Ref. Ares(2025)9907923 - 17/11/2025

Deliverable D2.4

Preliminary requirements scoping of AI-powered development and testing tools including data preparation activities

Project Number	**101189908**
Project Title	AI Engineering Suite to Support Agile Efficient Software Engineering
Project Acronym	AI4SWEng
Title of Deliverable	Preliminary requirements scoping of AI-powered development and testing tools including data preparation activities
Version	V1.0
Due Date of Delivery	M5
Distribution level	PUBLIC

Work Package	**WP2**
Responsible Partner	FORTH
Editor(s)	Prof. Dimitrios I. Fotiadis (FORTH)
Contributor(s)	Prof. Dimitrios I. Fotiadis (FORTH), Vasileios Pezoulas (FORTH), Nikolaos Tachos (FORTH), Fanis Gerodimos (FORTH), Stelios Sfakianakis (FORTH), Stefanos Gkikas (FORTH), Çağrı Terzibaş (ERARGE), İbrahim Arif (ETCH), Oğuzhan Herkiloğlu (BIT), Atta Badii(UREAD), Omer Oztin (AI4)
Reviewer(s)	Ali Atalay (AI4SEC), Alper Kanak (ERARGE), Sébastien Bois (CEA)
Keywords	Requirements Analysis, Framework, Architecture, KIO

Co-funded by the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101189908.

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

Abstract: *This document, Deliverable D2.4, presents the preliminary results of the requirements scoping and analysis for the AI4SWEng project's core AI-powered development and testing tools. It details the methodologies used to gather and define both Functional and Non-Functional Requirements (FR/NFR), including those for essential data preparation activities. Based on an initial user needs assessment and technical constraints, the deliverable specifies the requirements for the Key Innovation Outcomes (KIOs) developed within Work Package 2. The defined requirements cover critical aspects such as data privacy and GDPR compliance, tool compatibility, multi-language support, and explainability outcome tracking. The requirements are classified, prioritized (Mandatory, Desirable, Optional), and documented to guide the subsequent design and implementation phases of the AI4SWEng suite.*

Document Revision History

Version	Date completed	Description
v0.1	01/09/2025	Table of Contents (TOC) circulation
v0.2	17/09/2025	State of the art analysis
v0.3	26/09/2025	Requirements and user needs
v0.4	26/10/2025	Interviews
v0.5	11/11/2025	Conclusions and Appendix
v0.6	18/11/2025	Internal review
v0.7	25/11/2025	Final integration
v1.0	28/11/2025	Final version ready for submission

Statement of Originality

This deliverable paper contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation, or both.

Disclaimer and Copyright

The information appearing in this document has been prepared in good faith and represents the views of the authors. Every effort has been made to ensure that all statements and information contained herein are accurate; however, the authors accept no statutory, contractual or other legal liability for any error or omission to the fullest extent that liability can be limited in law.

This document reflects only the view of its authors. Neither the authors nor the Research Executive Agency nor the European Commission are responsible for any use that may be made of the information it contains. The use of the content provided is at the sole risk of the user. The reader is encouraged to investigate whether professional advice is necessary in all situations.

No part of this document may be copied, reproduced, disclosed, or distributed by any means whatsoever, including electronic without the express permission of the AI4SWEng project Partners. The same applies for translation, adaptation or transformation, arrangement or reproduction by any method or procedure whatsoever.

© Copyright

AI4SWEng Innovation Mission

AI4SWEng: AI Engineering Suite to Support Agile, Efficient Software Engineering	*AI4SWEng represents an excellent chemistry of 15 Partners including experts and practitioners in Model-Driven Software Engineering and AI Engineering (particularly LLMs) from across Europe with five use-cases from Software Development, Healthcare, Cyber-Physical System involving multi-architectural and resource-constrained systems and Next Generation Battery Management in EVs. The AI4SWEng mission is to significantly contribute to the transformation of agile software development into a sector turbocharged for design-by-contact, high efficiency, reliability, and security-privacy compliant delivery. This is to exploit advanced AI-powered software development support to deliver high-quality reliable fast time-to-market solutions that are socially responsible, scalable, sustainable and lend themselves to audible ethical and regulatory compliance. AI4SWEng aims to deliver AI-powered support to software developers to help reduce worker stress past the pain-points in agile SW development pipelines and higher productivity thus enhanced job satisfaction and creativity. Accordingly, the project will develop, deliver and validate, in 4 use-cases, a platform comprising a set of tools including the AISysDev and Test Automation Tool to support the software lifecycle end-to-end for both functional and non-functional deliverables by supporting a host of tasks such as code generation, cross-compilation, advanced debugging and dynamic slicing, adaptation, security-privacy by design, reliability, energy efficiency and adaptation as well as project management, process quality and maintenance support through data-driven insights for informed design decision-making and strategic software development planning. Additionally, with a commitment to user-centred design, it will provide structured design in advanced prompt engineering to support software developer teams to best exploit the capabilities of the platform.*

List of Contributing Beneficiaries

No	Organization name	Short name	Country
1	IDRYMA TECHNOLOGIAS KAI EREVNAS	FORTH	GR
2	AI4SEC OU (SME)	AI4	EE
3	EURECOM GIE (RTO)	EUR	FR
4	Commissariat à l'Énergie Atomique et aux Énergies Alternatives (RTO)	CEA	FR
5	Libera Universita di Bolzano (RTO)	UNIBZ	IT
6	Instituto Tecnológico de Castilla y León (RTO)	ITCL	ES
7	IDENER Research & Development (RTO)	IDE	ES
8	The University of Reading (RTO)	UREAD	UK
9	BITNET Bilisim Hizmetleri Limited (SME)	BIT	TR
10	Innova Integra Limited (SME)	INNO	UK
11	ERGUNLER R&D. Co. Ltd. (SME)	ERG	TR
12	ATPharma (SME)	ATT	TR
13	ERGTECH Research GmbH	ERGTECH	CH
14	HAUTE ECOLE SPECIALISEE DE SUISSE OCCIDENTALE	HES-SO	CH
15	UNIVERSITE DE GENEVE	UNIGE	CH

Table of Contents

1. Introduction 12
1.1 Relation with the AI4SWEng Description of the Action (DoA) 12
1.2 Description of the document's structure 12
2. State-of-the-art (SotA) analysis 13
2.1 AI-enabled code analysis and understanding 13
2.2 Assisted code generation 13
2.3 AI-enabled reverse execution and generation of pseudocode 13
2.4 Requirements and design automation 14
2.5 Code and trace analysis 14
2.6 AI in project management 14
2.7 AI-driven software testing and quality assurance 15
2.8 Data quality assessment, analytics, relevance, balance, integrity maintenance 15
2.9 Gaps and limitations of the state-of-the-art 15
3. Requirements and user needs 16
3.1 Types of users 16
3.1.1 Developers or practitioners 16
3.1.2 Integrators or technology providers 16
3.1.3 Regulators or certification authorities 17
3.1.4 End-user organizations and application stakeholders 17
3.1.5 Research and training communities 17
3.2 Data preparation activities and resources 18
3.2.1 UC1 (AI-enhanced software lifecycle optimization) 18
3.2.2 UC2 (Medical reporting and document automation) 19
3.2.3 UC3 (Large language models for advanced software generation for FPGA) 19
3.2.4 UC4 (AI-driven battery management for electric vehicles) 19
3.2.5 UC5 (AI-powered GIS integration for smart energy management) 20
3.3 Ecosystem actor behavior diagrams 20
3.3.1 AI-driven software development automation 20
3.3.2 AI-based software integration 21
3.3.3 AI-powered cybersecurity analysis 22
3.3.4 Human-centric AI-powered interfaces 24
3.3.5 Data security and privacy compliance 25
3.3.6 AI-powered regulatory compliance 26

3.3.7 Ethics and fairness assessment 27
3.3.8 AI-driven monitoring of changing security guidelines 28
3.3.9 Automated AI training and upskilling 29
3.3.10 AI-assisted decision support 30
3.4 High-level needs of the AI4SWEng tools and components 31
3.5 Preliminary Functional and Non-functional Requirements scoping and elicitation based on the UI-REF methodology 46
3.5.1 UI-REF methodology 47
3.5.2. Functional Requirements (FRs) 48
3.5.3. Non-functional Requirements (NFRs) 51
3.6 Priorities distribution 54
4. Validation 55
4.1 Description of the process 55
4.2 Results 56
4.2.1 Participant profiles 56
4.2.2 Key opinions and insights 57
4.2.2.1 The role of software developers in AI-augmented environments 57
4.2.2.2 How AI can improve software development 58
4.2.2.3 Efficiency improvements in the SDLC phases 58
4.2.2.4 Examples of how AI improves software delivery 58
4.2.2.5 Consequences of AI-driven data management 59
4.3 Requirements validation and prioritization 59
4.3.1 Level of agreement 59
5. AI4SWEng requirements evolving process 60
6. Conclusions 60
References 61
Appendix A: State of the art 64
Appendix B: Preliminary List of Functional Requirements (FRs) 66
KIO1 66
KIO2 67
KIO3 69
KIO4 70
KIO5 72
KIO6 73
KIO7 75

KIO8 77
KIO9 78
KIO10 79
KIO11 82
KIO12 83
KIO13 83
Appendix C: Preliminary List of Non-Functional Requirements (NFRs) 87
KIO1 87
KIO2 88
KIO3 89
KIO4 90
KIO5 92
KIO6 93
KIO7 95
KIO8 96
KIO9 98
KIO10 99
KIO11 102
KIO12 104
KIO13 105
Appendix D: Interview Form 109

List of Figures

Figure 1: AI-driven software development automation - Workflow showing Software Developers, AI Engineers, and DevOps Teams using AI tools for code generation, testing, debugging and deployment. 21
Figure 2: AI-based software integration - Workflow showing collaboration between Software Developers, AI Solution Providers and SME IT Teams to integrate AI models into business software with ongoing support. .. 22
Figure 3: AI-powered cybersecurity analysis - Workflow showing collaboration to strengthen cybersecurity, define policies, ensure privacy, automate compliance, detect threats, respond to alerts, ensure GDPR/ENISA compliance and mitigate risks. 23
Figure 4: Human-centric AI-powered interfaces - Workflow showing UX/UI Designers, AI Developers, and Market Researchers designing intuitive AI interfaces with testing and improvements. 24
Figure 5: Data security and privacy compliance - Workflow showing collaboration to analyze vulnerabilities, enforce standards, verify compliance, prevent data misuse, enhance protection and mitigate risks. 25
Figure 6: AI-powered regulatory compliance - Workflow showing collaboration to provide guidance, assess compliance, generate reports, enforce policies, assess risks, monitor enforcement and define standards. . 26
Figure 7: Ethics and fairness assessment - Workflow showing collaboration to assess bias, ensure transparency, refine models and recommend improvements. 27
Figure 8: AI-driven monitoring of changing security guidelines - Workflow showing collaboration to monitor updates, categorize changes, analyze impacts, provide insights, notify stakeholders and update frameworks. 28
Figure 9: Automated AI Training - Workflow showing Employees accessing courses, HR tracking progress and AI systems generating personalized learning paths. 29
Figure 10: AI-assisted decision support - Workflow showing SME Owners, Business Analysts, and Managers providing data, analyzing trends, generating insights, making data-driven decisions, optimizing operations and improving efficiency. 30
Figure 11: The distribution of requirements by priority level (Mandatory, Desirable and Optional) per KIO. 55
Figure 12: Bar chart showing how functional and non-functional requirements are distributed across the 13 KIOs. 55
Figure 13: The distribution of familiarity levels (Good, Average, Poor) among interviewees. 56
Figure 14: Distribution of key responsibilities identified by participants in AI-augmented software development environmens. (for a more comprenhensive view see Appendix D: Interview Form). 57
Figure 15: Perceived efficiency improvements across the Software Development Life Cycle (SDLC) phases, ranging from “Not at all” (blue) to “Essential for efficiency” (purple). (for a more comprenhensive view see Appendix D: Interview Form). 58
Figure 16: Distribution of the level of agreement of each participant according to the results of the interview process. 59

List of Tables

Table 1: Relation of D2.4 with the Task description from the DoA. 12
Table 2: Standardized user groups and terminology harmonization for the AI4SWEng project. This table defines the five primary user groups targeted by AI4SWEng and establishes consistent terminology to be used across user analyses, use case descriptions, and requirements specifications. 17
Table 3: High-level needs elicited for KIO1. 31
Table 4: High-level needs elicited for KIO2. 32
Table 5: High-level needs elicited for KIO3. 33
Table 6: High-level needs elicited for KIO4. 34
Table 7: High-level needs elicited for KIO5. 35
Table 8: High-level needs elicited for KIO6. 35
Table 9: High-level needs elicited for KIO7. 37
Table 10: High-level needs elicited for KIO8. 38
Table 11: High-level needs elicited for KIO9. 39
Table 12: High-level needs elicited for KIO10. 40
Table 13: High-level needs elicited for KIO11. 42
Table 14: High-level needs elicited for KIO12. 43
Table 15: High-level needs elicited for KIO13. 44

List of Abbreviations

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
AST	Abstract Syntax Tree
CI/CD	Continuous Integration / Continuous Deployment
DL	Deep Learning
FRs	Functional Requirements
HLS	Higl Level Synthesis
IDE	Integrated Development Environment
KIO	Key Innovative Output
KPI	Key Performance Indicator
LLM	Large Language Model
MITM	Man In the Middle
ML	Machine Learning
MLOps	Machine Learning Operations
NFRs	Non-Functional Requirements
NLP	Natural Language Model
OSS	Open-Source Software
QA	Quality Assurance
QAIF	Quality Assurance for Intelligent Functions
RPA	Robotic Process Automation
SDLC	Software Development Life Cycle
SE	Software Engineering
SotA	State-of-the-art
SQA	Software Quality Assurance
TAT	Testing Automation Tool
UI/UX	User Interface/User Experience
UI_REF	User Interface Reference
WP	Work Package

Executive Summary

The AI4SWEng project is an engineering suite that seeks to improve the agility and efficiency of software development. It strives to revolutionise European software engineering by introducing artificial intelligence and data-driven approaches, and by integrating a Responsible Research and Innovation (RRI) attitude throughout the entire software development process. Development approaches struggle to cope with the rising scope, quality, and legal requirements of the next generation of software systems, characterised by ever-increasing maturity levels, distributions, and distributed properties, as well as energy efficiency.

This deliverable defines the functional and non-functional requirements for the AI-powered software development and testing tools within the AI4SWEng project. It begins with a state-of-the-art (SotA) analysis to identify the most advanced AI-driven methods for automated data preparation, code analysis, generation, reverse execution, and testing. Through the analysis, several critical gaps were identified, including: (i) limited end-to-end automation from natural language specifications, (ii) barriers in deep semantic understanding of code, and (iii) technical complexity in reverse execution. Based on these findings, the deliverable consolidates the requirements and user needs identified across the 13 Key Innovative Outputs (KIOs) of the AI4SWEng project. For each KIO, the analysis focused on the collection of inputs, outputs, user interface, and hardware requirements, expected Technology Readiness Level (TRL) evolution, among others. The needs of five primary user groups, including software developers and engineers, technology providers, regulator or certification authorities, end-user organizations, and application stakeholders, and research and training communities, were then identified. Ecosystem actor behavior diagrams were developed to capture the interactions, roles, and dependencies among software engineers, AI engineers, DevOps teams. The collected functional and non-functional requirements are preliminary and represent the first iteration of the AI4SWEng specifications which provide the backbone for the development of the integrated suite. The requirements were prioritized based on feedback from the KIO leaders/contributors with respect to the UI-REF (User Interface Requirements Elicitation Framework). An interview process involving both experienced and less experienced software developers was carried out to validate the relevance and usability of the prioritized requirements. According to their feedback, the interviewees verified that this first iteration of requirements is practical, user-centric, and aligned with the project's objectives.

1. Introduction

1.1 Relation with the AI4SWEng Description of the Action (DoA)

This document is the initial deliverable related to Task 2.4. Table 1 outlines how the work conducted within this deliverable fulfills the objectives defined for the task.

Table 1: Relation of D2.4 with the Task description from the DoA.

DoA Task description	Addressed by D2.4
It includes both functional and non-functional aspects, alongside typical end-user scenarios, to clarify and prioritise the required functionalities to be delivered by each of the tools to support the most common prototypical use-contexts and most challenging software developers' pain-points, WP2, page 40.	The functional and non-functional requirements are reported under Section 3.5 respectively. These requirements are presented in detail in Appendix B (Functional requirements) and C (Non-functional requirements).
Accordingly, inline and in-person interviews will be conducted with software engineering practitioners to elucidate their needs and prioritise the capability set for each tool considering a realistic scoping of the project objectives, WP2, page 40.	The interview process that was conducted with six software engineering practitioners is described in Section 4.1. The results from the interviews are presented in Section 4.2.
These requirements will also serve as integral components of the elicitation process conducted in the subsequent task T2.5, WP2, page 40.	The connection of D2.4 with the activities under T2.5 is discussed in Section 5.

1.2 Description of the document's structure

This deliverable provides an overview of the requirements for AI-powered software development and testing tools within the AI4SWEng project. It aims to help the reader understand the project's basic analysis and the technical needs. The deliverable consists of the following sections. **State-of-the-art (SotA) analysis**. This section provides a thorough overview of the current state of AI-driven tools and methods in software engineering. It points out important improvements and critical gaps. **Requirements and user needs**. This section lists the preliminary list of functional and non-functional needs of the AI4SWEng suite. Moreover, it highlights the demands of five main user groups, for instance software developers, technology providers, and regulators. In addition, it provides use case descriptions and examples of ecosystem actor behavior diagrams. **Validation**. This section provides a description of the interview process which was conducted to assess the priortization of the functional and non-functional requirements based on the UI-REF. Furthermore, this section presents the results from the interview process including the level of agreement among the participants. **AI4SWEng requirements evolving process**. This final section summarizes the main findings of the deliverable (the first iteration of the requirements) and describes how this work will be associated with the subsequent deliverables of WP2. **Conclusions**. This is the last section which summarizes the main findings of the deliverable (the first iteration of the requirements). It also mentions how this deliverable will be associated with the other deliverables of WP2.

2. State-of-the-art (SotA) analysis

AI4SWEng envisages to develop an AI-powered tool that can support every step of the software development lifecycle, varying from the collection of requirements and code writing to advanced debugging, security, energy efficiency and project management. To identify real gaps and unmet demands, a state-of-the-art (SotA) analysis for AI-driven software engineering tools and methods was conducted, including both commercial solutions and cutting-edge research. This analysis underlines the existing capabilities, limitations, and opportunities that AI4SWEng will build upon to deliver next-generation software engineering support.

2.1 AI-enabled code analysis and understanding

AI-enabled methods use natural language processing (NLP), machine learning (ML), and deep learning (DL) to look at the structure and meaning of software code. This helps with code generation, testing, debugging, documentation, and maintenance [1]. The current research in this domain explores several challenges in code representation, synthesis, completion, summarization, recommendation, testing, refactoring, and security [2,3]. To this end several approaches have been proposed, including: (i) CodeBERT which is a pre-trained model for programming and natural languages that puts both domains into one semantic space [4], (ii) DeepCode [5] which uses AI to analyze and understand code by learning from billions of lines of code to find security holes and quality problems in real time, and (iii) Code2Vec [6] which learns distributed representations of code by extracting abstract syntax trees (ASTs) and thus can be used for summarization, classification, and similarity detection.

2.2 Assisted code generation

AI-assisted code generation, such as Copilot [7], integrates large language models (LLMs) into the coding environment to support developers towards the generation of code suggestions from natural language prompts. The Amazon CodeWhisperer [8] provides real-time code recommendations and incorporates a unique security scanning feature to identify and flag code vulnerabilities. Anthropic's Claude Code [9] operates directly in the terminal and interacts with the file system to understand the whole codebase and execute commands. In addition, DeepMind's AlphaCode [10] has the ability to solve competition-level programming challenges through the application of reinforcement learning (RL) to complex coding tasks. Moreover, Facebook AI's TransCoder (source-to-source translator) [11] is a project that focuses on the translation of programming languages in an unsupervised manner. This makes it easier to move legacy code and create multilingual applications. The CodeNet [12] provides a large-scale dataset of 14 million code samples across 50 languages to realize the training and benchmarking of advanced models. Finally, OpenAI's Codex [13] has recently shown performance in tasks like code generation, translation, refactoring, and bug detection. However, there are still challenges to solve in order to capture the complexity of programming language semantics and achieve deeper code understanding.

2.3 AI-enabled reverse execution and generation of pseudocode

Reverse execution allows a system to run backwards, undoing the effects of forward computation. This facilitates error detection and correction, system recovery of previous states, as well as the exploration of alternative execution scenarios. Akgul and Mooney's early work [14] introduced instruction-level reverse execution which allows programs to return to previous states, operating directly at the machine-instruction level. Based on these ideas, frameworks such as RAMSES [15] exploited reversibility and speculation to model agent-based simulations in parallel. In robotics, reversible computation has been applied to industrial assembly operations, where classical AI planning and embodied AI integrate with reversibility to ensure robustness [16]. Moreover, current advances highlight AI-enabled approaches that use reverse execution

along with symbolic reasoning, machine learning, and natural language models. ReVA [17] provides an AI assistant for reverse engineering tasks, whereas pseudocode-based methods are used to generate human-readable representations of executable code and simulate their reverse execution for testing and debugging purposes. Hao et al. [18] proposed the execution and verification of pseudocode using large language models for static analysis. In addition, Akola et al. [19] developed a retrieval-based transformer that combines retrieval and generation techniques to create pseudocode from executable code. AI-driven debugging systems also leverage reverse execution. RETracer and REPT [20] reconstruct execution histories from memory dumps or hardware traces to prioritize, classify, assess and debug software failures in production. Similarly, research on reverse engineering of protected software highlights challenges and opportunities in automating analysis with AI, using machine learning and symbolic execution [21]. Broader surveys on AI in software engineering [22] underline the role of reverse execution in engineering, debugging, and maintenance. However, efficient reverse execution still remains challenging due to factors such as the complexity of unstacking prior execution states.

