw Eng Methodol.* 2024; **33**(8): 1–79.
Publisher Full Text
5. Fan A, Gokkaya B, Harman M, *et al.*: **Large Language Models for software engineering: survey and open problems.** In: *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE).* IEEE, 2023; 31–53.
Publisher Full Text
6. Özkaya I: **Application of Large Language Models to software engineering tasks: opportunities, risks, and implications.** *IEEE Softw.* 2023; **40**(3): 4–8.
Publisher Full Text
7. AI4SWEng Consortium: **AI Engineering Suite to support Agile Efficient Software Engineering (AI4SWEng).** EU-Funded Project. 2024; Accessed: 05.01.2025.
Reference Source
8. Outeda CC: **The EU's AI act: a framework for collaborative governance.** *Internet Things.* 2024; **27**: 101291.
Publisher Full Text
9. Lano K, Xue Q: **Code generation by example using symbolic machine learning.** *SN Comput Sci.* 2023; **4**(2): 170.
Publisher Full Text
10. European Commission: **European industrial strategy**. 2021; Accessed: 05.01.2025.
Reference Source
11. European Commission: **COM/2019/640, Communication on The European Green Deal**. 2019; Accessed: 05.01.2025.
Reference Source
12. European Commission: **A Europe fit for the digital age, Empowering people with a new generation of technologies.** [n. d.]. Accessed: 05.01.2025.
Reference Source
13. Sainio K: **Generative Artificial Intelligence assisting in agile project pain points.** Master's thesis. Tampere University, 2023.
Reference Source
14. Feng Z, Guo D, Tang D, *et al.*: **CodeBERT: a pre-trained model for programming and natural languages.** *arXiv preprint.* arXiv:2002.08155, 2020.
Publisher Full Text
15. Kim H, Jiang Y, Kannan S, *et al.*: **DeepCode: feedback codes via deep learning.** *IEEE J Sel Areas Inf Theory.* 2020; **1**(1): 194–206.
Publisher Full Text
16. Alon U, Zilberstein M, Levy O, *et al.*: **code2vec: learning distributed representations of code.** *Proc ACM Program Lang.* 2019; **3**(POPL): 1–29.
Publisher Full Text
17. Prasad D, Jena AK, Das H, *et al.* (Eds.): **Automated Software Testing: foundations, applications and challenges.** Springer, Singapore, 2020.
Publisher Full Text
18. Nguyen N, Nadi S: **An empirical evaluation of GitHub Copilot's code suggestions.** In: *Proceedings of the 19th International Conference on Mining Software Repositories*. 2022; 1–5.
Publisher Full Text
19. Li Y, Choi D, Chung J, *et al.*: **Competition-level code generation with AlphaCode.** *Science.* 2022; **378**(6624): 1092–1097.
PubMed Abstract | **Publisher Full Text**
20. Roziere B, Lachaux MA, Chanussot L, *et al.*: **Unsupervised translation of programming languages.** In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020; 20601–20611.
Publisher Full Text
21. Puri R, Kung DS, Janssen G, *et al.*: **CodeNet: a large-scale AI for code dataset for learning a diversity of coding tasks.** *arXiv preprint.* arXiv:2105.12655, 2021.
Publisher Full Text
22. Finnie-Ansley J, Denny P, Becker BA, *et al.*: **The robots are coming: exploring the implications of OpenAI Codex on introductory programming.** In: *Proceedings of the 24th Australasian Computing Education Conference*. 2022; 10–19.
Publisher Full Text
23. JetBrains: **JetBrains AI**. 2025; Accessed: 27.01.2025.
Reference Source
24. Lanese I, Schultz UP, Ulidowski I: **Reversible execution for robustness in embodied AI and industrial robots.** *IT Professional.* 2021; **23**(3): 12–17.
Publisher Full Text
25. Cingolani D, Pellegrini A, Quaglia F: **RAMSES: Reversibility-Based Agent Modelling and Simulation Environment with Speculation-Support.** In:

Euro-Par 2015: Parallel Processing Workshops. Lecture Notes in Computer Science, Springer, Cham, 2015; **9523**: 603–614.
Publisher Full Text

26. ReVA Reverse Engineering Assistant: **ReVA - Reverse Engineering Assistant**. 2025; Accessed: 27.01.2025.
Reference Source

27. Hao Y, Chen W, Zhou Z, *et al.*: **E&V: prompting Large Language Models to perform static analysis by pseudo-code execution and verification.** *arXiv preprint.* arXiv:2312.08477, 2023.
Publisher Full Text

28. Alokla A, Gad W, Nazih W, *et al.*: **Retrieval-based transformer pseudocode generation.** *Mathematics.* 2022; **10**(4): 604.
Publisher Full Text

29. Cui W, Peinado M, Cha SK, *et al.*: **RETracer: triaging crashes by reverse execution from partial memory dumps.** In: *Proceedings of the 38th International Conference on Software Engineering*. 2016; 820–831.
Publisher Full Text

30. Ge X, Niu B, Cui W: **Reverse debugging of kernel failures in deployed systems.** In: *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 2020; 281–292.
Reference Source

31. Canavese D, Regano L, Lioy A: **Computer-aided reverse engineering of protected software.** In: *Digital Sovereignty in Cyber Security: New Challenges in Future Vision. CyberSec4Europe 2022. Communications in Computer and Information Science*, Springer, Cham, 2023; **1807**: 35–49.
Publisher Full Text

32. Isgrò F, Ferraris SD, Colombo S: **AI-enabled design tools: current trends and future possibilities.** In: *Congress of the International Association of Societies of Design Research.* Springer Nature Singapore, Singapore, 2021; 2836–2847.
Publisher Full Text

33. Jánki ZR, Bilicki V: **Rule-based architectural design pattern recognition with GPT models.** *Electronics.* 2023; **12**(15): 3364.
Publisher Full Text

34. Stoica BA, Sahoo SK, Larus JR, *et al.*: **Statistical program slicing: a hybrid slicing technique for analyzing deployed software.** *arXiv preprint.* arXiv:2201.00060, 2021.
Publisher Full Text

35. Busse F, Gharat P, Cadar C, *et al.*: **Combining static analysis error traces with dynamic symbolic execution (experience paper).** In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022; 568–579.
Publisher Full Text

36. Pham P, Nguyen V, Nadi S, *et al.*: **A review of AI-augmented end-to-end test automation tools.** In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022; 1–4.
Publisher Full Text

37. Dias T, Batista A, Maia E, *et al.*: **TestLab: an intelligent automated software testing framework.** In: *International Symposium on Distributed Computing and Artificial Intelligence.* Springer Nature Switzerland, Cham, 2023; 355–364.
Publisher Full Text

38. Ajorloo S, Jamarani A, Kashfi M, *et al.*: **A systematic review of machine learning methods in software testing.** *Appl Soft Comput.* 2024; **162**: 111805.
Publisher Full Text

39. Xie X, Ma L, Juefei-Xu F, *et al.*: **DeepHunter: a coverage-guided fuzz testing framework for deep neural networks.** In: *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2019; 146–157.
Publisher Full Text

40. Allamanis M, Jackson-Flux H, Brockschmidt M: **Self-supervised bug detection and repair.** *Adv Neural Inf Process Syst.* 2021; **34**: 27865–27876.
Publisher Full Text

41. Wong MF, Guo S, Hang CN, *et al.*: **Natural language generation and understanding of big code for AI-assisted programming: a review.** *Entropy (Basel).* 2023; **25**(6): 888.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

42. Nayan K, Paswan KK, Sharma VB, *et al.*: **Recent advancements in AI-assisted drug design and discovery systems.** In: *Industry 4.0 and Healthcare: Impact of Artificial Intelligence*. Springer Nature Singapore, Singapore, 2023; 19–36.
Publisher Full Text

43. Radharapu B, Robinson K, Aroyo L, *et al.*: **AART: AI-assisted red-teaming with diverse data generation for new LLM-powered applications.** *arXiv preprint.* arXiv:2311.08592, 2023.
Publisher Full Text

44. Patki N, Wedge R, Veeramachaneni K: **The synthetic data vault.** In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA).* IEEE, 2016; 399–410.
Publisher Full Text

45. Tijana Nikolic: **Data Quality Wrapper.** n.d. Accessed: 19.03.2025.
Reference Source

46. Corallo G, Papotti P: **FINCH: prompt-guided Key-Value cache compression for Large Language Models.** *Trans Assoc Comput Linguist.* 2024; **12**: 1517–1532.
Publisher Full Text

47. Pennington J, Socher R, Manning CD: **Glove: global vectors for word representation.** In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014; 1532–1543.
Publisher Full Text