2.4 Requirements and design automation

Recent research in software engineering focuses on the translation of natural language specifications into precise and readable by machine representations that can be used directly in the development phase. Agile Requirements Designer [23], DOORS [24], and JAMA [25] are tools that assist in documenting and managing requirements. Also, template-based approaches have been proposed to convert unstructured natural language into semi-structured user stories and use cases. However, most automation tools focus on requirements classification rather than generating directly from natural language input [26]. Tools like Zephyr [27] are some of the first attempts towards this direction. At the same time, design automation has advanced through tools such as Balsamiq [28] and Adobe XD [29], which generate visual prototypes from requirement description and through AI-supported approaches that enhance design consistency, reliability and reuse. Some critical contributions include: (i) the organization of automation solutions using design patterns [30], (ii) the exploration of machine learning to detect design patterns in source code with high accuracy [31], (iii) the review of AI-powered design tools for their role in generative and collaborative workflows [32]. These methods show that AI is making progress towards the formalization of requirements and the automation of design, but they also show that end-to-end automation from natural language is still not possible.

2.5 Code and trace analysis

Dynamic program slicing with AI focuses on the isolation of relevant parts of execution using backward execution, replay, taint analysis, or symbolic execution. Stoica [33] introduced statistical program slicing, a hybrid static-dynamic technique that balances accuracy and runtime cost, recovering ~94% of program statements in slices with only 5% overhead. Moreover, the current research in dynamic network slicing [34] underlines several applications of slicing concepts in large-scale, adaptive environments. At the same time, AI-enabled automated trace generation supports the definition and analysis of execution traces, which allows tasks such as reverse engineering, fault localization, and optimization. Furthermore, Shu et al. [35] explored automatic trace generation for indoor navigation systems without manual mapping. In addition, IBM [36] explored AI-driven end-to-end test automation tools that can integrate trace analysis into test generation, execution, and maintenance. All these advancements reveal how important slicing and trace analysis are to make adaptive, AI-driven software engineering feasible.

2.6 AI in project management

AI has become an important part of project management, going beyond the conventional analysis of data to help with projects' planning and execution. AI-supported tools that use machine learning to analyze vast

amounts of old project data, help for more accurate predictions for timelines, resource allocation, and risk assessment [37]. Atlassian Intelligence for Jira and Microsoft's Copilot for Project are two platforms that can automatically make detailed project plans, find dependencies, and point out possible bottlenecks by looking at requirements written in natural language. This process not only reduces the human effort of project managers but also provides data-driven insights to improve decision-making and increase overall project success rates [38].

2.7 AI-driven software testing and quality assurance

AI-driven software testing improves the quality, efficiency, and reliability of software validation through: (i) the generation and prioritization of test cases, (ii) the analysis of results, (iii) the detection of defects, and (iv) the delivery of feedback. Currently, there are several AI-powered frameworks which adopt tests for vulnerability detection, security analysis, and automated test-case generation [39]. AI-enhanced test automation [40] demonstrates how machine learning and computer vision techniques tackle problems that traditional testing struggles with, which aligns with feedback from developers who've actually used these AI-powered tools [41]. In addition, the performance evaluation and structural robustness of deep learning models is an essential component of contemporary quality assurance. Accuracy and loss metrics are standard for training and validating processes, but structural metrics, such as robustness, are essential in safety-critical domains, including autonomous vehicles, medical systems, and cybersecurity [42,43]. White-box testing for deep neural networks (DNNs) expands this scope through the examination of internal architectures, parameters, gradients, and metrics. DeepHunter [44] and DANCe [45] are examples of testing frameworks that use adversarial or unexpected inputs to test resilience, while AFL++ [46] and OSS-Fuzz [47] are also state of the art fuzzing tools examples that extend these methods to software ecosystems. Symbolic execution techniques provide abstract program executions that capture multiple paths simultaneously. Tools like KLEE [48] and Angr [49] demonstrate how symbolic execution can guide fuzzers and developers by producing concrete inputs for hard-to-reach execution paths. All these advances show how AI-driven techniques not only automate traditional testing workflows but also extend testing capabilities to modern domains such as deep learning systems.

2.8 Data quality assessment, analytics, relevance, balance, integrity maintenance

The Sogeti Quality AI Framework (QAIF) [50] provides clear guidance for the development, testing, and usage of AI solutions that are reliable, ethical, and robust. QAIF is based on the ideas of fairness, openness, and responsibility, and it follows the EU Guidelines for Trustworthy AI. It focuses on three principles of quality in AI engineering: Data Quality, Model Quality and Outcome Quality. The analysis of the current state of the art shows that while significant advancements have been made in AI-enabled code analysis, generation, and testing, several important gaps remain. AI4SWEng is strategically positioned to tackle these issues. The project's main goal is to bridge the identified gaps and advance the field of AI-assisted software engineering through an integrated, end-to-end AI-powered system for the entire software development lifecycle (from requirement analysis and code generation to advanced debugging and security). A concise table which summarizes the tools and techniques which are discussed in this section can be found in Appendix A: State of the art.

2.9 Gaps and limitations of the state-of-the-art

The analysis of the current state of the art confirms that AI is more and more integrated into the software development lifecycle. However, this integration is accompanied by important limitations and gaps that prevent the realization of truly agile, trustworthy, and end-to-end AI-augmented software engineering. To this

end, the requirements in the AI4SWEng project (Section 0) have been designed in such a way to address these gaps and limitations (which are briefly presented next):

- Current tools excel at specific tasks (e.g., code autocompletion or test generation) but lack efficiency working together. A major gap exists for true end-to-end automation, trusted, traceable, and automated flow from natural language, requirement analysis and high-level design to executable code.
- Several tools exist for the generation of pseudocode or the analysis of traces, they often fail to achieve semantic reverse execution. The main limitation is the inability to convert low-level binary code or execution traces back into verified architectural models that preserve security and performance.
- For the existing AI tools, it is almost impossible to underline each hardware architecture. There is a significant need for AI tools specifically engineered for multi-architecture support (e.g., IoT, heterogeneous systems) and capable to assist developers to build energy efficient software.
- A critical strategic gap lies in tools designed to ensure regulatory compliance and trustworthiness. Current solutions do not adequately provide the necessary traceability, assurance, and certification artifacts required by emerging European legislation, such as the AI Act, especially concerning bias detection, explainability, and auditing during development.
- The current state of the art analysis demonstrates a diverse set of tools for code quality. However, there is an insufficient focus on assuring the quality, bias mitigation, and integrity of the data used to train the AI engineering tools themselves. Without robust data QA, the generated code and tests inherit hidden risks and vulnerabilities.

These gaps have been taken into consideration to ensure that the proposed requirements within the scope of the AI-SysDev platform and the TAT can enable the AI4SWEng project to deliver targeted, high-impact solutions to the most emerging challenges in modern software engineering.

3. Requirements and user needs

3.1 Types of users

3.1.1 Developers or practitioners

The group of developers or practitioners refers to the direct users of the AI4SWEng, including individual developers, software engineers, and DevOps/MLOps professionals who work in agile, multi-architecture environments. Their main needs include: (i) efficient code analysis and code generation, (ii) automatic debugging, (iii) reverse execution, (iv) cross-compilation, (v) design-by-contract support, and (vi) energy-efficient development pipelines. AI4SWEng will deal with these needs to achieve: (i) reduced time-to-market, (ii) higher productivity, and (iii) improved software quality. These outcomes align with the project's KPIs and its mission to transform the existing software engineering ecosystem.

3.1.2 Integrators or technology providers

This group consists of professionals who use enterprise-level technology, including companies that develop and integrate complex systems, such as, cloud platforms, IoT/edge solutions, cyber-physical systems, and network infrastructures. Their primary needs include advanced tools for scalability, reusability, validation, security-by-design, and energy efficiency across the software lifecycle. AI4SWEng faces these needs through deliverables such as the AI-SysDev platform and the TAT. In this way, AI4SWEng will improve integration of AI-powered lifecycle support into enterprise pipelines. At the organization level, the expected benefits include cost reduction, optimized use of resources, and increased resilience in multi-architecture deployments.

3.1.3 Regulators or certification authorities

This group consists of regulators and certification authorities. Their needs extend beyond just functionality. They focus on reliable and transparent software engineering processes supported by AI that are compliant with GDPR [51], the AI Act [52], ENISA guidelines [53] and emerging standards. AI4SWEng aims to address these needs with respect to compliance-by-design mechanisms and social and ethical acceptability (SO5). The users in this group expect benefits such as automatic tracking and lower costs for audits and compliance. In this way, the project ensures that its outputs are aligned with European laws related to trustworthiness and regulatory compliance.

3.1.4 End-user organizations and application stakeholders

This group represents the entities that will benefit from the AI4SWEng outputs. These include businesses that rely on AI-intensive software systems. Their main needs lie on: (i) software engineering tools which are reliable, efficient, secure and flexible, and (ii) tools that produce AI-assisted applications with reduced operational risks and costs. AI4SWEng aims to address all these needs through its focus on resilience, reliability, energy efficiency and compliance-by-design. As a result, end-user organizations can expect higher-quality and more reliable solutions that enhance the resilience of the deployed systems.

3.1.5 Research and training communities

This group includes universities, research and technology centers, and training providers. Their needs are oriented on advanced methods, datasets, benchmarks, and training curricula for AI-supported software engineering. They focus on getting hands-on experience in emerging and AI-oriented domains, such as, prompt engineering, automated design, and testing. AI4SWEng can address these needs through SO6, which focuses on training developers to develop all these skills. The project aims to ensure that its' innovations have an impact on educational society and reduce the shortage of full stack AI software engineers. The expected benefits are related to the transfer of research into practice.

Table 2: Standardized user groups and terminology harmonization for the AI4SWEng project. This table defines the five primary user groups targeted by AI4SWEng and establishes consistent terminology to be used across user analyses, use case descriptions, and requirements specifications.

User Groups	Description	Typical Role in AI4SWEng
Developers or *Practitioners*	Individual professionals engaged in design, coding, debugging, and testing within agile, multi-architecture environments.	Direct end-users of the AI-SysDev and TAT.
Integrators or *Technology Providers*	Companies or consortia, delivering complex systems (e.g., IoT, cloud, CPS) that embed or extend AI4SWEng solutions.	System-level adopters and validators.
Regulators or *Certification Authorities*	National/EU-level organizations overseeing AI, cybersecurity and data compliance (AI Act, GDPR, ENISA).	Oversight, auditing, and compliance validation.
End-User Organizations or *Application Stakeholders*	Enterprises, deploying AI-driven systems in domains such as healthcare, mobility, and energy.	Beneficiaries of improved quality, resilience, and efficiency.

User Groups	Description	Typical Role in AI4SWEng
Research & Training Communities	Universities, Research and Technology Organizations and open-source communities advancing and teaching AI-assisted software engineering.	Contribute datasets, benchmarks and upskilling resources.

Based on these five main groups of users, the AI4SWEng project will identify a complete set of use cases to cover the range and variety of software engineering environments, where the project's innovations will be validated with respect to real-world needs.

3.2 Data preparation activities and resources

A very important aspect of the AI4SWEng project lies in the data preparation activities. This process makes sure that all datasets used to train, validate and test the AI4SWEng suite's KIOs are secure (in terms of privacy), standardized, and in-line with ethical standards. The project's multi-domain nature, which includes multi-domain use cases (e.g. industrial, medical, energy, and cybersecurity), underlines that the data preparation activities shall be carefully designed to ensure interoperability across all KIOs.

3.2.1 UC1 (AI-enhanced software lifecycle optimization)

The data sources in UC1 include: (i) historical software project repositories (e.g., Git, Jira, GitLab issues, documentation), (ii) requirements documents and change requests from past projects, (iii) source code samples for various architectures and languages, and (iv) testing artefacts, such as, unit test scripts and reports. The datasets will be collected from public repositories (Kubernetes templates, Tensorflow templates, GitHub Docs templates). The data preparation activities are described below:

- Removal of personal identifiers and sensitive details (preserve privacy).
- Evaluation of each data item and of the corresponding metadata to ensure full traceability and compliance with GDPR and AI Act principles (makes it auditable).
- Identification and removal of duplicates, incomplete records, or inconsistent entries and replace volatile elements (e.g., timestamps, emails, stack traces) with placeholders (data can be noisy).
- Normalisation of file formats (Markdown, JSON, XML), structure (templates) and standardization of naming conventions for uniform processing (to make the examples consistent so that the models can learn signals).
- Application of NLP preprocessing techniques (tokenisation, sentence segmentation, lemmatisation, stop-word removal) to convert unstructured text (e.g., issue descriptions, requirements) into a structured, machine-readable format (tokenisation, sentence segmentation and normalisation improve the model's ability to parse requirements and issue descriptions).
- Extraction of structural representations such as Abstract Syntax Trees (ASTs) from code files, tag functions or methods, and link them to the corresponding issues or requirements (to convert raw text into machine-understandable structure and enable explicit links between code blocks and issues/requirements).
- Application of class imbalance handling methods between issue types (bugs, features, refactors) by generating synthetic examples using controlled data augmentation (to allow the model to learn under-represented cases).
- Aggregation of curated and annotated datasets into a unified schema linking text, code, and metadata (consistent schema).
- Application of automated checks for consistency, completeness, and bias (to ensure the dataset will produce trustworthy outputs).

3.2.2 UC2 (Medical reporting and document automation)

The existing/planned datasets in UC2 include annotated patient monitoring records, medication history, text-based record data, DICOM radiology images and reports from the following sources: PhysioNet, BIMCV-COVID19, NIH Imaging Data Commons, and ner-turkish-radiology. The involved AI models include classification models (e.g., predicting AF recurrence, risk stratification), potentially time-series models for device recording analysis, ML frameworks (scikit-learn, XGBoost, PyTorch, LSTM, RNN), task-specific models trained on collected datasets, fine-tuned LLMs, specialized models for compliance (HIPAA, GDPR, EU AI Act, KVKK) and terminology consistency. The tools involved in UC2 are the following: (i) the data preparation and validation toolbox, (ii) the Neo4J for graph database generation, and (iii) the Model Unit Tester (MUT) for data validation through ML/DL test cases. The infrastructure aims to ensure secure institutional storage for sensitive clinical data, multi-level cybersecurity modules, access control, audit logs, and monitoring and logging tools to guarantee reliability and auditability. The data preparation activities in UC2 are described below:

- Extraction of relevant patient records, including demographic (age, sex, height, weight, BMI) and clinical fields (indication, diagnosis, imaging metadata, vitals, medication history).
- Standardisation and curation of categorical fields (e.g., metadata, labels, medication names) and encode them appropriately for ML/DL models.
- Validation of time/date fields for coherence and consistent formats.
- Detection of missing values in both clinical and device-related columns, using domain-appropriate strategies.
- Normalisation of numerical variables (height, weight, BMI, echo measurements) for model compatibility.
- Generation of derived features if needed (e.g., patient-dataset linkage, risk factor flags based on comorbidities).
- Divide the dataset into training, validation, and test sets, maintaining patient-level separation to avoid data leakage.

3.2.3 UC3 (Large language models for advanced software generation for FPGA)

The data preparation activities not only extend those of UC1 but also supplement it with unstructured data, including PDF manuals and execution logs. These documents and outputs are processed and segmented through a Retrieval-Augmented Generation (RAG) pipeline, while the unstructured runtime data (e.g. output streams) are cleaned and standardized through the Data Preparation Tool. The Data Validator plays a central role here in order to: (i) verify the consistency and correctness of the inputs and outputs across toolchain components, and (ii) ensure full traceability and reliability throughout the FPGA development workflow.

3.2.4 UC4 (AI-driven battery management for electric vehicles)

The data preparation and validation toolbox will manage all preprocessing and quality assurance steps in UC4 prior to AI model training and deployment. An Exploratory Data Analysis (EDA) will be applied by the Data Analyser to examine dataset distributions, correlations, and feature relationships. This analysis will also identify and tag key features, define variable bounds, and detect missing or inconsistent values. Metadata will be generated for semantic information to support Knowledge Graph (KG) construction, compensating for incomplete dataset metadata. Those will be used to establish the preparation pipeline according to the target task description. The datasets involved in UC4 include heterogeneous EV sensor datasets such as Battery voltage, current, temperature, charge/discharge cycles (e.g. Charger Data [54], Dacia/Nissan Data [55]), supplementary unstructured data, maintenance logs, environmental conditions, KG updates for contextual enrichment through LLM processing, Weather API [56], Maintenance Logs [57]. The data preparation activities in UC4 are described below:

- Removal or imputation of missing values and elimination of outliers.
- Normalization and scaling of sensor data to ensure consistency across devices and sampling rates.
- Application of peak detection and noise filtering methods for time-series sensor data.
- Generation of variables to capture battery health indicators (e.g., voltage fluctuation trends, temperature gradients).
- Feature engineering for graph representation and model interpretability.
- Division of the dataset into train/test/validation splits for temporal and contextual representativeness in federated learning environments.

The AI models in UC4 will include: (i) GNNs for the relational modeling of sensor data, (ii) LSTMs, RNNs, XGBoost, Transformers for sequence prediction and anomaly detection, (iii) Autoencoders, Random Forest for fault detection and performance optimization, and (iv) Pruned TinyML models for deployment on edge devices. The tools and frameworks in UC4 will include: (i) the TensorFlow Lite for TinyML model development and deployment, (ii) the TensorFlow Federated for privacy-preserving collaboration, (iii) the Neo4J graph database for the construction of knowledge graphs, and (iv) the data analyser and the automated data preparation/validation pipeline. As far as the infrastructure is concerned, it is required to support: (i) TinyML capabilities for local inference, (ii) secure federated servers for model aggregation, and (iii) collaborative AI environments integrating GNNs, LLMs with federated learning components.

3.2.5 UC5 (AI-powered GIS integration for smart energy management)

The data in UC5 include large-scale geospatial datasets, such as raster, vector, and temporal data. Data is provided in specialized formats such as GeoPackage, Parquet, GeoJSON, and Cloud-Optimized GeoTIFF, organized within STAC catalogs. These are integrated through ETL processes which are orchestrated by the Data Warehouse Agent. The Data Validator will ensure policies such as single-coordinate reference systems, “NoData” and temporal align, while the automated data preparation tool will introduce gap-filling mechanisms and cross-format synchronisation. Working with consortium partners makes sure that cloud storage and retrieval mechanisms are as scalable, cost-effective, and compliant with open geospatial standards as possible.

3.3 Ecosystem actor behavior diagrams

The diagrams in this section provide a high-level representation of how different ecosystem actors interact with the AI4SWEng suite and its KIOs. They capture the main functionalities expected from the system and illustrate the relationships between actors and the services offered.

3.3.1 AI-driven software development automation

Purpose	Utilize AI-powered technologies to automate software development processes.
Description	This module assists software providers towards the incorporation of AI into the software development process by automating code creation, testing, and debugging, AI-based solutions simplify software deployment, increase productivity, and lower mistakes.
Ecosystem actors	Software Developers, AI Engineers, DevOps Teams.
Flow/Usage	AI tools automate testing procedures, evaluate code, and recommend improvements. AI-generated outputs are examined and improved by developers, guaranteeing software deployment effectiveness and quality.

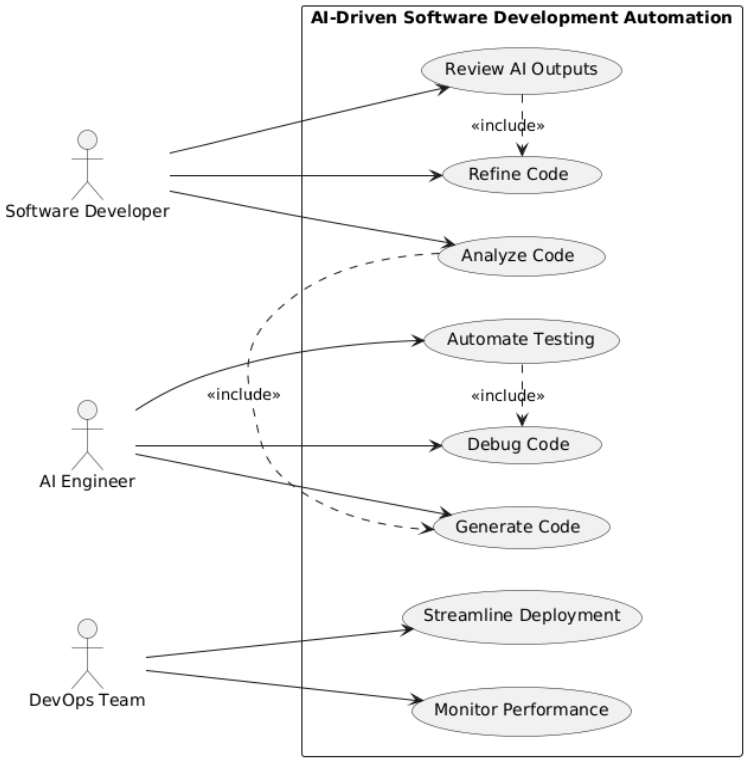

Figure 1: AI-driven software development automation - Workflow showing Software Developers, AI Engineers, and DevOps Teams using AI tools for code generation, testing, debugging and deployment.

3.3.2 AI-based software integration

Purpose	Ensure that AI-driven software solutions are seamlessly incorporated into SME operations.
Description	AI software vendors may provide scalable and flexible AI solutions that can be integrated into SMEs' current systems thanks to the integration module. This comprises middleware, cloud-based services, and APIs to support automation, data-driven decision-making, and interoperability.
Ecosystem actors	Software Developers, AI Solution Providers, SME IT teams.
Flow/Usage	AI models are created and tested by developers, and are also connected with corporate software and iteratively release updates. The IT departments of SMEs set up and manage these connections with ongoing assistance from technology suppliers.

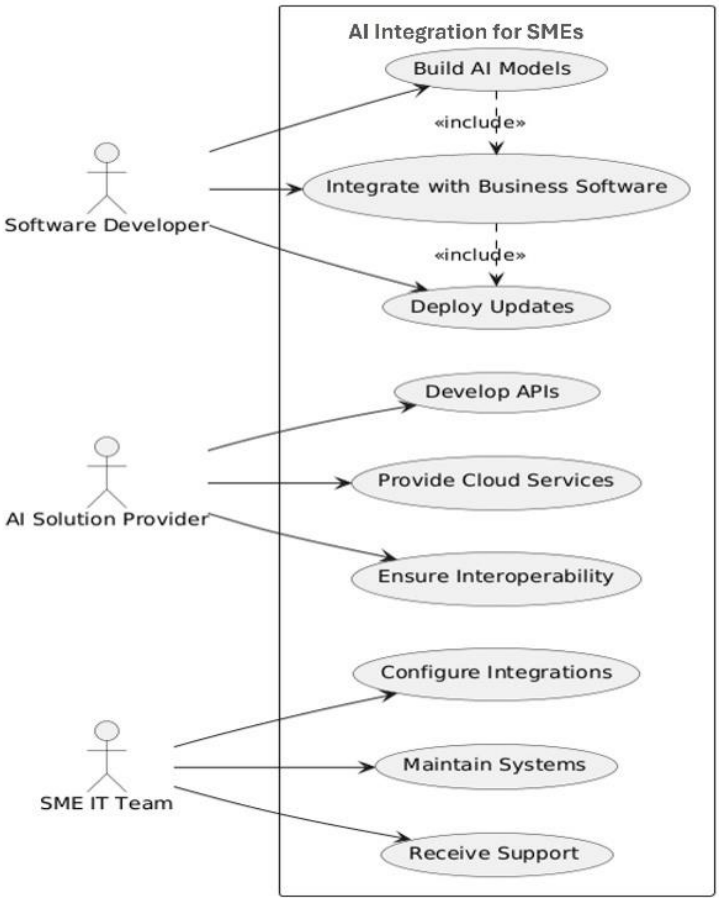

Figure 2: AI-based software integration - Workflow showing collaboration between Software Developers, AI Solution Providers and SME IT Teams to integrate AI models into business software with ongoing support.

3.3.3 AI-powered cybersecurity analysis

Purpose	Strengthen cybersecurity protocols for software solutions powered by AI.
Description	For SMEs, this system offers AI-based threat detection, privacy protections, and compliance automation. It assists software developers to make sure that their AI solutions adhere to security regulations such as ENISA and GDPR.
Ecosystem actors	Cybersecurity Experts, AI Developers, SMEs' IT Security Teams.
Flow/Usage	AI models keep an eye out for threats, cybersecurity specialists create security rules, and real-time notifications help SMEs' IT teams reduce risks while maintaining regulatory compliance.

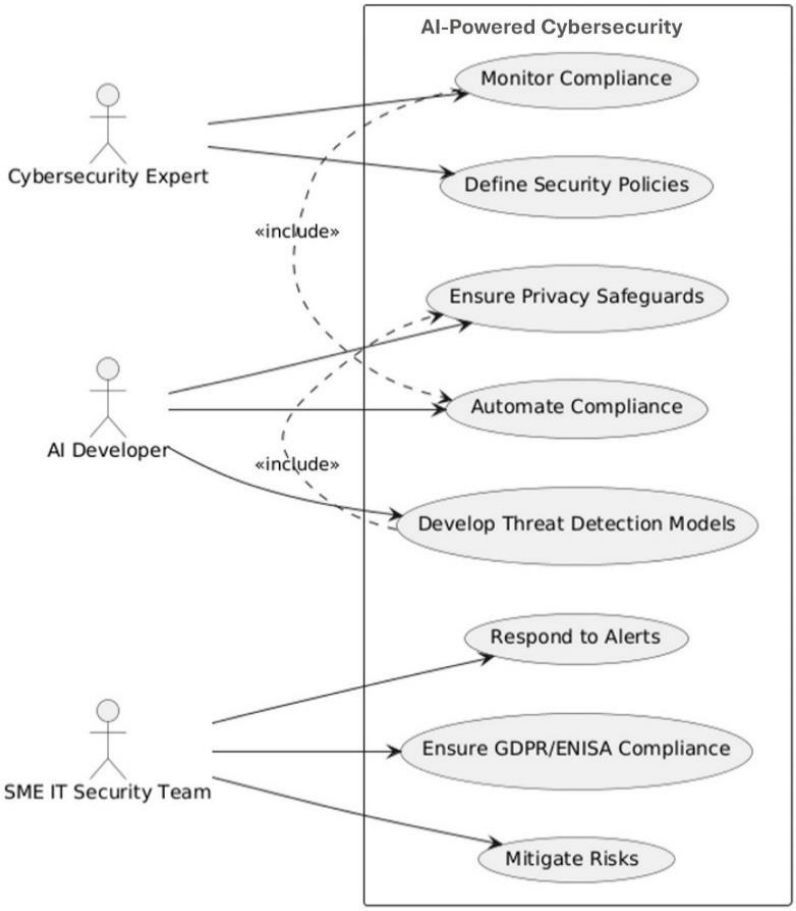

Figure 3: AI-powered cybersecurity analysis - Workflow showing collaboration to strengthen cybersecurity, define policies, ensure privacy, automate compliance, detect threats, respond to alerts, ensure GDPR/ENISA compliance and mitigate risks.

3.3.4 Human-centric AI-powered interfaces

Purpose	Make sure that AI-powered interfaces are ethical and easy to use.
Description	This toolkit provides AI developers with best practices, templates, and guidelines for creating user interfaces that are clear, easy to understand, and driven by AI. Its main goal is to make AI understandable to non-technical people as well.
Ecosystem actors	UX/UI Designers, AI Developers, Market Researchers.
Flow/Usage	Usability testing is performed, interfaces are improved based on input from actual users, and developers include interactive features and AI-driven recommendations into their apps.

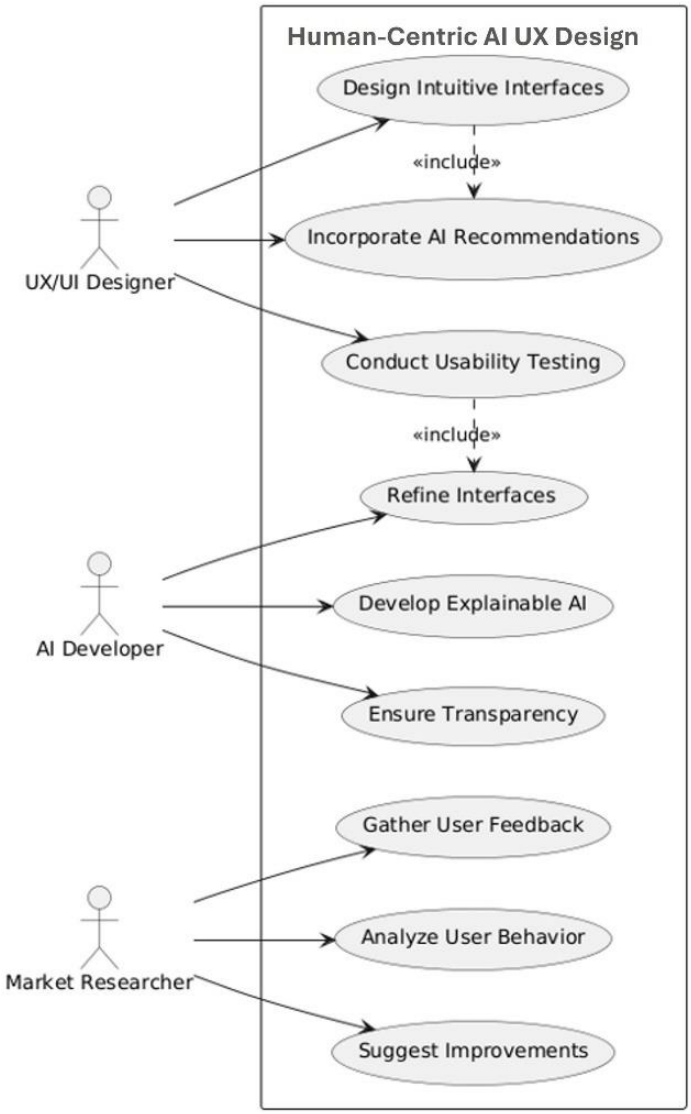

Figure 4: Human-centric AI-powered interfaces - Workflow showing UX/UI Designers, AI Developers, and Market Researchers designing intuitive AI interfaces with testing and improvements.

3.3.5 Data security and privacy compliance

Purpose	Improve privacy compliance and data security in AI-powered solutions.
Description	A regulatory framework that assists in making sure AI systems follow data protection regulations and cybersecurity best practices, avoiding security lapses and illegal data use.
Ecosystem actors	Cybersecurity Agencies, Data Protection Authorities, Compliance Officers.
Flow/Usage	The platform monitors weaknesses in AI systems, confirms that security guidelines are being followed, and makes suggestions to improve data security and reduce risks.

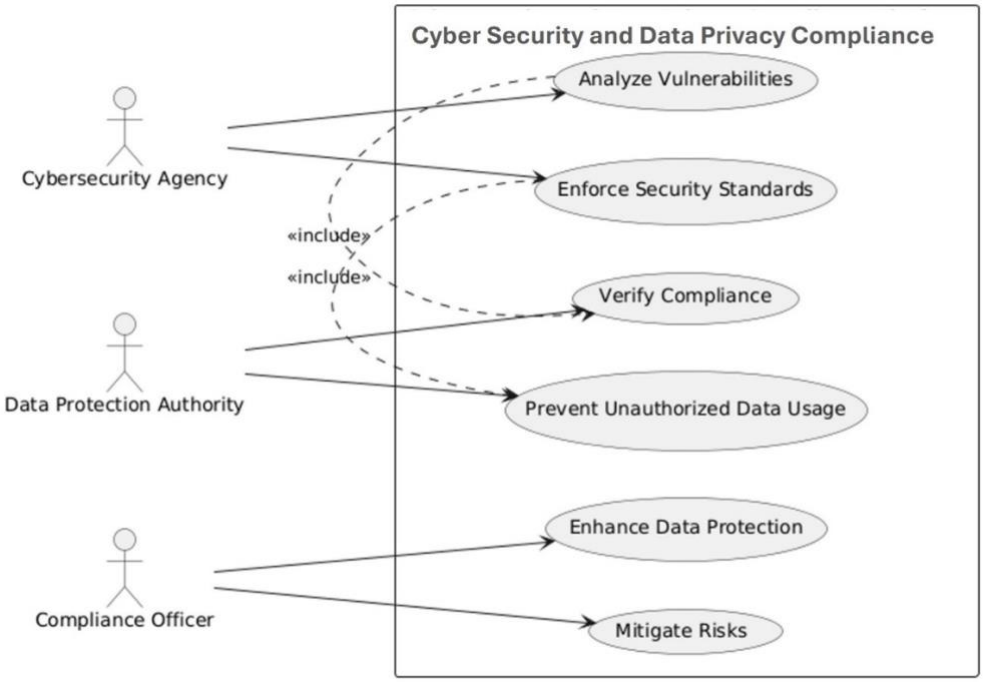

Figure 5: Data security and privacy compliance - Workflow showing collaboration to analyze vulnerabilities, enforce standards, verify compliance, prevent data misuse, enhance protection and mitigate risks.

3.3.6 AI-powered regulatory compliance

Purpose	Evaluate that software solutions powered by AI adhere to legal and regulatory requirements.
Description	This framework aids regulatory agencies in monitoring adherence to the AI Act, GDPR, ENISA security requirements, and other pertinent laws. It consists of risk evaluations, computerized audits, and procedures for enforcing policies.
Ecosystem actors	Regulatory Agencies, Compliance Officers, Policymakers.
Flow/Usage	To guarantee compliance with legal requirements, regulatory authorities employ AI-powered technologies to evaluate software compliance, provide reports, and offer advice to AI developers and companies.

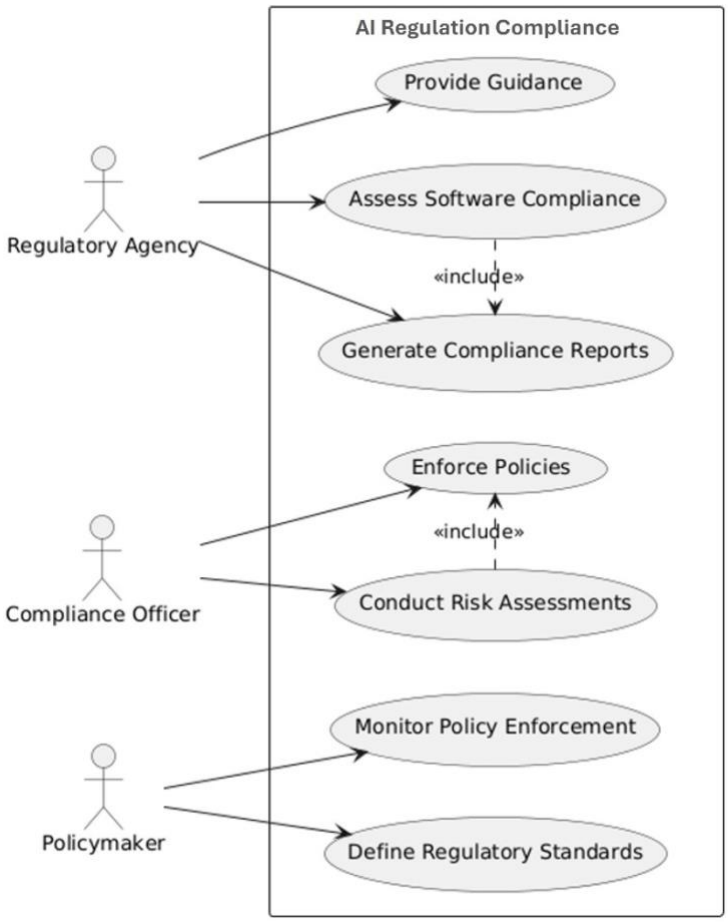

Figure 6: AI-powered regulatory compliance - Workflow showing collaboration to provide guidance, assess compliance, generate reports, enforce policies, assess risks, monitor enforcement and define standards.

3.3.7 Ethics and fairness assessment

Purpose	Ensure that AI solutions avoid biases and adhere to ethical norms.
Description	This tool assesses AI models for ethical, transparent, and equitable issues. It assists regulatory agencies in enforcing rules around bias detection and appropriate AI use.
Ecosystem actors	Regulatory bodies, Ethics Committees, AI Researchers.
Flow/Usage	To guarantee the ethical deployment of AI, regulators utilize the tool to evaluate AI models for bias and fairness, produce reports, and suggest enhancements to technology suppliers.

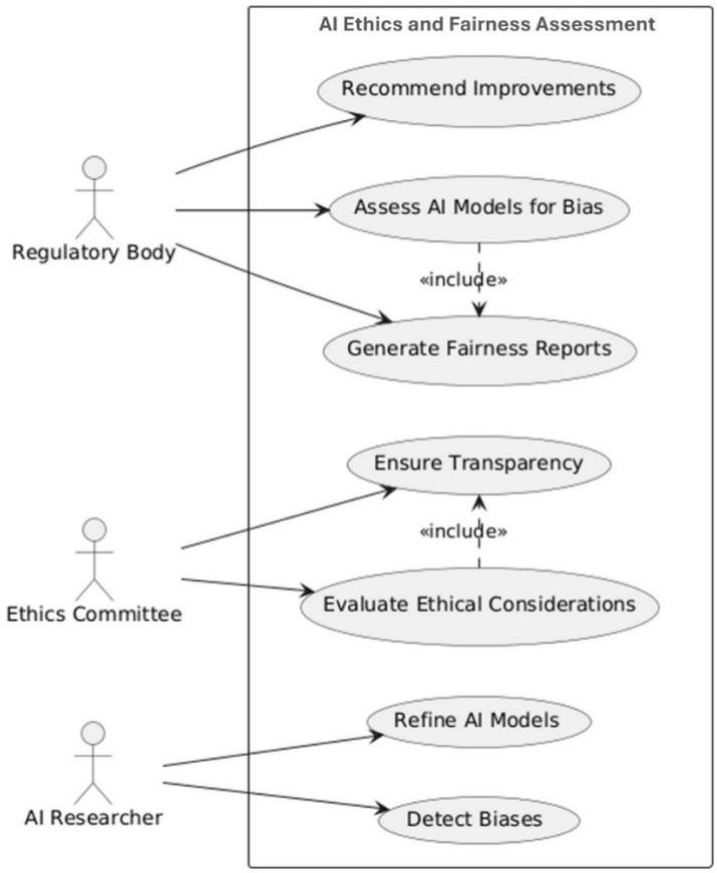

Figure 7: Ethics and fairness assessment - Workflow showing collaboration to assess bias, ensure transparency, refine models and recommend improvements.

3.3.8 AI-driven monitoring of changing security guidelines

Purpose	Monitor and evaluate changing security guidelines and legislation pertaining to AI.
Description	An AI-powered system that continually reviews and revises regulatory frameworks, offering up-to-date information on new laws and regulations that impact AI technology.
Ecosystem actors	Government Agencies, Legal Analysts, Policymakers.
Flow/Usage	To ensure that sure AI technologies continue to meet new standards, the system tracks regulatory updates globally, classifies legal changes, and alerts pertinent parties.

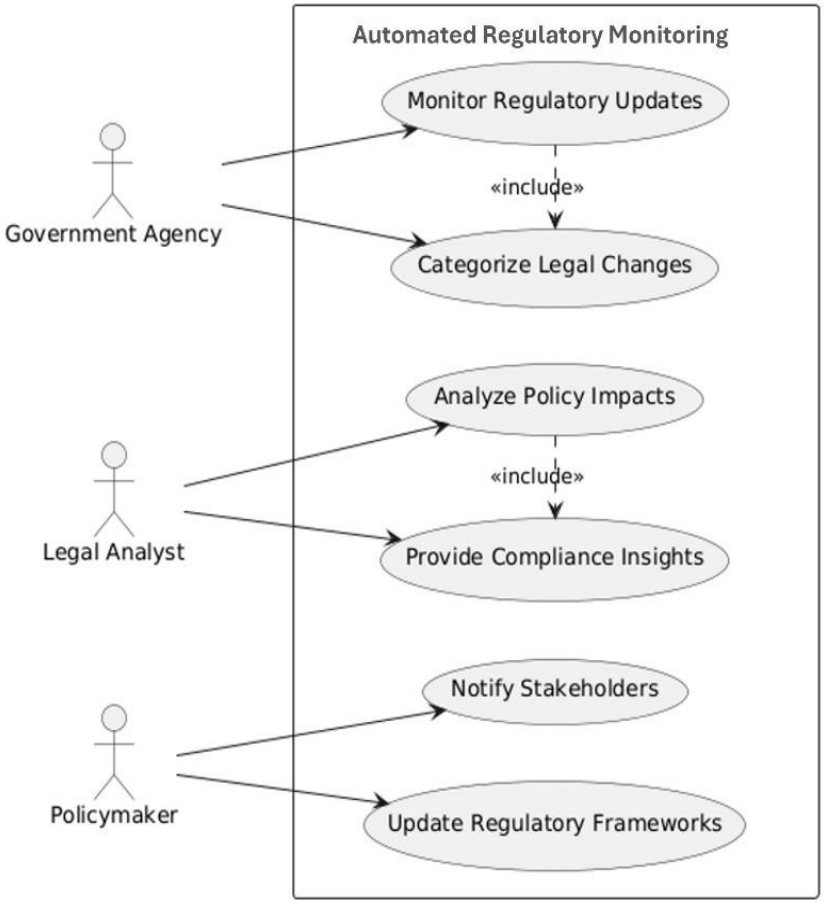

Figure 8: AI-driven monitoring of changing security guidelines - Workflow showing collaboration to monitor updates, categorize changes, analyze impacts, provide insights, notify stakeholders and update frameworks.

3.3.9 Automated AI training and upskilling

Purpose	Assist workers and SMEs in adjusting to AI-driven procedures.
Description	An AI-powered learning platform that guarantees workers acquire pertinent skills by offering interactive training on automation, digital transformation, and AI technologies.
Ecosystem actors	Employees, SME/Business Professionals, HR Departments, AI System Providers.
Flow/Usage	Workers have access to AI-powered training programs and role-specific simulations. Technology gives certificates for AI competency, records progress, and creates customized learning routes.