48. Umer M, Imtiaz Z, Ahmad M, *et al.*: **Impact of convolutional neural network and FastText embedding on text classification.** *Multimed Tools Appl.* 2023; **82**(4): 5569–5585.
Publisher Full Text

49. Lachaux MA, Roziere B, Chanussot L, *et al.*: **Unsupervised translation of programming languages.** *arXiv preprint.* arXiv:2006.03511, 2020.
Publisher Full Text

50. Angstadt K, Wadden J, Dang V, *et al.*: **MNCaRT: an open-source, multi-architecture automata-processing research and execution ecosystem.** *IEEE Comput Arch Lett.* 2017; **17**(1): 84–87.
Publisher Full Text

51. Hidalgo I, Fenández-de Vega F, Ceberio J, *et al.*: **Sustainable Artificial Intelligence systems: an energy efficiency approach.** Authorea Preprints, 2023.
Publisher Full Text

52. Dutta L, Bharali S: **TinyML meets IoT: a comprehensive survey.** *Internet Things.* 2021; **16**: 100461.
Publisher Full Text

53. Fioraldi A, Maier D, Eißfeldt H, *et al.*: **AFL++: combining incremental steps of fuzzing research.** In: *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. 2020.
Reference Source

54. Ding ZY, Le Goues C: **An empirical study of OSS-Fuzz bugs.** In: *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR).* IEEE, 2021; 131–142.
Publisher Full Text

55. Cadar C, Dunbar D, Engler DR: **KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs.** In: *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2008; 209–224.
Reference Source

56. Angr: **Angr: open-source binary analysis platform for Python.** [n. d.]. Accessed: 19.03.2025.
Reference Source

57. Adamowicz M: **Green deal, green growth and green economy as a means of support for attaining the sustainable development goals.** *Sustainability.* 2022; **14**(10): 5901.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ? ? ✗

Version 1

Reviewer Report 23 February 2026

https://doi.org/10.21956/openreseurope.22925.r65232

© **2026 Lofstead J.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✗

Jay Lofstead
Sandia National Laboratories, Albuquerque, USA

This article provides a long list of desired outcomes for software engineering using LLMs. Everything anyone wants LLMs to be able to do is present. However, what is presented here is fundamentally flawed in several different ways:

1. SO4 talks about auditable AI. What this ignores is that the AI software development engines are egregious copyright violators and a new engine must be built from scratch using only properly usable source code that has been vetted for correctness and security. This is an immense task considering the generality proposed in this letter. Ethically, this entire proposal, because it is not building such a new model, is fundamentally flawed.
2. Since this is intended to be open source, how does that help Europe with workforce development? Any open source, unless it is not really open source, can be used by anyone. That means that if the tools proposed are created, they will migrate to the rest of the world. Europe would only benefit from the investment if there were a way to keep them proprietary to Europe in some way.
3. The example use cases have problems. Use case 1 presupposes the PM is capable of describing the software architecture and design sufficiently that the generated items can be useful for building the system. The architecture and design are for the technical team and not the PM to develop. It is unlikely the PM, unless they are a world class software designer, could even do a passable job at this task using the AI assistance. The constant struggle with software is not writing lines of code. It is figuring out how to describe what is needed so the right code can be written, This is the requirements gathering and design generation phase. Rarely, if ever, is this done sufficiently to make automatic code generation even remotely feasible. The best standard used today is to describe target functionality and parameters and let the creativity flow on how to best meet that in code. Then validation that it meets standards. No one writes sufficiently detailed specs to generate code from specs. From this reviewer's time working for Siemens in the 1990s, they tried to write specs that good, but failed 100% of the time. Use case 3 is describing already existing tools that do not need AI to be successful. As these are standard templates, it is a simple replacement. Medical software has been doing this for decades. Use case 4 is a list of AI terms without enough concrete evidence of how it will work to believe it. Federated learning among EVs

is a great idea, but the challenges with dealing with a changing environment is paramount importance. The Waymo failure in San Francisco the end of December 2025 is a prime example. Yes, the technology is improving, but this description does not state any problems, but gives a lot of buzzwords as solutions. Case 5 is also in production today. Energy generation adaptation from prediction and realtime observation has been an active research area for many years and been in use already.
3. The economic impact listed there is optimistic. Green software development practices using horrendously power hungry and expensive AI models does not help the green footprint.