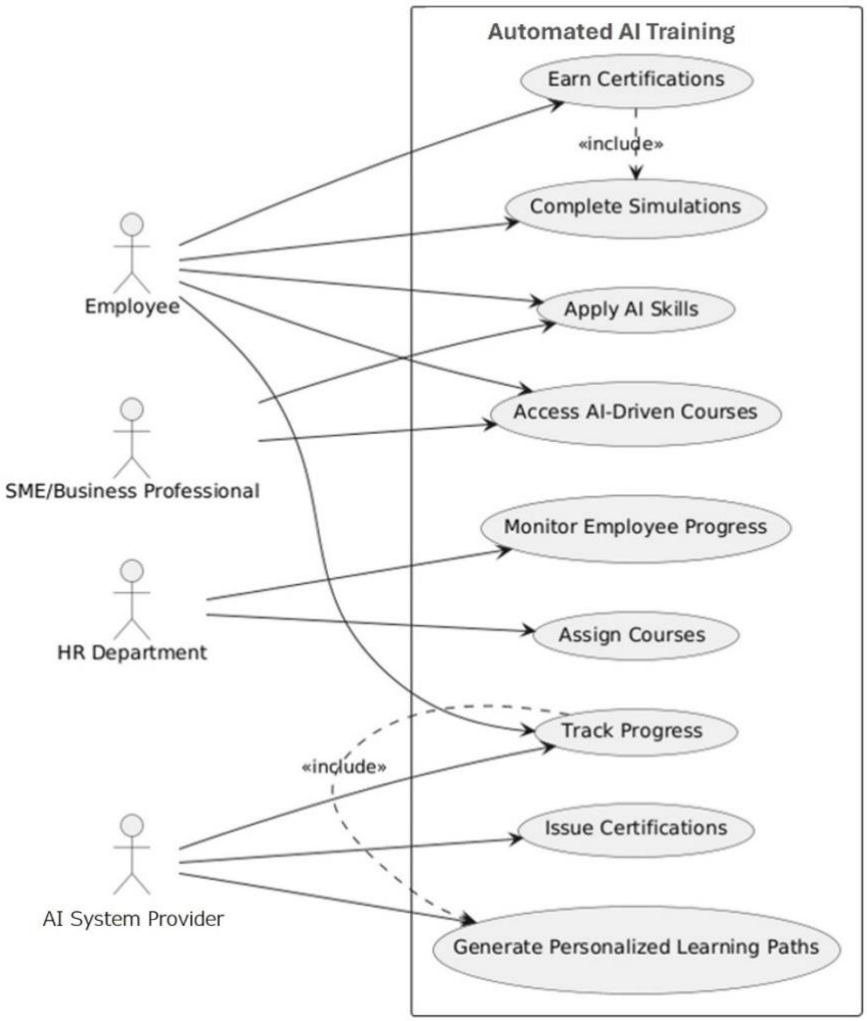

Figure 9: Automated AI Training - Workflow showing Employees accessing courses, HR tracking progress and AI systems generating personalized learning paths.

3.3.10 AI-assisted decision support

Purpose	Leverage AI insights to assist SMEs and enterprises in making data-driven choices.
Description	This system processes corporate data, finds patterns, and makes suggestions based on AI-powered analytics. It assists SMEs in increasing productivity, cutting expenses, and optimizing operations.
Ecosystem actors	SME Owners, Business Analysts, Managers.
Flow/Usage	Once business data is entered by users, the AI system looks for patterns and produces insights. Recommendations help managers make better decisions in areas like marketing, client interaction, and inventory management.

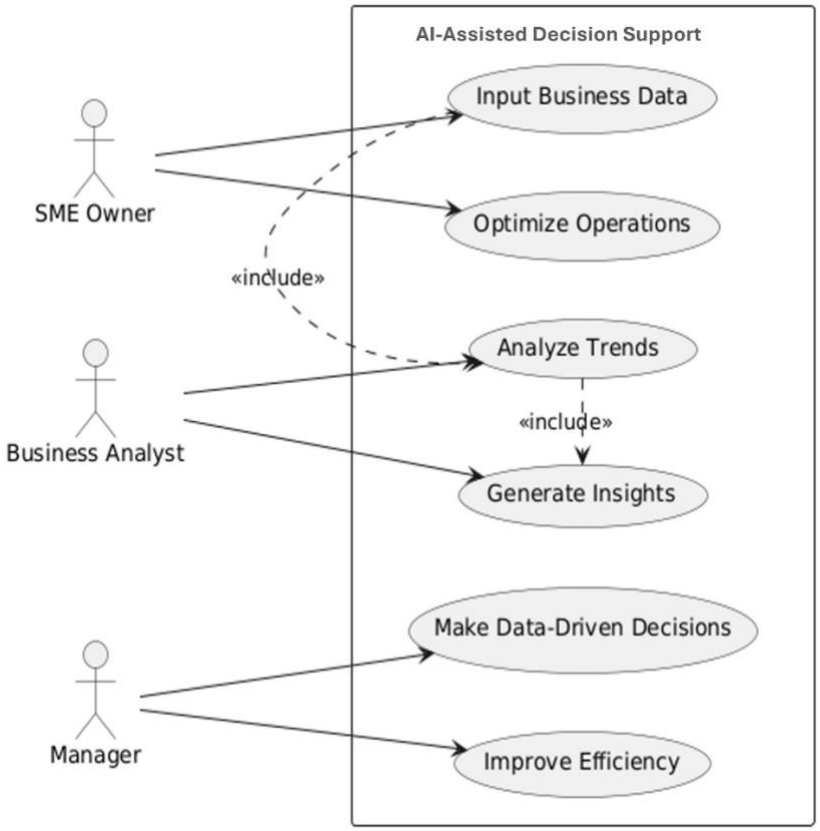

Figure 10: AI-assisted decision support - Workflow showing SME Owners, Business Analysts, and Managers providing data, analyzing trends, generating insights, making data-driven decisions, optimizing operations and improving efficiency.

3.4 High-level needs of the AI4SWEng tools and components

This section describes the technical specifications of the AI4SWEng KIOs. Each KIO represents a core building block of the AI4SWEng suite, addressing specific functionalities across the software engineering lifecycle. The specifications cover inputs, outputs, hardware and UI requirements, functional and non-functional requirements, and targeted stakeholders. Together, the KIOs form a coherent ecosystem, starting from data preparation (KIO5–KIO6), requirements formalization and architecture design (KIO3–KIO4), and AI-assisted development and testing (KIO1-KIO2, KIO7, KIO11–KIO12), to cross-cutting concerns such as compliance (KIO9), energy efficiency (KIO10), and skills development (KIO13). The specifications ensure that each component is both self-contained and interoperable within the DevOps-based framework envisioned by AI4SWEng. The following subsections present the high-level needs elicited for each KIO, including its short description, Lead partner, expected TRL improvement, high-level Inputs, provisioned outputs, hardware needs, UI needs, information flow, functional needs, non-functional needs, and targeted stakeholders. This level of detail is essential to harmonize design and implementation activities across the KIOs leaders/contributors and to ensure traceability between user needs, functional requirements and technical solutions in further stages of requirement engineering (see D2.6 for the full list of requirements).

Table 3: High-level needs elicited for KIO1.

Information	Description
AI4SWEng tool/component	AI4SWEng AI Engineering Suite (KIO1)
Short description	A unified platform integrating all AI-powered tools within a DevOps-based software development lifecycle.
Lead partner	UREAD
Expected TRL Improvement	3 to 5
High-level Inputs	1. Source code, 2. Configuration files, 3. Deployment parameters
Provisioned Outputs	1. Generated code, 2. Deployment artifacts, 3. Execution reports
Hardware needs	1. Container orchestration infrastructure (Kubernetes/Docker Swarm) 2. Compute resources: (16+ CPU cores, 64GB+ RAM for AI processing) 3. High-speed storage (SSD) 4. GPU acceleration
UI needs	1. Web-based GUI, 2. Responsive design, 3. Real-time feedback
Information flow	User Requirements → LLM Engine → Code Specifications → LLM Engine → Generated Code → Testing Pipeline → LLM Engine → Code Refinement → CI/CD Pipeline → Deployment Artifacts → Production Environment → Monitoring Data → Performance Analytics → LLM Engine → Optimization Recommendations → [Feedback Loop] → User Requirements Refinement
Functional needs	1. Unified KIO Integration Platform 2. Front-End GUI for Lifecycle Interaction 3. DevOps Workflow Orchestration 4. Containerized Service Deployment 5. Horizontal LLM Integration

Non-Functional needs	1. Modular Architecture Design 2. Flexibility in Workflow Configuration 3. Scalability for Integrated Services 4. Interoperability with External Systems 5. Security in Integration Layer 6. System Reliability 7. GDPR and Ethics Compliance
Targeted stakeholders	1. Development teams, 2. DevOps engineers, 3. Project managers, 4. Quality Assurance teams, 5. Researchers, 6. Early-stage startups

Table 4: High-level needs elicited for KIO2.

Information	Description
AI4SWEng tool/component	Bug Locate & Fix: Reverse Execution & Dynamic Slicing Component (KIO2)
Short description	Component that reconstructs execution flows, identifies fault-inducing code segments, and suggests corrective actions with explainable insights.
Lead partner	BITNET
Expected TRL Improvement	2 to 4, 5
High-level Inputs	1. Source code 2. System configuration information 3. Execution traces 4. Runtime logs
Provisioned Outputs	1. Identified bugs 2. Suggested fixes 3. Explanations to the root cause
Hardware needs	Computation environment for LLM: 1. memory 2. high-end GPU accelerator
UI needs	1. Input area for questions 2. Execution buttons for each test (or an all tests button) 3. Code blocks for displaying identified bugs 4. Text boxes for explanations 5. Search/filter UI for trace logs 6. In-line ChatBot message
Information flow	Runtime logs + System Info + Code base -> Reverse execution engine -> Dynamic slicing analyzer -> AI fault predictor -> Fix suggestion generation -> reports -> documentation
Functional needs	1. Source code language support (One language for PoC) 2. Trace Capture & Replay to generate reversible trace data 3. AI-assisted replay simulators that reconstruct previous states from logged diffs 4. Dynamic slicing to identify which statements and data dependencies contributed to a given program behaviour 5. AI Fault Localization to train from replayed traces and slice context

	6. The system shall employ a specialised Large Language Model (LLM) fine-tuned for bug detection to automatically identify, classify, and prioritise software faults from source code and execution trace 7. The system shall automatically generate, rank, and validate fix suggestions for detected faults using AI/LLM-based reasoning, ensuring each proposed patch compiles successfully, preserves intended functionality, and provides an explainable rationale for developer review. 8. Thread-aware trace capture and replay simulation
Non-Functional needs	1. Performance: Common operations in the Reverse Execution & Dynamic Slicing Component shall complete in ≤ 5 seconds under normal load conditions with desired hardware with proper GPU 2. Quality: minimum bug-localization accuracy of 95% compared to verified ground-truth datasets under normal load conditions with desired hardware with proper GPU 3. Modularity: Updatable structure : Modularity – Independently Updatable Component Structure (no cross-module impact), nsuring successful update and integration tests complete within 8 hours without regression failures.
Targeted stakeholders	1. Development teams, 2. Early-stage startups

Table 5: High-level needs elicited for KIO3.

Information	Description
AI4SWEng tool/component	Natural Language Processing (NLP) to Formal Requirements Component (KIO3)
Short description	Converts natural language requirements into formalized user stories. Uses LLMs to generate structured requirements for software development.
Lead partner	IDE (T3.2)
Expected TRL Improvement	3 to 5
High-level Inputs	Natural language requirement by stakeholder
Provisioned Outputs	Formal requirement
Hardware needs	Computation environment for LLM
UI needs	1. Input area for natural language requirement and acceptance criteria 2. Output panel for formal requirement 3. Edit tools for manual adjustments 4. Save/cancel buttons 5. Traceability viewer for natural language-requirement mapping
Information flow	User input -> LLM engine -> user story -> LLM engine -> formal requirement
Functional needs	1. Parse and extract key intent and actors from natural language 2. Generate formalized user stories using a trained LLM 3. Attach acceptance criteria to each story 4. Allow post-generation editing by users 5. Detect and highlight ambiguous or vague input

	6. Automated wireframe generation 7. Export functionality for project artifacts
Non-Functional needs	1. Performance: fast response to request 2. Scalability: handle concurrent requests 3. Accuracy: no ambiguity or contradiction 4. Security: securely stored data 5. Artifact versioning support 6. Graceful error detection and recovery 7. Usability: intuitive UI
Targeted stakeholders	product manager

Table 6: High-level needs elicited for KIO4.

Information	Description
AI4SWEng tool/component	Formal Requirements-to-Code Architectural Planning Component (KIO4)
Short description	Translates formal requirements into software architecture models. Uses LLM-powered architecture suggestion mechanisms for design planning.
Lead partner	IDE (T3.2)
Expected TRL Improvement	2 to 5
High-level Inputs	Formal requirement
Provisioned Outputs	
Hardware needs	Computation environment for LLM
UI needs	1. Formal requirements upload/selection 2. Architecture panel for interactive view for editing and exporting diagrams. 3. Retry button to show alternative designs and trade-offs 4. Traceability viewer for requirement-component mapping
Information flow	Formal requirement -> LLM engine -> architecture planning -> diagram generator
Functional needs	1. Parse formal requirements and identify architectural implications 2. Suggest high-level architecture 3. Generate UML diagrams 4. Provide traceability matrix between requirements and design components 5. Allow user interaction to accept, modify, or reject suggestions 6. Export project scaffolding/code skeletons based on architecture 7. Architectural Rationale Documentation and Attachment 8. Formal requirements ingestion 9. Architectural component export for tool integration
Non-Functional needs	1. Performance: fast response to request 2. Scalability: batch processing 3. Accuracy: high relevance of suggested architecture 4. Security: securely stored data 5. Usability: intuitive interface 6. Architectural component versioning support 7. Graceful error detection and recovery 8. Compliance: support for compatible output formats
Targeted stakeholders	developer

Table 7: High-level needs elicited for KIO5.

Information	Description
AI4SWEng tool/component	Synthetic Data Generation Tool (KIO5)
Short description	AI-driven data synthesis using Generative Models (GANs, VAEs, LLM). Supports privacy-preserving synthetic data generation for training AI models.
Lead partner	CEA (T3.1)
Expected TRL Improvement	3 to 5
High-level Inputs	1. Several Mo/Go/To of real data samples (.txt, .json, .csv, SQL, NoSQL, .pdf, .docx) preprocessed; 2. Formal requirements for generation (model hyperparameters, semantic/architecture constraints, data volume, data deviation
Provisioned Outputs	1. Generated data matching input requirements 2. Generation metrics (E.G. quality, time) 3. Generation report
Hardware needs	1. Computation environment for LLM; 2. Between 64G and 512G of memory
UI needs	1. Generation state 2. Metrics evaluation dashboard 3. Generated data preview with actions to delete row, feature, etc 4. Logs and alerts display 5. Export button 6. Download report button
Information flow	User input -> GAN/LLM -> data validation -> report generation
Functional needs	1. Generate compliant synthetic traffic at flow level 2. Generate synthetic traffic representing attacks at the flow level 3. Merge GIS layers from different sources to create a new synthetic map representing energy consumption 4. Generate synthetic GIS layers to create new synthetic maps representing non-existing energy consumption scenarios 5. Generate a master medical datasets by merging together datasets by creating synthetic link between the entries 6. Propose validation methods for each generation task 7. Report generation for generation tasks
Non-Functional needs	1. Performance: fast Generation, low latency for data interactions 2. Scalability: handle wide inputs and generate large synthetic datasets / multithreading for multi-GPU trainings 3. Accuracy: synthetic data matches input data 4. Security: data protection, injection protections 5. Usability: intuitive UI
Targeted stakeholders	- data scientist

Table 8: High-level needs elicited for KIO6.

Information	Description
AI4SWEng tool/component	Data Validation-Preparation Tool (KIO6)

Short description	Ensures data integrity, cleaning, balancing, and validation. Uses schema-based validation and AI-driven quality analysis.
Lead partner	CEA (T3.1)
Expected TRL Improvement	3 to 5
High-level Inputs	- Textual dataset (unstructured data): single-batch and inter-batch; - Structured dataset : well-defined format used in ML applications, single-batch and inter-batch.
Provisioned Outputs	- Ready to use data (clean, balanced & formated datasets); - Data validation report: comparison between ingested data and data properties schema, recommendations for updating the schema or correcting data, compliance metrics for single and inter batch validation; - Statistical properties (descriptive statistics, correlation mapping, visualisation); - Training code errors; - Data quality report issued by Data Quality Wrapper (DQW); - Suitable data for LLM training such as question/answer peers, annoted documents, etc. - Datasets for cybersecurity (compliant & attack traffics)
Hardware needs	- Computation environment for LLM; - NoSQL database for unstructured datasets and reports; - Tools for orchestration of the data pipeline (e.g Apache Airflow).
UI needs	- Data quality visualisation with graphs and KPI; - Data properties schema visualisation with an edit mode, should display schema update recommendations; - Error displaying with current state (resolved, in progress, ...); - Data statistics properties visualisation; - User interface to define the preprocessing steps and preview of preprocessed data; - Preview of raw data and transformed data for LLM use; - Report section with download functionnality.
Information flow	- Ingest data -> Data Quality Wrapper -> Data Analyser -> Model Unit Tester - Ingest data -> Data Quality Wrapper -> Data Validator -> Model Unit Tester - Ingest data (text or strcutured) -> Data preparation tool - Ingest query (API specifications or scenario) -> dataset generation for cybersecurity tool
Functional needs	1. Comparison between input data and properties schema 2. Automated data preprocessing 3. Report generation for data validation, statistical properties, code errors, agregate data quality 4. Data preparation for LLM 5. User Interface to display the tool outputs, the different steps' details and to edit the data schema
Non-Functional needs	1. Performance: optimised computation of statistic properties, fast validation and preprocessing of data 2. Scalability: must support large number of data and schema 3. Reliability: error detection, analysis and validation procedures and reports must be accurate 4. Sustainability: code must be documented and easy to update/debbug

	5. Transparency: the user must be able to retrieve logs and observe the different steps of the preparation and validation of data 6. Ergonomic/user-friendly: UI must be intuitive and reports easily readable 7. Security: data must be securely stored
Targeted stakeholders	- data scientists - developpers - cybersecurity experts

Table 9: High-level needs elicited for KIO7.

Information	Description
AI4SWEng tool/component	AI-SysDev Tool incorporating LLM Fine Tuning and Inferencing (KIO7)
Short description	Core AI-driven software development tool, integrating: LLM Fine-Tuning for task-specific adaptations. Code generation, debugging, testing, and self-optimization. Uses Retrieval-Augmented Generation (RAG) to enhance AI-based software development.
Lead partner	IDE (T3.2)
Expected TRL Improvement	3 to 5
High-level Inputs	-Fine-tuning datasets -Codebase -Documentation -Natural language requirements
Provisioned Outputs	-Source code - Tests - Debugging logs and fixes - Fine-tuned model - Optimisation report
Hardware needs	Computation environment for LLM
UI needs	- Multi-pane interface with prompt panel, code editor, test view and console output - Model controls to select fine-tuned model and view RAG status - Debugger view for diagnostics and suggestions - Test harness panel for auto-generated test suggestions, run controls and test coverage reports - Fine-Tuning dashboard to upload dataset, monitor training and evaluate performance
Information flow	- User dataset -> dataset validator -> LLM train -> LLM evaluation -> save model - Instruction/requirement -> retrieve relevant files -> LLM engine -> source code -> code analysis -> error explanation + fix -> generate tests -> run tests - Source code -> code analysis -> error explanation + fix -> generate tests -> run tests
Functional needs	1. Accept prompts and generate code 2. Retrieve relevant code patterns or documentation (RAG) 3. Fine-tune LLMs 4. Debug code and return annotated diagnostics and fixes 5. Auto-generate and execute test cases 6. Suggest performance or security improvements

	7. Support multiple programming languages and frameworks 8. Fine-tuned model checkpoint management 9. Training data ingestion 10. Fine-tuned model export 11. Feedback response to toolchains 12. High-Level Synthesis (HLS) Programming Support
Non-Functional needs	1. Performance: fast code generation and debugging 2. Accuracy: High code correctness 3. Scalability: parallel inference and fine-tuning tasks 4. Security: isolated execution for debugging, no execution of arbitrary code without sandboxing 5. Usability: Intuitive UI 6. Maintainability: versioning for prompts, fine-tuned models and output code 7. High code coverage test generation 8. Graceful error detection and recovery 9. Coding standard and best practice compliance
Targeted stakeholders	- developer - data scientist

Table 10: High-level needs elicited for KIO8.

Information	Description
AI4SWEng tool/component	AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components (KIO8)
Short description	AI-assisted code adaptation and optimization for multi-platform compatibility. Cross-compilation and automated reconfiguration across hardware architectures.
Lead partner	CEA (T4.3)
Expected TRL Improvement	2 to 5
High-level Inputs	- Code base; - Information about compatibility and architectures/hardware requirements;
Provisioned Outputs	- Optimized version for specified architectures; - Report about identified opportunities and changes proposed/made; - Recommendation in terms of software configuration for compatibility with the platform;
Hardware needs	- Computation environment for LLM - Adequate storage if we use a RAG
UI needs	- Input area/file download system for codebase; - Architecture selection field; - Optimised code for every specified architectures; - Logs/report for identified opportunities and code optimisations; - Cancel specific optimisations section;
Information flow	- Code base + architecture requirements -> AI Cross-compilation tool -> Optimized code + configurations + report
Functional needs	1. Code optimisation depending on the specified architectures and softwares

	2. Software reconfiguration recommendations depending on the optimised code and the choosen architecture 3. Report of the identified optimisation opportunities and the proposed changes 4. Possibility to change manually some unwanted optimisations 5. Ensure iterative code generation via interaction with toolchain
Non-Functional needs	1. Performance: fast and efficient in terms of resources 2. Scalability: the tool must be able to perform optimisations for large code bases 3. Sustainability: the tool must be documented and easy to update/debbug 4. Transparency: the user must be able to retrieve information about the optimisations made by the tool 5. Ergonomic/user-friendly: UI must be intuitive and reports easily readable 6. Secure: data must be securely stored
Targeted stakeholders	- software engineers

Table 11: High-level needs elicited for KIO9.

Information	Description
AI4SWEng tool/component	Responsible Innovation and Design-by-Contract Compliant Coding (KIO9)
Short description	Ensures AI-generated code adheres to ethical, privacy, and regulatory standards. Uses ontology-based AI frameworks for compliance enforcement.
Lead partner	ETCH
Expected TRL Improvement	2 to 4, 5
High-level Inputs	-code base - information and rules (e.g. rule set or an ontology-based risks model) about ethical and regulatory constraints (e.g. AI-Act, Cyber-Act, ALTAI, etc.)
Provisioned Outputs	Hash-based compliance check results summary (report) with alerts Recommendations to correct privacy, security or ethical concerns
Hardware needs	- Computation environment for a decentralized hashing mechanism (e.g. blockchain) - Adequate data storage if we need to store the code-base and related version-control system (we can use Github-related repositories and versioning systems as agreed in Tallinn)
UI needs	- Input area/file download system for codebase ; - Links to privacy, security, ethical compliance rules information (GDPR requirement); - Highlighted compliance errors or defects - Recommendations as comments for the developers - multifactor authentication and access control interfaces supporting the decentralised and collaborative working environment - Logs/reports ;
Information flow	- Code base + privacy/security/ethics requirements -> Decentralised identity management and access control (for accountability and authentication) -> AI compliance check tool -> Compliance check results + report + recommendations

Functional needs	1. Hash-based decentralised code integrity and security check mechanism to identify code manipulations and preventing untrusted sources 2. Multifactor Authentication mechanism for users' identity management 3. End-to-end security enabling encrypted storage or data transmission (e.g. against MitM)
Non-Functional needs	1. Performance: fast authentication and efficient data transmission, downloading and reporting in terms of resources 2. Scalability: should support a decentralised network that can be used by huge development teams connecting to the system from different end points 3. Sustainability: Energy-efficient decentralised networks and tools should be selected, easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects) 4. Transparency: inputs and outputs should be transparent to the user as the compliance check should be conducted at backend 5. Ergonomic/user-friendly: the compliance check analysis must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user 6. Security: end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Targeted stakeholders	Developers, product owners

Table 12: High-level needs elicited for KIO10.

Information	Description
AI4SWEng tool/component	Energy-efficiency with Model-Driven TinyML and LLMs (KIO10)
Short description	Optimizes AI-driven software development for low-power environments. Model-driven TinyML techniques for resource-constrained devices (IoT, Edge AI, Embedded Systems).
Lead partner	UREAD
Expected TRL Improvement	3 to 5
High-level Inputs	1. Real-time sensor data (battery voltage, temperature, cycle count, usage patterns) 2. Task-specific LLM prompts optimized for energy-efficient inference 3. Multi-level abstraction models (platform-independent to platform-specific) 4. Battery RUL prediction training datasets with energy consumption metrics 5. Target platform specifications and ultra-low-power constraints 6. Energy efficiency benchmarks and sustainability requirements
Provisioned Outputs	1. Optimised and quantized task-specific LLMs for Battery RUL prediction 2. Platform-specific embedded code generated via model-driven engineering

	3. Real-time Battery RUL predictions with energy consumption tracking 4. Performance metrics: inference latency, energy usage, CO2 footprint reduction 5. Multi-platform deployment packages for diverse IoT hardware 6. Energy efficiency reports comparing cloud vs. edge processing
Hardware needs	1. Ultra-low-power microcontrollers 2. Battery-powered IoT devices with integrated condition sensors 3. TensorFlow Lite Micro-compatible processors 4. Low-power wireless modules 5. Energy harvesting capabilities (solar/thermal) for sustained operation 6. Power management units with deep sleep modes
UI needs	1. Energy-conscious dashboard displaying: (a) Real-time power consumption metrics (b) CO2 footprint reduction comparisons (c) Battery RUL predictions with confidence intervals (d) Model deployment and optimization tools 2. Sustainability impact visualization (energy saved vs. cloud processing) 3. Multi-level abstraction model editor for platform-independent design 4. Alert system for critical battery states and energy thresholds
Information flow	Multi-Level Models → Model-Driven Engineering → Platform-Specific Code → TensorFlow Lite Micro Optimization → Battery Sensor Data → Ultra-Low-Power MCU → On-Device LLM Inference → Battery RUL Predictions → Energy Metrics Logging → Sustainability Impact Assessment → Model Performance Analysis → Code Generation Optimization
Functional needs	1. Execute Machine Learning Tasks on Resource-Constrained Devices 2. Enable ML Execution Using TinyML 3. Use Task-Specific LLMs for On-Device Analytics 4. Perform Analytics Directly on Edge Devices 5. Apply Model-Driven Software Engineering Principles 6. Address Complex Design Challenges with Domain-Specific Modeling 7. Address Complex Validation Challenges with Domain-Specific Modeling 8. Generate Code for Diverse IoT Platforms 9. Support Deployment on Diverse IoT Platforms 10. Support Deployment on Ultra-Low-Power Microcontrollers 11. Enable Battery Remaining Useful Life Prediction Use Case 12. Integrate with TensorFlow Lite for ML Operations 13. Integrate with Microcontroller Variant of TensorFlow Lite
Non-Functional needs	1. Reduce Energy Consumption by Limiting Cloud Data Transfer 2. Minimize Energy Needs via On-Device ML Tasks 3. Mitigate AI's Contribution to Global Electricity Usage 4. Mitigate AI's Contribution to Global CO2 Emissions 5. Promote Sustainability in Low-Power AI Development 6. Uphold Privacy Through On-Device Processing 7. Uphold Security Through On-Device Processing

	8. Enhance ML Performance Across Network Conditions 9. Enhance Service Quality Across Network Conditions 10. Provide Multi-Level Abstraction in Software Models 11. Support Abstraction from Platform-Independent Models 12. Support Abstraction to Platform-Specific Models 13. Facilitate Robust Development Through Multi-Level Abstraction 14. Facilitate Robust Validation Through Multi-Level Abstraction 15. Energy Consumption Metrics
Targeted stakeholders	1. Embedded systems developers 2. AI/ML researchers working on edge computing 3. IoT device manufacturers 4. Battery management system (BMS) engineers 5. Industries relying on smart IoT.

Table 13: High-level needs elicited for KIO11.

Information	Description
AI4SWEng tool/component	AI-Powered Test Automation Tool (TAT) (KIO11)
Short description	Automated software testing framework using: Fuzz testing (AFL++, OSS-Fuzz). Symbolic execution (KLEE, Angr). AI-driven security validation and compliance checks. Seamless integration into DevOps pipelines.
Lead partner	UCCS -> ERGTECH (T3.4) --> UNIGE
Expected TRL Improvement	3 to 5
High-level Inputs	*Codebase or API * software development workflow *AI-powered tools and SW development framework (T4.1 outputs) * Fine-tuned LLM for vulnerability detection (T4.1 output) *LLMs for generating test scenarios and simulating various patterns, intrusion, con-compliant transactions (T4.2 output) * Selection of the static/dynamic binary symbolic execution tool (e.g. Angr, T4.2 output)
Provisioned Outputs	*Functional Test suite specific to the target architecure codebase *Testing and validation results (documentation or report, training materials for KIO13)
Hardware needs	- Computation environment for the LLMs, Plugins and cyber attack database - Adequate data storage if we need to store the code-base and related version-control system (we can use Github-related repositories and versioning systems as agreed in Tallinn)
UI needs	- Input area/file download system for codebase or link to the API to be assessed ; - Selection of the tools for fuzztesting, Angr, etc.; - Visualisation of the workflow and test scenarios and test coverage; - Logs/reports ;
Information flow	Code base or API + Test requirements -> Decentralised identitity management and access control (for accountability and authentication) -> generated test scenarios --> underlying TAT tool(s), e.g. Angr, Fuzz,etc. --> TAT results + report + recommendations

Functional needs	1. Automated self-healing testing to improve context awareness via contextual AI models, LLMs 2. AI-driven test case generation 3. Refinement of test cases (prioritation and redundancy detection, prunin if needed) 4. Automated test script generation 5. Visualisation of TAT results (GUI)
Non-Functional needs	1. Performance 2. Scalability: solution must be used by huge development teams connecting to the system from different end points 3. Sustainability: easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects) 4. Transparency: inputs and outputs should be transparent to the user as the TAT should be conducted at backend 5. Ergonomic/user-friendly: the TAT must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user 6. Security: Cyber coverage including MitM, DoS, SQL-injection, XSS, phishing and authentication related attacks; end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Targeted stakeholders	Developers, product owners

Table 14: High-level needs elicited for KIO12.

Information	Description
AI4SWEng tool/component	AI-based Cybersecurity & Multi-Level Security by Design (KIO12)
Short description	AI-driven intrusion detection and vulnerability assessment. Automates API security testing and real-world attack scenario analysis.
Lead partner	ERARGE
Expected TRL Improvement	3 to 5
High-level Inputs	- codebase or API to be assessed - Fine-tuned LLM for vulnerability detection (T4.1 output) - IDE Plugins (T4.1 output) - ENISA cyber threat taxonomy (for cyber attack database) + Pre-TARA - LLMs for generating test scenarios and simulating various patterns, intrusion, con-compliant transactions (T4.2 output) - Selection of the static/dynamic binary symbolic execution tool (e.g. Angr, T4.2 output)
Provisioned Outputs	Test report including test scenarios and patterns + intrusion detection and con-compliant transaction detection results Materials for training purposes (to be input to KIO13)
Hardware needs	- Computation environment for the LLMs, Plugins and cyber attack database - Adequate data storage if we need to store the code-base and

	related version-control system (we can use Github-related repositories and versioning systems as agreed in Tallinn)
UI needs	- Input area/file download system for codebase or link to the API to be assessed ; - Links to privacy, security rules information (GDPR requirement) (KIO9); - Highlighted vulnerabilities, defects and weaknesses (defect localization); - summary of testing procedures (or links to testing procedures enabling end users to see the test coverage) interlinked with the ENISA threats; - Recommendations as comments or decision-making help/hints for the developers; - Logs/reports ;
Information flow	- Code base or API + security requirements -> Decentralised identity management and access control (for accountability and authentication) -> generated test scenarios --> LLM-based vulnerability analysis tool(s) --> Check results + report + recommendations
Functional needs	1. AI-based security check mechanism and vulnerability analysis to identify MitM, DoS, SQL-injection, XSS, phishing and authentication related attacks 2. Security test scenario generation 3. End-to-end security enabling encrypted storage or data transmission + multifactor authentication
Non-Functional needs	1. Performance: accurate test scenario generation and vulnerability detection 2. Scalability: should support adding new types of vulnerabilities or attack types depending on the development; solution must be used by huge development teams connecting to the system from different end points 3. Sustainability: easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., top-secret, commercial, or toy projects) 4. Transparency: inputs and outputs should be transparent to the user as the vulnerability check and test scenario generations should be conducted at backend 5. Ergonomic/user-friendly: the test generation and vulnerability analysis must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user 6. Security: Cyber coverage including MitM, DoS, SQL-injection, XSS, phishing and authentication related attacks; N13end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., top-secret, commercial, or toy projects)
Targeted stakeholders	Developers, product owners

Table 15: High-level needs elicited for KIO13.

Information	Description

AI4SWEng tool/component	Developer Training for using AI-Powered Software Development Tool (KIO13)
Short description	Training and educational resources for developers on AI-assisted software engineering. Ontologically framed Prompt Engineering for optimizing AI-based code generation.
Lead partner	UREAD
Expected TRL Improvement	2 to 4
High-level Inputs s	1. Ontological commitment frameworks for software requirements 2. Multi-stage development workflows (NLP → formal requirements → architecture → code) 3. LLM prompt templates for software engineering tasks 4. Agile development pipeline specifications and constraints 5. Compliance objectives and regulatory requirements for software systems 6. Alternative prompting approaches and assessment criteria 7. Software engineering skill gap analysis and training needs assessment
Provisioned Outputs	1. Structured multimedia training programme for AI-powered software development 2. Advanced prompt engineering certification modules 3. Ontology-based requirements framing and visualization tools 4. Qualitative assessment frameworks for prompt effectiveness evaluation 5. Interactive learning pathways for agile AI-development integration 6. Career development roadmaps for software-AI engineering talent 7. Enterprise productivity enhancement training packages 8. Web-based ontological engineering and visualization platform
Hardware needs	1. Web servers for hosting training service 2. Storage infrastructure for multimedia content and ontologies
UI needs	1. Comprehensive web-based interface featuring: (a) Interactive ontological engineering and visualization workspace (b) Multi-stage software development pipeline simulator (c) Prompt engineering laboratory with real-time LLM testing (d) Requirements framing and prioritization collaborative tools (e) Progress tracking and competency assessment dashboards 2. Mobile-responsive design for flexible learning access 3. Real-time collaboration tools for team-based training exercises 4. Adaptive learning interface responding to individual skill progression
Information flow	Skills Gap Analysis → Ontological Commitment Framework → Requirements Specification → Multi-Stage Development Workflow → Prompt Engineering Templates → LLM Integration Training → Qualitative Assessment Module → Alternative Approach Evaluation → Competency Certification → Career Development Planning → Enterprise Productivity Metrics → Continuous Learning Optimization → Industry Impact Assessment
Functional needs	1. Ontological Input Handling 2. Requirements Derivation 3. Ontology Building Tools 4. Visualization Generation 5. Training Content Delivery

	6. Prompt Skills Training 7. Output Evaluation 8. Pipeline Integration 9. Progress Monitoring 10. Impact Reporting 11. Interactive Learning Modules 12. Full user companion guide
Non-Functional needs	1. Interface Usability 2. Accessibility Standards 3. Response Time 4. User Capacity Handling 5. System Availability 6. Data Protection 7. Tool Compatibility 8. Content Update Ease 9. Resource Optimization 10. Outcome Tracking 11. Multi-Language Support
Targeted stakeholders	1. Software developers transitioning to AI-powered development workflows 2. AI-software engineering professionals seeking advanced prompt engineering skills 3. Agile development teams adopting LLM-integrated pipelines 4. Technical educators and curriculum developers in software engineering 5. Enterprise learning and development departments 6. Software engineering managers implementing AI-powered productivity tools 7. Career development counselors in tech industry 8. Academic institutions offering software engineering programs 9. Professional certification bodies for software-AI engineering 10. HR departments addressing software engineering talent shortages

3.5 Preliminary Functional and Non-functional Requirements scoping and elicitation based on the UI-REF methodology

This section refines the KIO descriptions into a set of high-level functional requirements (FRs) and non-functional requirements (NFRs) based on the specifications in Section 3.3. These requirements are based on the preliminary analysis of actor-system interactions and the elicited needs as presented in Sections 3.3 and 3.4, respectively. The detailed requirements can be seen in D2.6 with their technical specifications, interdependencies, and quantified evaluation methodology. Each requirement is linked to one or more KIOs and is mapped to the relevant UI references. This mapping ensures that the system design remains user-driven, transparent, and traceable during the development process. Requirements are divided into three groups: mandatory, desirable, or optional [58]. This shows how important they are in order to fulfill the project's goals.

3.5.1 UI-REF methodology

The UI-REF (User Interface Requirements Elicitation Framework) methodologically guided framework, as introduced by Badii in 2008 [59], serves as a comprehensive approach to various aspects of requirements elicitation, conflict resolution, ranking, and usability and impact assessment. UI-REF is used in AI4SWEng to re-elicit the requirements aligned with the KIOs and Use Cases (UCs) in the project. UI-REF will be used to generate an overall AI4SWEng requirements ontology to illustrate the relations among these items of the project scope. Before proceeding, an overview of the key components and principles of UI-REF is as follows:

1. Holistic Requirements Elicitation: UI-REF aims to comprehensively capture all relevant requirements for a system. It considers the diverse needs and contexts of users to ensure that no critical requirements are overlooked.
2. Conflict Resolution: UI-REF provides mechanisms to identify and resolve conflicts among different requirements or stakeholder interests. This ensures that the final set of requirements is coherent and can be effectively implemented.
3. Ranking and Prioritisation: UI-REF categorises requirements into priority levels. The framework supports a descending-order prioritisation approach to determine which requirements should be implemented first. This helps in focusing on the most critical and high-impact features or functionalities. The requirements are ranked as follows:
    - MUST-have/Mandatory – These are those core design features which are perceived by the majority of a user group as offering the most needed affordances and the target system can provide that.
    - SHOULD-have/Desirable – These are those features that are the lower-priority design features to be provided by the system within the project scope (within the limits of resources and technological constraints).
    - NICE-to-have/Optional – These are those features that are highly contextualised to particular (sub)-sectors of the user group and are less common, and/or possibly more controversial and conflictual categories.
4. Relationships-Based Usability and Impact Assessment: UI-REF incorporates methods for assessing usability and the potential impact of requirements. It considers how each requirement affects the overall user experience and the achievement of key performance indicators (KPIs).
5. Ontologically-Guided Models: UI-REF utilises ontological models to ensure a clear and structured representation of requirements. This helps in formalising and documenting requirements in a standardised manner.
6. Techniques and Tools: UI-REF employs a variety of techniques and tools, including online self-reporting, card sorting, laddering, nested video interviews, cognitive probes, and task walk-throughs. These techniques are used to gather user input, understand their needs, and clarify requirements. Techniques and tools are planned to be developed and deployed across AI4SWEng work packages, specifically in WP3 to WP6.
7. Contextualization: UI-REF emphasizes the importance of understanding the context in which requirements will be applied. It takes into account the specific use-contexts or scenarios in which the AI4SWEng solutions will be used.
8. User Involvement: UI-REF promotes user involvement throughout the requirements elicitation process. Users play a crucial role in specifying their needs, confirming requirements, and prioritising them.
9. Co-Design: UI-REF supports co-design approaches, where users actively participate in shaping the design of the solution. It enables users to specify context-aware requirements, including those related to privacy, security, and usability.

In summary, UI-REF is a methodologically guided framework that provides a structured and user-centric approach to requirements elicitation, conflict resolution, prioritisation, and usability assessment. It ensures

that system designers have a clear understanding of user needs and context, leading to the development of solutions that better meet user expectations and KPIs. The outputs of UI-REF methodology will be the baseline of requirements and use case specifications in D2.6. Moreover, UI-REF forms the basis of State-of-X (X=Art, Market, Practice) analyses, as will appear in D2.2.

3.5.2. Functional Requirements (FRs)

The FRs define the main capabilities of the AI4SWEng suite. The suite's KIOs must address the needs of its targeted user groups and use cases. These requirements cover features such as AI-enabled code analysis and generation, reverse execution, automated testing, compliance-by-design, and energy-aware deployment. They are derived from three main sources: (i) user and stakeholder needs, (ii) mapping of project use cases to measurable outcomes, and (iii) gaps identified in the state of the art. This approach ensures that the FRs are in line with the AI4SWEng Specific Objectives (SOs) and Key Performance Indicators (KPIs).

3.5.2.1 Functional Requirements for KIO1: AI4SWEng AI Engineering Suite

The FRs for the **KIO1** set up the basic architecture and operations for the AI4SWEng suite. The requirements are all defined as mandatory and comprise the core technical foundation that is required for successful instantiation and subsequent interoperability of the KIOs. The five main requirements are designed to address the most important needs of systems integration, user experience, process management, deployment, and access to advanced AI services. The suite should include a unified KIO integration platform, which will serve as the backbone of the suite to ensure efficient communication between all the KIOs. There is also a need for a front-end GUI to offer users a simple, central way to manage the entire AI-augmented software development lifecycle. The platform must support DevOps workflow orchestration, natively integrating AI-powered tools directly into continuous integration and continuous deployment pipelines. Technical delivery will be ensured through the requirement for containerized service deployment, which establishes a scalable and portable mechanism for distributing all platform services and KIO outputs. Finally, a critical functional layer shall be established via the Horizontal LLM Integration requirement, which standardizes the interface for all large language models and other AI services, ensuring uniform accessibility across the AI4SWEng ecosystem. The complete, detailed specifications for these requirements are provided in KIO1.

3.5.2.2 Functional Requirements for KIO2: Execution Analysis and AI-Powered Diagnosis

The FRs for **KIO2** focus on intelligent execution analysis and fault localization. This KIO introduces the essential foundation for tracing, replaying, and analyzing behavior, which helps developers with debugging and validation. KIO2 enables developers to monitor execution, reconstruct previous states, and trace back from error conditions to their root causes through reversible trace capture, AI-assisted replay, and dynamic slicing. These requirements aim to provide a diagnostic workflow in which trace data support slicing, replay, and AI-based fault localization to ensure that information flows across the analysis pipeline. These mechanisms can enable developers to better understand the system's behavior and fix bugs. KIO2 aims to reduce manual debugging processes and increase reliability during software validation through the implementation of automated state reconstruction with explainable fault analysis. Optional AI components further extend this process with capabilities for bug detection and fix suggestions to strengthen the platform's adaptability and developer productivity. A detailed description of these requirements is given in KIO2.

3.5.2.3 Functional Requirements for KIO3: Natural Language Processing (NLP) to Formal Requirements Component

The FRs for **KIO3** provide the most important link between communication with stakeholders and the subsequent steps in technical development. KIO3 has four mandatory requirements that form its core function. The KIO must analyze natural language to extract key intent and actors. It must use an AI model to generate formal user stories from this input. It is also required to attach specific acceptance criteria to every story. Moreover, it must include an export function for all project artifacts to ensure interoperability. There are also three desirable requirements to improve quality and productivity. The KIO shall be able to detect and

highlight vague or ambiguous user input. It should allow users to edit all AI-generated content to maintain oversight. The KIO should also have the capacity for automated wireframe generation. Detailed tables of these requirements are given in KIO3.

3.5.2.4 Functional Requirements for KIO4: Formal Requirements-to-Code Architectural Planning Component

The FRs for **KIO4** represent the essential transition from the requirements engineering process to the high-level architectural design architecture. The core set of mandatory requirements focuses on the foundational support for the system's integrity. The KIO must first support the formal ingestion of requirements (from KIO3) and subsequently parse them as well as to identify architectural implications that govern the software under development. Based on this analysis, the KIO is required to suggest a high-level architecture as well as to export an architectural component to support tool integration, which is necessary for the subsequent development stages in WP3. The KIO also requires a mandatory function to allow the user to accept, modify, or reject suggestions, thereby strictly maintaining a human-in-the-loop validation process for all AI-generated designs. Furthermore, there is also a mandatory requirement to provide a traceability matrix between requirements and design components, as it is an essential compliance feature for auditing and verification that must be built into the architecture. Several desirable and optional features are needed in order to enhance user productivity and quality assurance as well. These include the ability to generate UML diagrams for visualization and the capacity to export project scaffolding/code skeletons to immediately accelerate the coding phase. Additionally, the documentation of the architecture is desirable to ensure that the justification behind critical design decisions is preserved for future maintenance and review. The complete, detailed specification for these requirements is provided in KIO4.