Overall, this is an ambitious project. Unfortunately, it is strong on potentials, but light on evidence that the potentials are reachable. For example, the code generation is proving easy for common things like webpages, but impossible for things like science and engineering codes as the number of training examples is just too small to be useful. AI will help on these common tasks, but anything on that blurry line between engineering and research is unlikely to be possible with AI. The creativity required is often from an inherent understanding of a broad array of tradeoffs and choosing which ones make the most sense for this case. Each time someone approaches the same problem, if they are a skilled engineer, evaluate all of these factors and create the systems according to those parameters. Each one can be similar or fundamentally different, depending on these factors. Having skilled software requirements gatherers, designers, and engineers are hard to replace. Eliminating work for those doing relatively simple, but repetitive things is laudable, but it doesn't upskill those people to being able to do the hard things AI cannot.

The authors should spend time with deeply experienced large system architects, designers, and developers to understand what problems could best be solved with AI and propose to work on those. What is proposed does not match that list as currently written.

Is the rationale for the Open Letter provided in sufficient detail? (Please consider whether existing challenges in the field are outlined clearly and whether the purpose of the letter is explained)

Yes

Does the article adequately reference differing views and opinions?

No

Are all factual statements correct, and are statements and arguments made adequately supported by citations?

No

Is the Open Letter written in accessible language? (Please consider whether all subject-specific terms, concepts and abbreviations are explained)

Yes

Where applicable, are recommendations and next steps explained clearly for others to follow? (Please consider whether others in the research community would be able to implement guidelines or recommendations and/or constructively engage in the debate)

No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Software egnineering; trustable AI; Ethics in ML; ethics in HPC; high performance computing storage; scientific and engineering workflows

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Reviewer Report 09 February 2026

https://doi.org/10.21956/openreseurope.22925.r68370

© 2026 Lanubile F. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Filippo Lanubile
University of Bari, Bari, Italy

This open letter presents the AI4SWEng project, a Horizon Europe-funded Research and Innovation Action (2025–2028) involving 15 partners across eight countries. The project aims to develop an AI-powered software engineering suite that supports the entire software development lifecycle—from requirements engineering and code generation to testing, debugging, security, and energy-efficient deployment—leveraging LLMs and trustworthy AI principles. The letter describes six Specific Objectives (SOs), thirteen Key Innovation Outcomes (KIOs), five validation use cases spanning healthcare, electric vehicles, FPGA development, GIS energy management, and general SDLC optimization, and articulates expected economic, technological, societal, and environmental impacts. The article also situates the project within the EU policy landscape, referencing the AI Act, GDPR, the Green Deal, and the Digital Single Market strategy.

The AI4SWEng project addresses a timely and important topic at the intersection of AI and software engineering. The consortium is well-qualified, the scope is ambitious, and the vision is compelling. However, in its current form, the open letter reads more like a project proposal than a focused scientific communication. The manuscript would benefit substantially from condensation, clearer articulation of novelty, engagement with critical perspectives, and inclusion of evaluation methodology. I recommend reservations for approval. I believe a revised version that addresses these concerns would make a valuable contribution to the discourse on AI-assisted software engineering.

Detailed Comments
- The open letter format typically calls for a concise, focused communication that presents a clear position or call to action. At 17 pages, including references, this manuscript far exceeds what is expected of an open letter and instead resembles a project deliverable or technical roadmap. The authors should consider significantly condensing the document, focusing on the core message they wish to convey to the software engineering research community.

- The authors should avoid overlapping content. For instance, RAG is introduced in at least four separate locations with essentially the same explanation each time.
- While the thirteen KIOs are described in considerable detail, their novelty relative to the state of the art is insufficiently articulated. Highlighting the specific advancements would substantially strengthen the paper.
- For an open letter aimed at engaging the research community, at least a high-level description of how success will be measured is necessary. For instance, which code generation benchmarks (e.g., HumanEval, MBPP, SWE-bench) will be used? How will developer productivity gains be quantified?
- Important LLM-based coding tools beyond Copilot and Codex are missing: Claude Code and Gemini Code Assist. Same gap for benchmarks such as SWE-bench and CrossCodeEval.
- Although the authors provide an abbreviation table, many domain-specific compound terms are used without adequate explanation.
- The article presents an overwhelmingly positive view of AI in software engineering without engaging with critical perspectives. A balanced treatment would strengthen the letter.

Is the rationale for the Open Letter provided in sufficient detail? (Please consider whether existing challenges in the field are outlined clearly and whether the purpose of the letter is explained)
Yes

Does the article adequately reference differing views and opinions?
Partly

Are all factual statements correct, and are statements and arguments made adequately supported by citations?
Yes

Is the Open Letter written in accessible language? (Please consider whether all subject-specific terms, concepts and abbreviations are explained)
Partly

Where applicable, are recommendations and next steps explained clearly for others to follow? (Please consider whether others in the research community would be able to implement guidelines or recommendations and/or constructively engage in the debate)
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Software Engineering

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 02 January 2026

https://doi.org/10.21956/openreseurope.22925.r64718

© 2026 Nguyen-Duc A. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Anh Nguyen-Duc
Norwegian University of Science and Technology, Trondheim, Norway

This open letter serves as a position paper and early project-level communication of the AI4SWEng initiative, a Horizon Europe-funded R&I project running from 2025–2028. The authors, representing a broad and interdisciplinary consortium of 15 partners across Europe and associated countries, present a comprehensive overview of the project's vision, objectives, technical architecture, innovation strategies, and expected impacts across economic, technological, societal, and environmental dimensions.

The letter outlines a future agenda: to deliver an AI-powered software development suite (the AI4SWEng Engineering Suite) capable of supporting all stages of the software lifecycle through integration of large language models (LLMs), advanced automation, responsible AI practices, and policy alignment (e.g., GDPR, AI Act). It touches on key challenges in software engineering and outlines an ambitious plan with Key Innovation Outcomes (KIOs) and use cases spanning several high-impact sectors.

Overall, this is a well-articulated, visionary, and technically rich document. However, it is notably long for an open letter and reads more like a project deliverable or technical roadmap than a focused, accessible communication aimed at the general scientific community. The manuscript can be significantly improved in terms of clarity of audience and purpose, conceptual focus, measurable and well-scoped objectives, organization of content, methodological transparency, grounding in recent literature, and articulation of the novelty and scope of proposed innovations.

Audience and purpose ambiguity
The open letter lacks a clearly defined audience and oscillates between multiple roles—serving simultaneously as a manifesto, a technical roadmap, a funding dissemination document, and a quasi-scientific article. This multiplicity of intentions results in a document that is difficult to position within conventional scientific communication formats. For instance, some parts appear to target researchers with highly technical detail, while others adopt a broad, policy-oriented tone meant for stakeholders or funding bodies. This shifting register hinders both accessibility for non-experts and depth for specialists.

Excessive jargon and lack of conceptual clarity
Throughout the manuscript, the language is heavily saturated with buzzwords and layered terminology—terms like "reverse execution," "dynamic slicing," "energy-aware model-driven enhancement," "X-by-design," and "ontology-based prompt engineering" are introduced with minimal explanation or grounding in existing literature. While these phrases suggest advanced technical ambition, they often lack precise definitions, conceptual unpacking, or citations, which undermines their credibility. The overuse of unexplained compound terms may alienate readers unfamiliar with niche concepts and obscure the practical or scientific significance of the proposed innovations. Greater terminological discipline, including first-use definitions, explanatory footnotes, and references to foundational literature, is necessary to improve both clarity and

transparency.

Redundancy and structural repetition
A recurring issue across the letter is redundancy in content. Several core themes—such as the use of large language models (LLMs) in code generation, the integration of responsible AI practices, and the benefits of AI for developer productivity—are reiterated in multiple sections (e.g., introduction, scientific background, innovation landscape, and use cases), often using slightly varied phrasing but delivering largely the same message. This repetition inflates the document's length without contributing substantial new insights and can overwhelm readers with unnecessary detail. A more concise structure would consolidate these themes into focused, non-redundant sections, improving the overall readability and impact of the document.

Lack of evaluation methodology and risk analysis
Despite the breadth and ambition of the AI4SWEng initiative, the open letter does not provide a coherent framework for evaluating the project's success. There is little to no detail on validation protocols, performance metrics, benchmarking strategies, or testing plans for the proposed tools and methods. For example, how the AI-SysDev Tool or the TAT will be assessed for efficiency, accuracy, or usability is not specified. Equally important, there is no discussion of risks—technical, ethical, or organizational—that could jeopardize the project's goals. Issues like LLM hallucination, over-reliance on AI, lack of developer trust, or integration failures are entirely absent. Including a robust evaluation and risk mitigation plan would significantly strengthen the scientific credibility of the proposal.