3.5.2.5 Functional Requirements for KIO5: Synthetic Data Generation Tool

The FRs for **KIO5** focus entirely on synthetic data generation and management to produce high-fidelity, usable data for critical use cases. The core requirement is the generation of three distinct types of synthetic assets: (i) network traffic (including both compliant flows and attack scenarios), (ii) complex GIS layers to model energy consumption scenarios, and (iii) master medical datasets created by synthetically linking entries. A major focus is placed on quality and maintainability to propose validation methods for every generation task and desirable to provide thorough reporting. This confirms KIO5's role as the main component responsible for the generation of diverse, validated, and re-usable synthetic data for testing and development. A complete, detailed specification for these requirements is provided in KIO5.

3.5.2.6 Functional Requirements for KIO6: Data Validation-Preparation Tool

The FRs for **KIO6** focus on the establishment of data provenance, quality assurance, and automated preparation pipelines which are essential to train and operate the AI-powered tools within the AI4SWEng project. KIO6 has one mandatory requirement; the system must perform automated data pre-processing. This is considered a core check. The rest of the requirements are desirable and related to usability improvement. The KIO shall be able to compare input data against a pre-defined schema. It should be able to generate detailed reports on data validation, statistics, errors, and quality. A function to prepare data specifically for an LLM is also desirable. Finally, a user interface is required to display outputs, show step details, and allow users to edit the data schema. The complete, detailed specification for these requirements is provided in KIO6.

3.5.2.7 Functional Requirements for KIO7: AI-SysDev Tool incorporating: (i) LLM Fine Tuning Component, and (ii) LLM Inferencing (Code Generation, Code Understanding & Debugging, Pro-Active Coding Suggestion Component

The FRs for **KIO7** include the core AI services and the necessary MLOps pipeline for the suite. This KIO focuses on the use of LLMs to support developers in order to ensure model maintainability and improvement. The mandatory requirements include lifecycle management and core code-oriented operations. This begins with

the MLOps pipeline. It requires both training data ingestion and the ability to fine-tune LLMs to adapt them to proprietary and project-specific codebases. The integration with MLOps necessitates the need to export fine-tuned models as well as the ability to provide feedback response to toolchains. This will ensure that model improvements can be re-integrated into the platform. On the developer interaction side, the KIO requires the ability to accept prompts and generate code and, critically, to debug code and return annotated diagnostics and fixes. Furthermore, HLS Programming Support is mandatory in order to optimize the LLM's code generation process. The desirable requirements focus on the enhancement of the usability as well as on the expansion of the KIO capabilities. In addition, they focus on the inclusion of features to improve code quality, such as the recommendation of performance or security improvements, and functionalities to increase efficiency, such as the capability to auto-generate and execute test cases. To improve the LLM's context, the ability to retrieve relevant code patterns or documentation (RAG) is desirable. Finally, it is important to support multiple programming languages and frameworks and to manage the resulting models via fine-tuned model checkpoint management. This will enhance the general applicability and long-term maintainability of the AI4SWEng solution. The complete, detailed specification for these requirements is provided in KIO7.

3.5.2.8 Functional Requirements for KIO8: AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components

The FRs for KIO8 cover the post-generation refinement loop to make sure that the final delivered code is efficient and aligned with the overall architecture. The KIO must perform code optimization based on specified architectures. It must provide software reconfiguration recommendations based on optimized code schemas, and it must also ensure iterative code generation by interacting with the toolchain. The only desirable requirement is usability. The KIO should be able to generate a detailed report with the identified optimization opportunities and the proposed changes. Finally, there is also one optional requirement for user control. The solution should allow users to manually change or to revert any unwanted optimizations. The complete, detailed specification for these requirements is provided in KIO8.

3.5.2.9 Functional Requirements for KIO9: Responsible Innovation and Design-by-Contract Compliant Coding

The FRs for **KIO9** provide the foundational security baseline for the entire AI4SWEng suite. The FRs aim to secure the system's infrastructure and data integrity. Specifically, the system must implement a Multifactor Authentication (MFA) mechanism to establish high-assurance user identity management and control access. For data and code integrity, a hash-based decentralized code integrity and security check mechanism is required to identify any unauthorized manipulations and prevent untrusted sources from entering the workflow. Finally, end-to-end security must be provided to support encrypted storage and data transmission in order to protect all sensitive information against external threats, such as MITM attacks. The detailed list of FRs for this KIO is provided in KIO9.

3.5.2.10 Functional Requirements for KIO10: Energy-efficiency with Model-Driven TinyML and LLMs

KIO10 has ten mandatory requirements which focus on core functionality and deployment. The KIO must be able to execute machine learning tasks on resource-constrained devices. It must enable the execution of ML workflows using TinyML and perform analytics directly on edge devices. The KIO must apply model-driven software engineering principles. It is mandatory to use domain-specific modeling to address complex design challenges, to support deployment on diverse IoT platforms. It must also support deployment across ultra-low-power microcontrollers. The KIO must also enable a "battery remaining useful life" prediction and be able to integrate with TensorFlow Lite for ML operations. Finally, the integration with the microcontroller variant of TensorFlow Lite is mandatory. There is only one desirable requirement which lies in the improvement of the validation process as the system should use domain-specific modeling to address complex validation challenges. Finally, there are two optional requirements designed to enhance capabilities. The KIO shall use

task-specific LLMs to support on-device analytics and generate code for diverse IoT platforms. The detailed list of FRs for this KIO is provided in KIO10.

3.5.2.11 Functional Requirements for KIO11: AI-Powered Test Automation Tool (TAT)

The FRs for **KIO11** include the mandatory capabilities which are required to automate and improve the testing phase of the software development lifecycle. The KIO's mandatory requirements include: (i) automated self-healing testing to improve context awareness via contextual AI models and LLMs, (ii) AI-driven test case generation, and (iii) automated test script generation and visualisation of TAT results (through a GUI). There is one desirable requirement which includes the refinement of test cases (prioritization and redundancy detection, pruning if needed). The complete list of these requirements is provided in KIO11.

3.5.2.12 Functional Requirements for KIO12: AI-based Cybersecurity & Multi-Level Security by Design

The FRs for **KIO12** are entirely mandatory since they aim to establish critical security and testing mechanisms within the system's operational environment. More specifically, this KIO aims to build a robust multi-level defense mechanism which incorporates security intelligence into the development and runtime process. Emphasis shall be given to AI-based security check and vulnerability analysis. This AI-based component must be able to identify a wide array of adversarial attacks, including MITM, DoS, SQL injection, XSS, phishing, and various authentication exploits. To ensure defense coverage, this KIO also requires security test scenario generation for automated validation of these defenses. Crucially, the establishment of a security by design principle requires end-to-end security and MFA for secure data handling and user identity management. The complete list of the FRs is provided in KIO12.

3.5.2.13 Functional Requirements for KIO13: Developer Training for using AI-Powered Software Development Tool

The FRs for **KIO13** are crucial in order to ensure user adoption and to maximize the productivity gains from the entire AI4SWEng system. KIO13 is all about building a smart training platform, especially for teaching developers how to use AI and write good prompts. The mandatory requirements are the core features: It needs tools to build and handle structured (ontological) inputs and then use those to figure out project requirements. It also has to create visuals to make these complex structures easy to understand. For the training part, the KIO must be able to deliver the lessons, specifically to train developers on "prompt skills," to let them practice with interactive modules, and then evaluate their work. It also needs to monitor their progress and provide a full user guide. On the other hand, there are two desirable requirements. The system should support pipeline integration, and it should also provide impact reporting. The complete, detailed tables for these requirements are presented in KIO13.

3.5.3. Non-functional Requirements (NFRs)

The NFRs define the conditions under which these functionalities must operate. They describe how the suite should perform and address dimensions such as scalability, interoperability, robustness, energy efficiency, explainability, and compliance with European regulatory frameworks including the AI Act, GDPR, and ENISA cybersecurity guidelines. By including these non-functional aspects, AI4SWEng ensures that its outputs are not only functionally advanced but also trustworthy, auditable, and sustainable, in line with European socio-ethical expectations for responsible AI.

3.5.3.1 Non-Functional Requirements for KIO1: AI4SWEng AI Engineering Suite

The NFRs for **KIO1** define the essential quality attributes needed to ensure the system's adoption, reliability, and compliance. The mandatory NFRs establish the core foundation for the suite's design and operation.

Architecturally, the system is required to be modular and support flexible workflow configurations for easy adaptation. Operationally, it must deliver reliability and scalability to handle increasing workloads and continuous DevOps. It is also crucial for the suite to ensure security in the integration layer and adhere to GDPR and Ethics Compliance standards. The single desirable NFR lies in the interoperability with external systems, which broadens the suite's scope by allowing it to integrate with existing legacy systems and third-party tools. The complete NFRs list for these requirements is provided in KIO1.

3.5.3.2 Non-Functional Requirements for KIO2: Bug Locate & Fix: Reverse Execution & Dynamic Slicing Component

The NFRs for **KIO2** define the main attributes that govern the performance, quality, and modularity of the debugging component, to ensure that the system can operate on complex, large-scale codebases. This critical baseline includes two core performance requirements: (i) performance (quick processing for common operations) and (ii) quality (better accuracy). In terms of performance, the KIO must meet a standard of high detection accuracy of faults. The KIO shall complete in ≤ 5 seconds under normal load conditions with desired hardware while achieving a minimum bug-localization accuracy of 95% compared to verified ground-truth datasets under normal load conditions. The remaining requirement focuses on modularity (updatable structure), which is an optional one which focuses on long-term maintenance, and convenience to facilitate future updates. This will ensure that successful update and integration tests are completed within 8 hours without regression failures. The detailed list of the NFRs is presented in KIO2.

3.5.3.3 Non-Functional Requirements for KIO3: Natural Language Processing (NLP) to Formal Requirements Component

The NFRs for **KIO3** are mainly divided into mandatory and desirable. The mandatory requirements focus on integrity which demands unambiguous accuracy in formal requirements, error recovery for continuous operation, security for stored data, and compliance with output formats. The desirable requirements prioritize futureproofing and developer experience, including high performance, robust scalability (especially for batch processing), intuitive usability, and support for architectural versioning. The detailed requirements are provided in KIO3.

3.5.3.4 Non-Functional Requirements for KIO4: Formal Requirements-to-Code Architectural Planning Component

The mandatory NFRs for **KIO4** aim to guarantee the integrity and reliability of the architecture. Accuracy is required to ensure that the proposed architecture has a high relevance to the FRs. Security is also mandatory to protect securely stored architectural data. The KIO must be able to support compatible output formats in order to be integrated into the DevOps toolchain. Furthermore, operational stability is critical for error detection and recovery to ensure continuous operation without losing progress during complex planning tasks. The remaining requirements are classified as desirable, which focus on the enhancement of efficiency and long-term usability. Performance and scalability are desired features that enhance operational efficiency for complex or high-volume requests. Usability is desired to improve the developer's experience and adoption. For a detailed list of these NFRs see KIO4.

3.5.3.5 Non-Functional Requirements for KIO5: Synthetic Data Generation Tool

The mandatory NFRs for **KIO5** define the core integrity and security baseline for the generated data. Accuracy is critical and requires that the synthetic data must statistically match the input data characteristics. Security is also mandatory to safeguard seed data and the generation process via protection, and injection safeguards. Desirable requirements focus on efficiency and adoption, including high performance (fast generation/low latency), scalability (to handle large datasets), and an Intuitive UI for easy configuration. The complete, detailed specification for these requirements is provided in KIO5.

3.5.3.6 Non-Functional Requirements for KIO6: Data Validation-Preparation Tool

The NFRs for **KIO6** aim to ensure secure and accurate data handling for AI model development. Reliability is crucial for accurate error detection and data validation reports, while security requires that all data must be securely stored throughout the preparation lifecycle. The desirable requirements focus on improving efficiency, maintainability, and user experience[1]. These include performance (optimized computation and fast processing), scalability (support for large data/schemas), and transparency (documented code, easy debugging, and user access to processing logs). A complete list of the requirements is provided in KIO6.

3.5.3.7 Non-Functional Requirements for KIO7: AI-SysDev Tool incorporating LLM Fine Tuning and Inferencing

The NFRs for **KIO7** are mandatory and demand high code correctness from generated outputs along with high code coverage test generation to guarantee reliability. Security requires robust defenses, isolated execution for debugging, and no execution of arbitrary code without sandboxing to protect the host system. Stability must be ensured by error detection and recovery throughout the complex fine-tuning and generation tasks. The remaining requirements are classified as desirable and focus on developer experience and efficiency. Performance and scalability are desired to enhance efficiency for large models. Usability (intuitive UI) and maintainability (maintain versioning for prompts, models, and code) are desired for ease of use and long-term project stability. Finally, coding standards and best practice compliance are desired for generated code quality. The complete, detailed specification for these requirements is provided in KIO7.

3.5.3.8 Non-Functional Requirements for KIO8: AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components

The NFRs for **KIO8** primarily focus on a set of desirable attributes which are related to efficiency, usability, and maintainability. Almost all requirements in this KIO are classified as desirable, to optimize cross-compilation and reconfiguration processes and enhance the developers' experience. Performance is desired to be fast and efficient in terms of resources. Scalability is also desirable to ensure that the KIO can perform optimisation tasks on large code bases. Usability requires that the UI should be intuitive, and that the extracted reports are easily readable. The KIO shall be also secure to ensure that the data will be securely stored. Sustainability is desired to ensure that the KIO is well documented and easy to update/debug. Transparency[2] requirement emphases its importance for explainability and trust in the optimization process. The complete and detailed list of NFRs is given in KIO8.

3.5.3.9 Non-Functional Requirements for KIO9: Responsible Innovation and Design-by-Contract Compliant Coding

The mandatory NFRs for **KIO9** define the main security and operational efficiency controls. Security is of paramount importance for end-to-end secure data handling and multifactor authentication, with an adaptable lightweight/heavyweight security approach based on project criticality. This adaptability is reinforced by sustainability, which also requires energy-efficient decentralized networks and adaptable security protocols. Fast performance for authentication and data transmission is also required. The desirable requirements aim to: (i) enhance scalability and user experience, in order to support a decentralized network for large teams, (ii) ensure the transparency of the inputs/outputs (with backend compliance checks), and (iii) ensure ergonomic and user-friendly compliance analysis via visual aids and recommendations. The complete, detailed specification for these requirements is provided in KIO9.

---

[1] The requirement "**Ergonomic/user-friendly**: UI must be intuitive and reports easily readable," is designated by the Functional Requirement ID FR-KIO6-03, a specific classification noted for retention in the prioritization process

[2] It is noted that the **Transparency** requirement was intended to be a **Functional Requirement** in the original specification, emphasizing its importance for explainability and trust in the optimization process.

3.5.3.10 Non-Functional Requirements for KIO10: Energy-efficiency with Model-Driven TinyML and LLMs

The NFRs for **KIO10** are predominantly mandatory, aiming to establish high standards centered on three core pillars. The sustainability and metrics require a 30-50% reduction in energy consumption (compared to cloud solutions) by limiting data transfer, thus mitigating the system's emissions. Robust edge operation requires on-device processing to uphold privacy and security, to ensure consistent ML performance and service quality regardless of network conditions. Finally, model abstraction requires multi-level software modeling to support flexible development from platform-independent to platform-specific models. The desirable and optional requirements of this KIO focus on energy minimization and validation processes. The complete detailed specification for these requirements is provided in KIO10.

3.5.3.11 Non-Functional Requirements for KIO11: AI-Powered Test Automation Tool (TAT)

The NFRs for **KIO11** aim to prioritize security, performance, and scalability as mandatory requirements for operational integrity. KIO11 has four mandatory NFRs. The KIO must meet its performance targets and demonstrate scalability, to enable its use by large, distributed development teams. It must also ensure transparency by making its inputs and outputs clear to the user even as tests run in the backend. A thorough security solution is required to address various cyber-attacks, ensure secure data handling and adaptable security levels based on project criticality. There are also two desirable requirements to ensure quality and usability. The system shall ensure sustainability through easy documentation and flexible security levels. Finally, it should be ergonomic and user-friendly to provide self-explanatory features, color-coding, and clear recommendations. The complete list of NFRs is provided in KIO11.

3.5.3.12 Non-Functional Requirements for KIO12: AI-based Cybersecurity & Multi-Level Security by Design

The NFRs for **KIO12** aim to establish the compliance of the system with quality and security standards. The system must ensure high precision in test scenario generation and vulnerability detection. Scalability is also important to handle large development teams and support the dynamic addition of vulnerability and attacks. A critical mandatory requirement is security. This has to do with cyber coverage against all common attacks (MITM, DoS, XSS, phishing, etc.), end-to-end data transmission and storage. The remaining requirements are desirable and focus on usability and maintainability, including sustainability (easy documentation), transparency (transparent I/O despite backend checks), and ergonomic features (self-explainable analysis with color codes and recommendations). The complete and detailed tables for these requirements can be shown in KIO12.

3.5.3.13 Non-Functional Requirements for KIO13: Developer Training for using AI-Powered Software Development Tool

The NFRs for KIO13 focus on the definition of a highly accessible, legally compliant, and secure training system for developers. KIO13 has six mandatory NFRs. The KIO must ensure interface usability. It must also adhere to accessibility standards and provide user capacity handling, guarantee data protection, and include outcome tracking. Finally, it must provide multi-language support. In addition, there are four desirable NFRs. The KIO should have a fast response time, maintain high system availability, and tool compatibility. As far as the optional requirements are concerned, the KIO should include resource optimization mechanisms. The complete specification of the NFRs is provided in KIO13.

3.6 Priorities distribution

The overall prioritization distribution is depicted in Figure 11 reflects that the majority of the FRs and NFRs have been assigned a mandatory priority. The increased concentration of mandatory requirements ensures that every KIO delivers a complete, secure, and fully compliant initial operational capability. This strategy mitigates project risk by emphasizing core automation and security baselines early in the project, while the desirable and optional requirements can provide flexibility for evolution and partner-specific customizations.

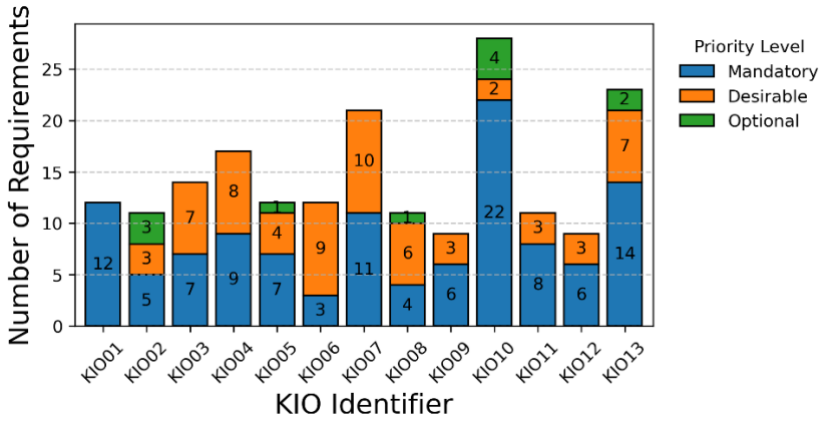

Figure 11: The distribution of requirements by priority level (Mandatory, Desirable and Optional) per KIO.

According to Figure 12, the distribution of the FRs and NFRs varies across the 13 KIOs. More specifically, KIO10 has the highest total number of requirements (28), with 15 non-functional and 13 functional ones, followed by KIO13, which includes 23 requirements (11 non-functional and 12 functional). In contrast, KIO9 and KIO12 have the fewest total requirements, both with only 12. Several KIOs, such as KIO3, KIO4, KIO8, KIO11 and KIO13, appear to have a more balanced composition between the functional and non-functional requirements. The heterogeneous distribution of the requirements across the different KIOs reflects the varying technical specifications and performance needs of the KIOs in terms of operational, usability, and quality considerations.

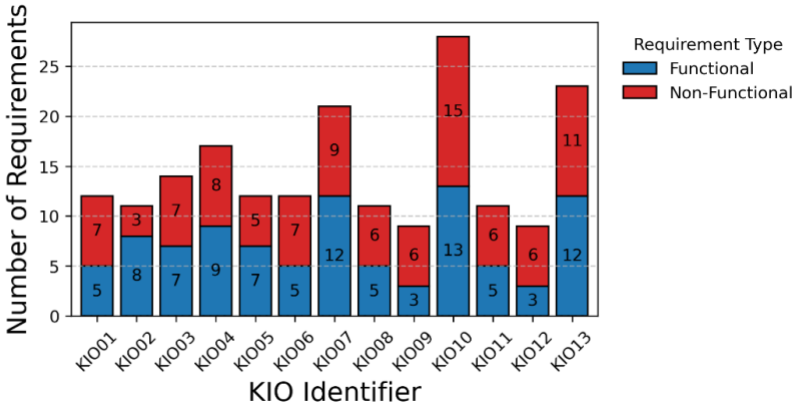

Figure 12: Bar chart showing how functional and non-functional requirements are distributed across the 13 KIOs.

4. Validation

4.1 Description of the process

The next step was to assess the relevance of the requirements for the 13 KIOs by also taking into consideration the identified gaps from the state of the art analysis. Specifically, the goal was to confirm if the identified requirements focus on areas where efficiency could be improved, the most, within the SDLC. For this reason, and in order to ensure a robust and robust assessment, a targeted survey and interview process was conducted to gather insights from six experienced software developers [59]. The main goal of this process was

to utilize their professional expertise in order to evaluate the practical relevance and impact of the collected FRs and NFRs, with a focus on the role of AI in software development. For this reason, a Google Form based interview (see Appendix D: Interview Form) was utilized to realize this structured evaluation process. The form consists of a series of questions which were designed to extract both qualitative and quantitative insights from the interviewees. The interview process began with the identification of the professional background of each participant through questions regarding their years of experience in the field and their exposure to AI technologies. Core questions related to the AI4SWEng project were then defined (e.g. what people thought a software developer's role would be in an AI-driven environment). To this end, participants were asked to identify the most important tasks, like the management of AI-powered tools, in order to make sure that AI is used ethically and building full-stack AI systems. In addition, the interview included questions regarding the way that AI affects the different stages of the SDLC. In particular, the software developers were asked to provide a specific example of how AI can improve their team's software development process with reference to the tools or methods that they use. The interview concluded with the exploration of the broader implications of AI on data management within the SDLC, an area of increasing importance. After the interview process was finalized, the participants were asked to evaluate the prioritization of the collected FRs and NFRs. Their feedback was based on their professional experience in the field. The information which was gathered from this first iteration was crucial not only for the evaluation of the prioritization of the collected requirements but also for their continuous refinement.