Concerns with the formulation of Specific Objectives (SOs)
The six Specific Objectives (SOs) outlined in the letter are overly broad, conceptually overloaded, and uneven in their level of specificity and measurability. For instance, SO1 combines architectural analysis, log analysis, bug prediction, and fault recovery into a single objective, diluting focus and making it hard to evaluate. SO2 similarly merges distinct activities like requirements engineering, architecture specification, and code generation without clarifying the interdependencies or defining measurable targets. Objectives like SO4 and SO5 refer to "auditable" AI and "socio-ethical acceptability" but provide no operational framework for these outcomes. While the ambition is commendable, these objectives require clearer scoping, with quantifiable goals and indicators of success. A more structured articulation—linking each SO to concrete deliverables and expected impacts—would improve both their clarity and feasibility.

Conceptual overload and lack of focus
Conceptually, the letter attempts to cover too many facets of modern software engineering simultaneously—ranging from formal requirements engineering, reverse execution, automated testing, energy-efficient LLMs, and cyber-physical systems, to ethical AI and developer training. This thematic sprawl dilutes the coherence of the project's vision and raises doubts about feasibility. Instead of attempting to solve all contemporary challenges in AI-assisted software engineering, the authors could focus on a narrower set of tractable problems and more clearly define the project's unique contribution to the field.

Unstructured and incomplete literature review
The literature review portion of the letter, while extensive in terms of citations, lacks organization and thematic coherence. The referenced tools and studies (e.g., DeepHunter, Code2Vec, Copilot, ReVA) are presented in an enumerative rather than analytical fashion. Key recent tools (i.e. Google

Gemini Code Assist) and research in the LLM space and recent FSE/ICSE/NeurIPS/ICML benchmarks on LLM performance in software tasks—are notably absent. Furthermore, the review disproportionately focuses on automated testing and dynamic slicing, while other critical areas (e.g., architectural planning, prompt engineering, model robustness) receive minimal attention. To be effective, the literature review should be structured by topic, directly linked to project objectives, and framed to highlight where AI4SWEng extends or challenges the current state of the art.

Ambiguity in the role and novelty of KIOs
The Key Innovation Outcomes (KIOs) form the backbone of the AI4SWEng architecture, yet their relation to the state of the art and their added value are not sufficiently justified. While each KIO is described in technical language, their novelty is often unclear, and it remains difficult to understand how they differ from or improve upon existing tools. For example, KIO2 (bug localization via reverse execution) or KIO11 (AI-powered test automation) are areas already heavily researched with existing tools available. Without clear benchmarking or comparative baselines, the reader is left unsure about the real innovation or impact. A comparative matrix aligning KIOs with existing tools and articulating the improvements would help substantiate their contributions and justify the development effort.

Unclear scope of software development focus (Agile vs. SDLC, application domains)
The letter refers variously to "agile development," "DevOps/MLOps," and the broader "software development lifecycle (SDLC)" but does not specify which development paradigms are primarily targeted. Similarly, the types of software systems—whether embedded, safety-critical, enterprise, mobile, or general-purpose—are not clearly delineated. While use cases touch on healthcare, EVs, and GIS, it is unclear whether these are demonstrators or representative domains. This lack of focus introduces confusion: for example, many techniques described (e.g., formal verification, symbolic execution, energy-aware compilation) are more applicable to regulated or embedded systems, whereas agile methodologies typically prioritize rapid iteration and adaptability. The authors should clarify whether AI4SWEng targets high-assurance systems, general SDLC acceleration, or a hybrid. Specifying the intended software domains and development paradigms would allow for better evaluation of relevance, feasibility, and impact.

Is the rationale for the Open Letter provided in sufficient detail? (Please consider whether existing challenges in the field are outlined clearly and whether the purpose of the letter is explained)

Yes

Does the article adequately reference differing views and opinions?

Partly

Are all factual statements correct, and are statements and arguments made adequately supported by citations?

Yes

Is the Open Letter written in accessible language? (Please consider whether all subject-specific terms, concepts and abbreviations are explained)

Partly

Where applicable, are recommendations and next steps explained clearly for others to follow? (Please consider whether others in the research community would be able to implement guidelines or recommendations and/or constructively engage in the debate)
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Empirical Software Engineering

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.