4.2 Results

4.2.1 Participant profiles

Six software development practitioners (N=6) participated in the evaluation process.

Figure 13: depicts the distribution of their self-reported AI skills along with a breakdown of their professional experience. The participants consist of five less-experienced developers (less than 7 years in the field) and a single highly experienced professional (7 years or more). Regarding AI familiarity, two respondents reported average experience, which indicates basic exposure to AI concepts through media and initial exploration. The remaining four participants reported good experience, which signifies their active engagement with AI tools both privately and professionally.

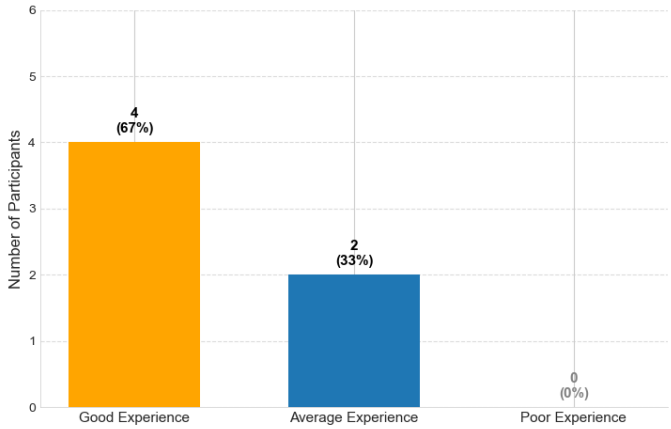

Figure 13: The distribution of familiarity levels (Good, Average, Poor) among interviewees.

4.2.2 Key opinions and insights

The responses of the practicioners revealed that everyone agreed that AI may change the way software is developed, but they also stressed the necessity for human control, ethical issues, and problems with integration. An overview of the main results that came out of the research is presented below.

4.2.2.1 The role of software developers in AI-augmented environments

It is worth noting that all participants recognized the existence of overlapping responsibilities for human professionals in AI-augmented environments, which highlights the importance of monitoring and integration rather than the replacement by AI. To this end, all participants identified several complex and overlapping tasks, which suggest a shift in how the role of software developers is perceived. The most frequently mentioned tasks are presented below:

- Building, monitoring, and maintaining AI-driven systems (cited by 4 respondents, 66.7%): This was the most frequently mentioned task, which shows that developers are starting to see AI models and tools as important parts of the development process.
- Managing and integrating AI-driven tools (cited by 3 respondents, 50.0%): This percentage reveals the practical use of AI programming assistants, such as GitHub Copilot. These tools are properly configured, used efficiently, and work well with existing DevOps pipelines.
- Ensuring ethical and secure use of AI (cited by 2 respondents, 33.3%): This finding shows that developers are becoming more conscious of data privacy and algorithmic bias. It also highlights their necessity to ensure that AI outputs are trustworthy and accurate.
- Proactively championing innovation and change (cited by 1 respondent, 16.7%): This point of view emphasizes the strategic part of the developer's job: uncovering new AI capabilities and figuring out how to use them to make both technical and business benefits.
- Driving the creation of more sophisticated, human-like software (cited by 1 respondent, 16.7%): This result supports the assumption that professionals perceive AI as a way to improve human skills, not as a way to replace them.

These findings, which are shown inFigure 14, highlight the fact that AI can function complementary to human capability. As one participant wrote: *"AI enhances software development by automating routine tasks and accelerating the development lifecycle. It doesn't replace practitioners but rather augments their capabilities, allowing them to focus on higher-level, creative problem-solving."*

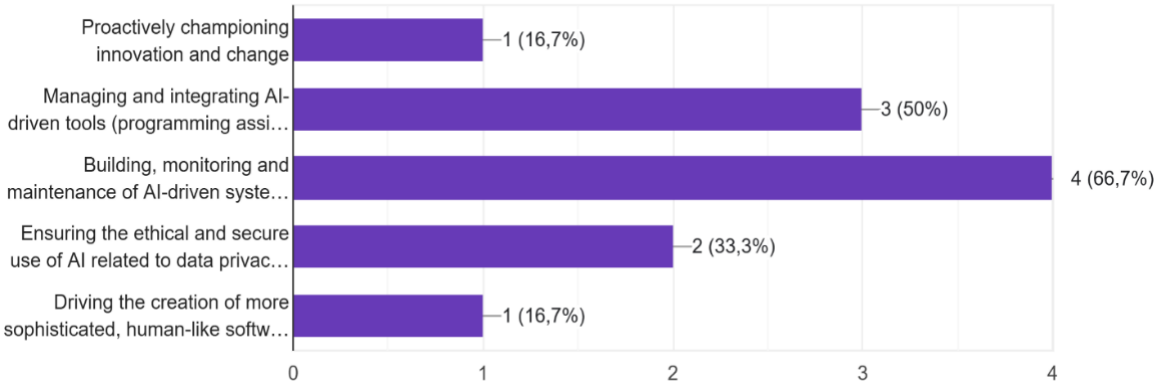

Figure 14: Distribution of key responsibilities identified by participants in AI-augmented software development environmens. (for a more comprenhensive view see Appendix D: Interview Form).

4.2.2.2 How AI can improve software development

The analysis of the textual data emphasizes how AI accelerates the process, improves output quality, and changes the developer's role toward more strategic and higher-level thinking. AI's most evident impact is in automation and efficiency, where it is seen to increase the overall speed of software development and delivery. This efficiency extends to project management, as AI assists with elements like task prioritization, effort estimation, and workflow optimization. Beyond speed, a strong theme emerges regarding the enhancement of code quality and security. Developers confirm that AI helps improve the final product by facilitating tasks such as code documentation, testing, and providing strong security analysis. Furthermore, the coding process is made faster and easier through code autocompletion and code generation. Finally, AI is perceived as a technology that augments human capabilities, rather than replacing them. AI takes care of repetitive and mechanical tasks which allow software developers to orient their attention to more complex and strategic work, such as higher-level, creative problem-solving. This process is also extended in other phases of the SDLC, such as requirement gathering and analysis which confirms the strategic importance of the AI4SWEng objectives.

4.2.2.3 Efficiency improvements in the SDLC phases

The responses reveal a strong agreement that AI's effectiveness is highly concentrated in the phases of code generation and validation. This tendency highlights the impact of AI towards the transformation of the way of writing code. This is closely related to testing and quality assurance, where participants rated the contribution of AI as a significant improvement. As shown in Figure 15, the participants recognize the power of AI in automated checks and error detection. For the remaining four phases, participants indicated a moderate level of agreement. This applies not only to the early stages of software requirements gathering, analysis and architecture design, but also to the later operational phases that include deployment, CI/CD, maintenance and support. The responses for these phases suggest that while AI can be useful, its influence is not transformative; human judgment and expertise remain the primary drivers. In summary, AI is seen as an essential “tool” for the execution phases of the SDLC (coding and testing) but to a less extent for the design and long-term lifecycle phases (requirements, architecture, deployment, maintenance).

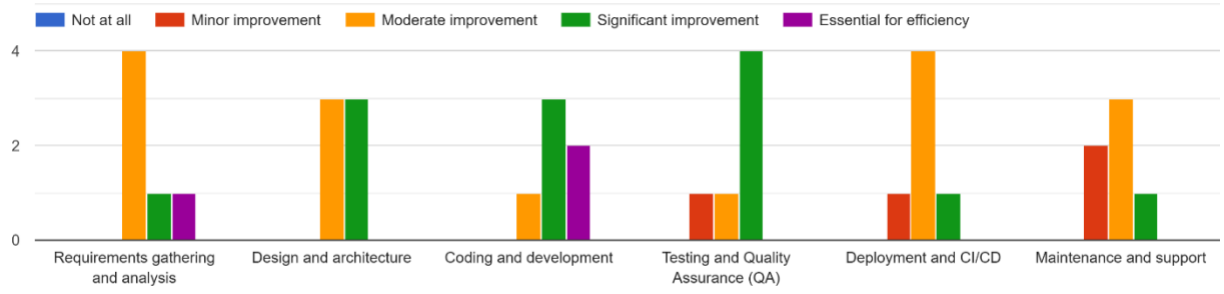

Figure 15: Perceived efficiency improvements across the Software Development Life Cycle (SDLC) phases, ranging from “Not at all” (blue) to “Essential for efficiency” (purple). (for a more comprenhensive view see Appendix D: Interview Form).

4.2.2.4 Examples of how AI improves software delivery

The responses demonstrate that AI is fundamentally transforming the software delivery process by accelerating timelines while significantly elevating product quality. The most important advantage lies in speed, where specific tools like GitHub Copilot and Postman AI Agent Builder can help one team to reduce

development time nearly in half. This efficiency is replicated across the workflow via AI assistants that provide code suggestions and automate test case generation, both contributing to faster merges and shorter release cycles. AI also drastically reduces time by automating tasks like code review and testing. The ability to automate high-volume quality assurance tasks frees developers to dedicate more time to strategic, high-value work such as architectural design and innovation. AI also serves as an effective learning and development tool, where one respondent mentions the use of LLMs as an interactive tutor to generate foundational code and explain best practices for new frameworks.

4.2.2.5 Consequences of AI-driven data management

According to the responses, it is evident that AI-driven data management is transforming software development through three main axons: (i) speed, (ii) quality, and (iii) developer's role. The most immediate benefit is a much faster delivery timeline. AI achieves this through the automation of complex data preparation tasks, including curation, monitoring, integration, and validation. This process reduces bugs due to low quality data that could otherwise lead to critical bugs. As a consequence, developers gain the opportunity to concentrate on another tasks, for instance architectural design, innovation and complex decision-making. However, according to the developers' responses, this reliance on AI can yield a new risk: developers must maintain strong oversight of data models and prioritize privacy and ethical governance to prevent the AI from making errors related to data management.

4.3 Requirements validation and prioritization

4.3.1 Level of agreement

The evaluation process of the 190 preliminary FRs and NFRs of the AI4SWEng suite (60) suggests that all participants are in great agreement. The results confirm that the proposed set of requirements is robust, well-aligned with project goals, and widely accepted by the participants. According to Figure 16, the validation process showed a ~95% overall agreement rate across all 1,140 individual requirement responses (190 requirements × 6 users). This high level of consensus indicates a strong initial quality in the requirements gathering and definition phase.

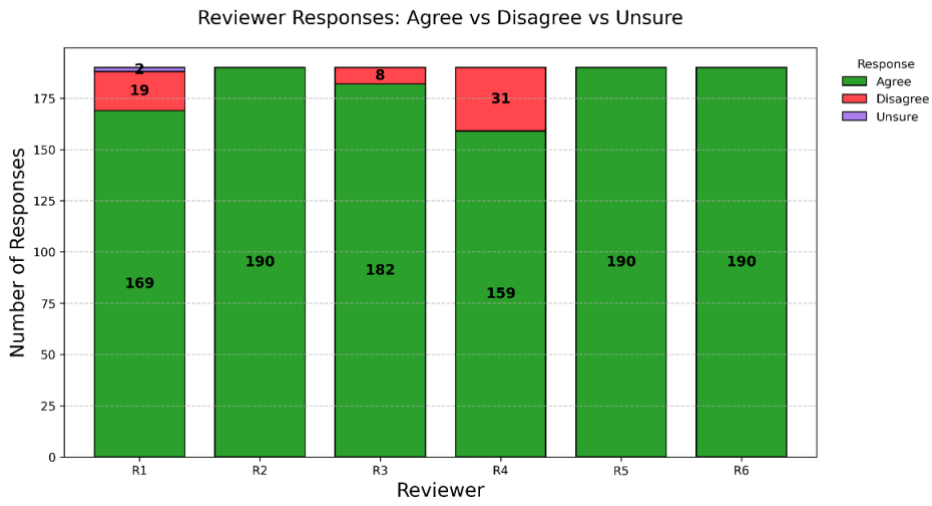

Figure 16: Distribution of the level of agreement of each participant according to the results of the interview process.

Requirements, which include the core capabilities and functionalities of the system, such automated requirement analysis and architecture design, code generation, bugs detection and data management, saw an extremely high level of agreement. This result confirms that the main requirements' collection during this first iteration process are essential for an effective AI-powered tool. However, the results highlight that refinement and revision are necessary in a second iteration, in order to improve the overall functionality of the suite. After the developers' feedback process, this concept mostly centered on a more detailed description and/or reprioritization of some requirements extending the existing functionalities. As is typical in complex software projects like AI4SWEng, the FRs and NFRs, which govern the quality attributes like security, performance, and integration, have to redefine and readjust through use case control scenarios. Nevertheless, the overwhelming majority of the quality requirements were fully accepted by the evaluation process. Overall, the evaluation process has confirmed that the requirements baseline is solid, well-accepted by stakeholders, and provides sufficient confidence to authorize the project's progression to the next phase of architectural design and implementation. On the other hand, although overall consensus was high, areas of minor disagreement were observed, mostly in the NFRs. The overall conclusion is that the project can proceed with the current requirements list, where immediate refinement efforts can follow in the future for priority adjustment. The disagreement confirms the robustness of this first iteration of the requirements' collection and the alignment with both practitioners' feedback and the AI4SWEng's strategic goals.

5. AI4SWEng requirements evolving process

While the AI4SWEng consortium maintains an evolving list of requirements in the project's GitHub repository, this deliverable focuses on the first iteration of requirements. These requirements represent an initial set on which the interview process and validation activities described in Section 4 were based. The inclusion of the latest updates on the requirements engineering process at this stage could create inconsistencies since the requirements refinement process will be dynamic and iterative throughout the project's lifecycle. This deliverable therefore references only the first iteration of requirements to ensure clarity and traceability. The ongoing revisions and additions recorded in the shared repository will be fully reflected and analyzed in subsequent deliverables (notably D2.6).

6. Conclusions

This deliverable marks the successful completion of the first iteration of the functional and non-functional requirements gathering process based on the high-level needs of the tools and components of the AI4SWEng project. The collected requirements were prioritized, and the assigned priorities were assessed via interviews with software engineering practitioners. The requirements documented in this deliverable refer to the first iteration of the requirements engineering process and serve as valuable input for the subsequent planning and development phases of the project. More specifically, the activities reported in D2.4 are directly related with Task 2.5, where the requirements will be completed and translated into formalized user needs and user stories to finalize the project's overall requirements specification. In fact, the requirements will be re-assessed, harmonized, and expanded in Task 2.5 to yield a unified set of requirements that reflect both partner-specific inputs and cross-tool dependencies. This approach will guarantee that the requirements elicited under D2.4 not only provide the basis for subsequent specification activities but also guide the elicitation, refinement, and prioritization process in T2.5. Furthermore, these requirements will be fundamental for Task 2.6 (Use-cases, test-cases, performance verification and framework architecture specifications) for the systematic definition of clear use-cases, test-cases, performance verification standards, and the detailed architecture specifications.

References

1. Sainio, K. (2023). *Generative Artificial Intelligence Assisting in Agile Project Pain Points* (Doctoral dissertation, Master's Thesis, Faculty of Management and Business, Tampere University, Finland).
2. Swimm. (n.d.). *Current Progress and Future Directions*. Retrieved from https://swimm.io/learn/ai-tools-for-developers/ai-code-review-how-it-works-and-3-tools-you-should-know
3. Turing. (n.d.). *AI Code Review: Improving Software Quality*. Retrieved from https://www.turing.com/blog/ai-code-review-improving-software-quality/
4. Codacy. (n.d.). *AI-Assisted Coding: 7 Pros and Cons to Consider*. Retrieved from https://blog.codacy.com/ai-assisted-coding-7-pros-and-cons-to-consider
5. Alon, U., Zilberstein, M., Levy, O., & Yahav, E. (2019). code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, *3*(POPL), 1-29.
6. Prasad, D. (2020). *Automated software testing: foundations, applications and challenges* (pp. 1-165). A. K. Jena, H. Das, & D. P. Mohapatra (Eds.). Springer.
7. Nguyen, N., & Nadi, S. (2022, May). An empirical evaluation of GitHub copilot's code suggestions. In *Proceedings of the 19th International Conference on Mining Software Repositories* (pp. 1-5).
8. Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv preprint arXiv:2304.10778*.
9. https://docs.anthropic.com/en/docs/claude-code/overview
10. Li, Yujia, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles et al. "Competition-level code generation with alphacode." *Science* 378, no. 6624 (2022): 1092-1097.
11. Roziere, B., Lachaux, M. A., Chanussot, L., & Lample, G. (2020). Unsupervised translation of programming languages. *Advances in neural information processing systems*, *33*, 20601-20611.
12. IBM. Project CodeNet. Retrieved from https://developer.ibm.com/exchanges/data/all/project-codenet/
13. OpenAI. OpenAI Codex. Retrieved from https://openai.com/blog/openai-codex
14. Akgul, T., & Mooney III, V. J. (2002). Instruction-level reverse execution for debugging. ACM SIGSOFT Software Engineering Notes, 28(1), 18-25.
15. Cingolani, D., Pellegrini, A., & Quaglia, F. (2015, August). RAMSES: Reversibility-based agent modeling and simulation environment with speculation-support. In European Conference on Parallel Processing (pp. 466-478). Cham: Springer International Publishing.
16. Lanese, I., Schultz, U. P., & Ulidowski, I. (2021). Reversible execution for robustness in embodied AI and industrial robots. It Professional, 23(3), 12-17.
17. https://github.com/cyberkaida/reverse-engineering-assistant
18. Hao, Y., Chen, W., Zhou, Z., & Cui, W. (2023). E&v: Prompting large language models to perform static analysis by pseudo-code execution and verification. arXiv preprint arXiv:2312.08477.
19. Alokla, A., Gad, W., Nazih, W., Aref, M., & Salem, A. B. (2022). Retrieval-based transformer pseudocode generation. Mathematics, 10(4), 604.
20. https://crypto.stanford.edu/seclab/sem-18-19/cui.html
21. Canavese, D., Regano, L., & Lioy, A. (2022, April). Computer-Aided Reverse Engineering of Protected Software. In International Workshop on Digital Sovereignty in Cyber Security: New Challenges in Future Vision (pp. 3-15). Cham: Springer Nature Switzerland.
22. Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. AI Perspectives, 2(1), 1.

23. Broadcom. Agile Requirements Designer. Retrieved from https://www.broadcom.com/products/software/continuous-testing/agile-requirements-designer
24. IBM. Requirements Management. Retrieved from https://www.ibm.com/products/requirements-management
25. Jama Software. Jama Connect. Retrieved from https://www.jamasoftware.com/platform/jama-connect/
26. Dias Canedo, E., & Cordeiro Mendes, B. (2020). Software requirements classification using machine learning algorithms. *Entropy*, *22*(9), 1057.
27. SmartBear. (n.d.). Zephyr Enterprise. Retrieved from https://smartbear.com/test-management/zephyr-enterprise/
28. Balsamiq. (n.d.). Balsamiq. Retrieved from https://balsamiq.com/
29. Adobe. (n.d.). Adobe XD. Retrieved from https://helpx.adobe.com/support/xd.html
30. Automation Solutions Using Design Patterns, Muller. J, 2022. https://www.uipath.com/community-blog/tutorials/automation-solutions-using-design-patterns
31. Nord, R.; Kurtz, Z. Using Machine Learning to Detect Design Patterns, Carnegie Mellon University, Software Engineering Institute's Insights (blog). March 2020; Available online: https://insights.sei.cmu.edu/blog/using-machine-learning-to-detect-design-patterns/
32. Isgrò, F., Ferraris, S. D., & Colombo, S. (2021, December). AI-enabled design tools: Current trends and future possibilities. In *Congress of the International Association of Societies of Design Research* (pp. 2836-2847). Singapore: Springer Nature Singapore.
33. Stoica, B. A., Sahoo, S. K., Larus, J. R., & Adve, V. S. (2021). Statistical program slicing: a hybrid slicing technique for analyzing deployed software. *arXiv preprint arXiv:2201.00060*.
34. Chochliouros, I. P., Spiliopoulou, A. S., Lazaridis, P., Dardamanis, A., Zaharis, Z., & Kostopoulos, A. (2020, May). Dynamic network slicing: Challenges and opportunities. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 47-60). Cham: Springer International Publishing.
35. Shu, Y., Li, Z., Karlsson, B., Lin, Y., Moscibroda, T., & Shin, K. (2019, April). Incrementally-deployable indoor navigation with automatic trace generation. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (pp. 2395-2403). IEEE.
36. https://www.ibm.com/docs/en/cloud-paks/cp-integration/2021.2.1?topic=implementing-ai-test-generation
37. Salimimoghadam, S., Ghanbaripour, A. N., Tumpa, R. J., Kamel Rahimi, A., Golmoradi, M., Rashidian, S., & Skitmore, M. (2025). The Rise of Artificial Intelligence in Project Management: A Systematic Literature Review of Current Opportunities, Enablers, and Barriers. *Buildings*, *15*(7), 1130.
38. Ząb, E. (2024). Will Artificial Intelligence Replace Project Managers? A Case Study on the Use of Three Leading Tools: Wrike AI, JIRA With Atlassian Intelligence, and Microsoft Project With Copilot AI. *Informatyka Ekonomiczna/Uniwersytet Ekonomiczny we Wrocławiu*, (1-2 (67-68)), 106-116.
39. A Vision Of AI-Enhanced Test Automation https://www.forbes.com/sites/forbestechcouncil/2023/09/11/a-vision-of-ai-enhanced-test-automation/
40. Best AI Testing Tools for Test Automation, Patel. T. ,2023 https://testgrid.io/blog/ai-test-automation-tools/
41. AI testing: A Web Developer's Guide to AI-Powered Testing https://www.testim.io/blog/whats-ai-testing/

42. Pei, K., Cao, Y., Yang, J., & Jana, S. (2017, October). Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).
43. Wu, J., Li, S., Li, J., Luo, L., Yu, H., & Sun, G. (2022, July). DeepCov: coverage guided deep learning framework fuzzing. In 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC) (pp. 399-404). IEEE.
44. Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., ... & See, S. (2019, July). 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 (pp. 146-157).
45. Ye, A., Wang, L., Zhao, L., & Ke, J. (2022, July). DANCe: Dynamic Adaptive Neuron Coverage for Fuzzing Deep Neural Networks. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.
46. https://github.com/AFLplusplus/AFLplusplus
47. https://github.com/google/oss-fuzz
48. https://github.com/klee
49. https://angr.io/
50. https://www.sogeti.com/ai-for-qe/section-9-trust-ai/chapter-1
51. https://eur-lex.europa.eu/eli/reg/2016/679/oj
52. https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai
53. https://www.enisa.europa.eu/publications/cybersecurity-of-ai-and-standardisation
54. Deng, Z., Xu, L., Liu, H., Hu, X., Duan, Z., & Xu, Y. (2023). Prognostics of battery capacity based on charging data and data-driven methods for on-road vehicles. *Applied Energy*, *339*, 120954.
55. https://github.com/Vicomtech/d-EVD_dual-Electric-Vehicle-Dataset?utm_source=chatgpt.com
56. https://www.weatherapi.com/
57. https://www.kaggle.com/datasets/datasetengineer/eviot-predictivemaint-dataset
58. Lang, M., & Duggan, J. (2001). A tool to support collaborative software requirements management. Requirements Engineering, 6(3), 161-172.
59. Badii, A. (2008). User-intimate requirements hierarchy resolution framework (UI-REF)
60. Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. AI Perspectives, 2(1), 1.

Appendix A: State of the art

Category	Tool/Technique	Brief Description
AI-enabled Code Analysis and Understanding	CodeBERT	A pre-trained model for programming and natural languages that encodes both into a unified semantic space for code tasks.
	DeepCode	A system that applies feedback-guided synthesis to generate executable code from test cases.
	Code2Vec	Learns distributed representations of code by extracting abstract syntax trees (ASTs) for tasks like summarization and classification.
Assisted Code Generation	Copilot	Integrates large language models into development environments to suggest code from natural language prompts.
	Amazon CodeWhisperer	Provides real-time code recommendations and includes a unique security scanning feature to identify vulnerabilities.
	Claude Code	An agentic development tool that operates directly in the terminal to understand and execute commands on a codebase.
	AlphaCode	Solves competition-level programming challenges by applying reinforcement learning.
	TransCoder	A source-to-source translator that performs unsupervised translation between programming languages.
	CodeNet	A large-scale dataset of 14 million code samples across over 50 languages for training and benchmarking models.
	Codex	A model demonstrating strong performance in code generation, translation, refactoring, and bug detection.
AI-enabled Reverse Execution and Generation of Pseudocode	Instruction-Level Reverse Execution	An early technique enabling a program to return to a previous state at the machine instruction level for debugging.
	RAMSES	A framework that models agent-based simulations in parallel by exploiting reversibility and speculation.
	Reversible Computation in Robotics	Applies reversible computation to industrial assembly operations to enhance robustness.
	ReVA	An AI assistant for reverse engineering tasks.
	Pseudo-Code Execution & Verification	A method using large language models for static analysis via pseudocode execution and verification.

	Retrieval-Based Transformer Pseudocode Gen	A technique that combines retrieval and generation to create human-readable pseudocode from executable code.
	RETracer & REPT	Systems that reconstruct execution histories from memory dumps or traces to debug production software failures.
	AI-Aided RE of Protected SW	Automates the analysis of protected software using machine learning and symbolic execution.
Requirements and Design Automation	Agile Requirements Designer	Assists in documenting and managing requirements by representing them as mathematically precise visual flows.
	DOORS	A tool for documenting and managing requirements.
	JAMA	A tool that assists in documenting and managing requirements.
	Zephyr	An early tool for requirements engineering that imposes assumptions on input language.
	Balsamiq & Adobe XD	Wireframing tools that generate visual prototypes from requirement descriptions.
	ML for Design Pattern Detection	An approach that applies machine learning to detect design patterns in source code with high accuracy.
Code and Trace Analysis	Statistical Program Slicing	A hybrid static-dynamic technique that isolates relevant portions of execution with high accuracy and low overhead.
	Dynamic Network Slicing	Applies slicing concepts to large-scale, adaptive networking environments.
	Automatic Trace Generation	Supports the creation and analysis of execution traces for tasks like fault localization.
AI in Project Management	AI-Powered PM Tools	Tools that leverage machine learning to analyze historical project data for more accurate predictions on timelines, resources, and risk.
	Atlassian Intelligence	An AI-powered feature for Jira that automates project plans and flags bottlenecks.
	Microsoft Copilot for Project	An AI tool that generates detailed project plans and identifies dependencies from natural language requirements.
AI-Driven Software Testing and Quality Assurance	AI-Enabled Automated Testing	Frameworks that generate and prioritize test cases, analyze results, and provide intelligent feedback.
	DeepXplore	An automated white-box testing framework for deep learning systems.
	DeepCov & DeepHunter	Coverage-guided fuzz testing frameworks for deep neural networks (DNNs).
	AFL++	A state-of-the-art fuzzing tool that extends coverage-guided fuzzing to broader software ecosystems.
	OSS-Fuzz	A continuous fuzzing service for open-source software.
	KLEE & Angr	Tools that use symbolic execution to generate high-quality test inputs and analyze binaries.
Data QA & Integrity Maintenance	Sogeti Quality AI Framework (QAIF)	A framework providing structured guidance for developing and deploying trustworthy, ethical, and robust AI solutions.

Appendix B: Preliminary List of Functional Requirements (FRs)

KIO1

Information	Specifications
Related KIO	KIO1
Requirement Name	Unified KIO Integration Platform
Requirement Type	Functional
Requirement ID Nr.	FR-KIO1-01
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO1
Requirement Name	Front-End GUI for Lifecycle Interaction
Requirement Type	Functional
Requirement ID Nr.	FR-KIO1-02
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO1
Requirement Name	DevOps Workflow Orchestration
Requirement Type	Functional
Requirement ID Nr.	FR-KIO1-03
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO1
Requirement Name	Containerized Service Deployment
Requirement Type	Functional
Requirement ID Nr.	FR-KIO1-04
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO1
Requirement Name	Horizontal LLM Integration
Requirement Type	Functional
Requirement ID Nr.	FR-KIO1-05

Priority	Mandatory
Rationale	Core functionality

KIO2

Information	Specifications
Related KIO	KIO2
Requirement Name	Source code language support (One language for PoC)
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-01
Priority	Mandatory
Rationale	Essential for diagnostic workflow

Information	Specifications
Related KIO	KIO2
Requirement Name	Trace Capture & Replay to generate reversible trace data
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-02
Priority	Mandatory
Rationale	Essential for diagnostic workflow

Information	Specifications
Related KIO	KIO2
Requirement Name	AI-assisted replay simulators that reconstruct previous states from logged diffs
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-03
Priority	Mandatory
Rationale	Essential for diagnostic workflow

Information	Specifications
Related KIO	KIO2
Requirement Name	Dynamic slicing to identify which statements and data dependencies contributed to a given program behaviour
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-04
Priority	Mandatory
Rationale	Essential for diagnostic workflow

Information	Specifications
Related KIO	KIO2
Requirement Name	AI Fault Localization to train from replayed traces and slice context
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-05
Priority	Mandatory
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO2
Requirement Name	The system shall employ a specialised Large Language Model (LLM) fine-tuned for bug detection to automatically identify, classify, and prioritise software faults from source code and execution trace
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-06
Priority	Optional
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO2
Requirement Name	The system shall automatically generate, rank, and validate fix suggestions for detected faults using AI/LLM-based reasoning, ensuring each proposed patch compiles successfully, preserves intended functionality, and provides an explainable rationale for developer review.
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-07
Priority	Optional
Rationale	Value-add feature

Information	Specifications
Related KIO	KIO2
Requirement Name	Thread-aware trace capture and replay simulation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO2-08
Priority	Desired
Rationale	Value-add feature

KIO3

Information	Specifications
Related KIO	KIO3
Requirement Name	Parse and extract key intent and actors from natural language
Requirement Type	Functional
Requirement ID Nr.	FR-KIO3-01
Priority	Mandatory
Rationale	Core outcomes

Information	Specifications
Related KIO	KIO3
Requirement Name	Generate formalized user stories using a trained LLM
Requirement Type	Functional
Requirement ID Nr.	FR-KIO3-02
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO3
Requirement Name	Attach acceptance criteria to each story
Requirement Type	Functional
Requirement ID Nr.	FR-KIO3-05
Priority	Mandatory
Rationale	Core outcomes

Information	Specifications
Related KIO	KIO3
Requirement Name	Allow post-generation editing by users
Requirement Type	Functional
Requirement ID Nr.	FR-KIO3-08
Priority	Desirable
Rationale	Improves quality

Information	Specifications
Related KIO	KIO3
Requirement Name	Detect and highlight ambiguous or vague input
Requirement Type	Functional
Requirement ID Nr.	FR-KIO3-02
Priority	Desirable

Rationale	Improves quality

Information	Specifications
Related KIO	KIO3
Requirement Name	Automated wireframe generation
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO3-04)
Priority	Desirable
Rationale	Productivity booster

Information	Specifications
Related KIO	KIO3
Requirement Name	Export functionality for project artifacts
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO3-07)
Priority	Mandatory
Rationale	Enables interoperability

KIO4

Information	Specifications
Related KIO	KIO4
Requirement Name	Parse formal requirements and identify architectural implications
Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-02
Priority	Mandatory
Rationale	Core deliverable

Information	Specifications
Related KIO	KIO4
Requirement Name	Suggest high-level architecture
Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-06
Priority	Mandatory
Rationale	Core deliverable

Information	Specifications
Related KIO	KIO4
Requirement Name	Generate UML diagrams

Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-05
Priority	Desirable
Rationale	Productivity booster

Information	Specifications
Related KIO	KIO4
Requirement Name	Provide traceability matrix between requirements and design components
Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-03
Priority	Desirable
Rationale	Improves usability

Information	Specifications
Related KIO	KIO4
Requirement Name	Allow user interaction to accept, modify, or reject suggestions
Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-06
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO4
Requirement Name	Export project scaffolding/code skeletons based on architecture
Requirement Type	Functional
Requirement ID Nr.	FR-KIO4-10
Priority	Desirable
Rationale	Productivity booster

Information	Specifications
Related KIO	KIO4
Requirement Name	Architectural Rationale Documentation and Attachment
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO4-07)
Priority	Desirable
Rationale	General improvement

Information	Specifications

Related KIO	KIO4
Requirement Name	Formal requirements ingestion
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO4-01)
Priority	Mandatory
Rationale	Defines component input

Information	Specifications
Related KIO	KIO4
Requirement Name	Architectural component export for tool integration
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO4-08)
Priority	Mandatory
Rational	Enables interoperability

KIO5

Information	Specifications
Related KIO	KIO5
Requirement Name	Generate compliant synthetic traffic at flow level
Requirement Type	Functional
Requirement ID Nr.	FR-KIO5-07
Priority	Mandatory
Rationale	Core deliverable

Information	Specifications
Related KIO	KIO5
Requirement Name	Generate synthetic traffic representing attacks at the flow level
Requirement Type	Functional
Requirement ID Nr.	FR-KIO5-09
Priority	Mandatory
Rationale	Core deliverable

Information	Specifications
Related KIO	KIO5
Requirement Name	Merge GIS layers from different sources to create a new synthetic map representing energy consumption
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT FR-KIO5-01
Priority	Mandatory

Rationale	Core deliverable

Information	Specifications
Related KIO	KIO5
Requirement Name	Generate synthetic GIS layers to create new synthetic maps representing non-existing energy consumption scenarios
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT FR-KIO5-02
Priority	Optional
Rationale	Feature for later phase

Information	Specifications
Related KIO	KIO5
Requirement Name	Generate a master medical datasets by merging together datasets by creating synthetic link between the entries
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT FR-KIO5-03
Priority	Mandatory
Rationale	Core deliverable

Information	Specifications
Related KIO	KIO5
Requirement Name	Propose validation methods for each generation task
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT FR-KIO5-04
Priority	Mandatory
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO5
Requirement Name	Report generation for generation tasks
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT FR-KIO5-05
Priority	Desirable
Rationale	Essential operational quality

KIO6

Information	Specifications
Related KIO	KIO6

Requirement Name	Comparison between input data and properties schema
Requirement Type	Functional
Requirement ID Nr.	FR-KIO6-?
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO6
Requirement Name	Automated data preprocessing
Requirement Type	Functional
Requirement ID Nr.	FR-KIO6-06
Priority	Mandatory
Rationale	Core checks

Information	Specifications
Related KIO	KIO6
Requirement Name	Report generation for data validation, statistical properties, code errors, aggregate data quality
Requirement Type	Functional
Requirement ID Nr.	FR-KIO6-07
Priority	Desirable
Rationale	Maintainability/usability

Information	Specifications
Related KIO	KIO6
Requirement Name	Data preparation for LLM
Requirement Type	Functional
Requirement ID Nr.	FR-KIO6-08
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO6
Requirement Name	User Interface to display the tool outputs, the different steps' details and to edit the data schema
Requirement Type	Functional
Requirement ID Nr.	FR-KIO6-03
Priority	Desirable
Rationale	Improves usability

KIO7

Information	Specifications
Related KIO	KIO7
Requirement Name	Accept prompts and generate code
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-05
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	Retrieve relevant code patterns or documentation (RAG)
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-03
Priority	Desirable
Rationale	Maintainability/usability

Information	Specifications
Related KIO	KIO7
Requirement Name	Fine-tune LLMs
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-02
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	Debug code and return annotated diagnostics and fixes
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-06
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	Auto-generate and execute test cases
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-11
Priority	Desirable

Rationale	Enhanced capabilities

Information	Specifications
Related KIO	KIO7
Requirement Name	Suggest performance or security improvements
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-07
Priority	Desirable
Rationale	Enhanced capabilities

Information	Specifications
Related KIO	KIO7
Requirement Name	Support multiple programming languages and frameworks
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-05
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	Fine-tuned model checkpoint management
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO7-09)
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	Training data ingestion
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO7-01)
Priority	Mandatory
Rationale	Enables fine-tuning process

Information	Specifications
Related KIO	KIO7
Requirement Name	Fine-tuned model export
Requirement Type	Functional
Requirement ID Nr.	NEW REQUIREMENT (FR-KIO7-08)

Priority	Mandatory
Rationale	Enables MLOps integration

Information	Specifications
Related KIO	KIO7
Requirement Name	Feedback response to toolchains
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-26
Priority	Mandatory
Rationale	Key part of the workflow

Information	Specifications
Related KIO	KIO7
Requirement Name	High-Level Synthesis (HLS) Programming Support
Requirement Type	Functional
Requirement ID Nr.	FR-KIO7-25
Priority	Mandatory
Rationale	Support to LLM code generation

KIO8

Information	Specifications
Related KIO	KIO8
Requirement Name	Code optimisation depending on the specified architectures and softwares
Requirement Type	Functional
Requirement ID Nr.	FR-KIO8-01
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO8
Requirement Name	Software reconfiguration recommendations depending on the optimised code and the choosen architecture
Requirement Type	Functional
Requirement ID Nr.	FR-KIO8-02
Priority	Mandatory
Rationale	General improvement

Information	Specifications

Related KIO	KIO8
Requirement Name	Report of the identified optimisation opportunities and the proposed changes
Requirement Type	Functional
Requirement ID Nr.	FR-KIO8-03
Priority	Desirable
Rationale	Maintainability/usability

Information	Specifications
Related KIO	KIO8
Requirement Name	Possibility to change manually some unwanted optimisations
Requirement Type	Functional
Requirement ID Nr.	FR-KIO8-04
Priority	Optional
Rationale	General improvement

Information	Specifications
Related KIO	KIO8
Requirement Name	Ensure iterative code generation via interaction with toolchain
Requirement Type	Functional
Requirement ID Nr.	FR-KIO8-10
Priority	Mandatory
Rationale	Core functionality

KIO9

Information	Specifications
Related KIO	KIO9
Requirement Name	Hash-based decentralised code integrity and scecurity check mechanism to identify code manipulations and preventing untrusted sources
Requirement Type	Functional
Requirement ID Nr.	FR-KIO09-01, FR-KIO9-04, FR-KIO9-05
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO9
Requirement Name	Multifactor Authentication Mechanism for users' identity management
Requirement Type	Functional
Requirement ID Nr.	FR-KIO9-02, FR-KIO9-06, FR-KIO9-07

Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO9
Requirement Name	End-to-end security enabling encrypted storage or data transmission (e.g. against MitM)
Requirement Type	Functional
Requirement ID Nr.	FR-KIO9-03, FR-KIO9-04, FR-KIO9-08
Priority	Mandatory
Rationale	Security baseline

KIO10

Information	Specifications
Related KIO	KIO10
Requirement Name	Execute Machine Learning Tasks on Resource-Constrained Devices
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-01
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Enable ML Execution Using TinyML
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-02
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Use Task-Specific LLMs for On-Device Analytics
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-03
Priority	Optional
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10

Requirement Name	Perform Analytics Directly on Edge Devices
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-04
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Apply Model-Driven Software Engineering Principles
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-05
Priority	Mandatory
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO10
Requirement Name	Address Complex Design Challenges with Domain-Specific Modeling
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-06
Priority	Mandatory
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO10
Requirement Name	Address Complex Validation Challenges with Domain-Specific Modeling
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-07
Priority	Desirable
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO10
Requirement Name	Generate Code for Diverse IoT Platforms
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-08
Priority	Optional
Rationale	Improves speed or convenience

Information	Specifications

Related KIO	KIO10
Requirement Name	Support Deployment on Diverse IoT Platforms
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-09
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Support Deployment on Ultra-Low-Power Microcontrollers
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-10
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Enable Battery Remaining Useful Life Prediction Use Case
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-11
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Integrate with TensorFlow Lite for ML Operations
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-12
Priority	Mandatory
Rationale	Core functionality

Information	Specifications
Related KIO	KIO10
Requirement Name	Integrate with Microcontroller Variant of TensorFlow Lite
Requirement Type	Functional
Requirement ID Nr.	FR-KIO10-13
Priority	Mandatory
Rationale	Core functionality

KIO11

Information	Specifications
Related KIO	KIO11
Requirement Name	Automated self-healing testing to improve context awareness via contextual AI models, LLMs
Requirement Type	Functional
Requirement ID Nr.	FR-KIO11-01
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO11
Requirement Name	AI-driven test case generation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO11-02
Priority	Mandatory
Rationale	Core automation capability

Information	Specifications
Related KIO	KIO11
Requirement Name	Refinement of test cases (prioritation and redundancy detection, prunin if needed)
Requirement Type	Functional
Requirement ID Nr.	FR-KIO11-03
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO11
Requirement Name	Automated test script generation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO11-04
Priority	Mandatory
Rationale	Core automation capability

Information	Specifications
Related KIO	KIO11
Requirement Name	Visualisation of TAT results (GUI)
Requirement Type	Functional
Requirement ID Nr.	FR-KIO11-05

Priority	Mandatory
Rationale	Efficiency and insight

KIO12

Information	Specifications
Related KIO	KIO12
Requirement Name	AI-based security check mechanism and vulnerability analysis to identify MitM, DoS, SQL-injection, XSS, phishing and authentication related attacks
Requirement Type	Functional
Requirement ID Nr.	FR-KIO12-01, FR-KIO12-04, FR-KIO12-05
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO12
Requirement Name	Security test scenario generation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO12-03
Priority	Mandatory
Rationale	Core security functionality

Information	Specifications
Related KIO	KIO12
Requirement Name	End-to-end security enabling encrypted storage or data transmission + multifactor authentication
Requirement Type	Functional
Requirement ID Nr.	FR-KIO12-10
Priority	Mandatory
Rationale	Core security functionality

KIO13

Information	Specifications
Related KIO	KIO13
Requirement Name	Ontological Input Handling
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-01
Priority	Mandatory
Rationale	Enables structured ontological input for precise requirements framing

Information	Specifications
Related KIO	KIO13
Requirement Name	Requirements Derivation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-02
Priority	Mandatory
Rationale	Core to deriving requirements from ontological frameworks

Information	Specifications
Related KIO	KIO13
Requirement Name	Ontology Building Tools
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-03
Priority	Mandatory
Rationale	Core to creating ontological structures for training

Information	Specifications
Related KIO	KIO13
Requirement Name	Visualization Generation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-04
Priority	Mandatory
Rationale	Facilitates comprehension of complex ontological structures through visual aids

Information	Specifications
Related KIO	KIO13
Requirement Name	Training Content Delivery
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-05
Priority	Mandatory
Rationale	Ensures accessible and engaging delivery of training content to maximize learning outcomes

Information	Specifications
Related KIO	KIO13
Requirement Name	Prompt Skills Training
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-06

Priority	Mandatory
Rationale	Equips developers with advanced prompt engineering skills to leverage LLMs effectively

Information	Specifications
Related KIO	KIO13
Requirement Name	Output Evaluation
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-07
Priority	Mandatory
Rationale	Enhances learning depth and breadth

Information	Specifications
Related KIO	KIO13
Requirement Name	Pipeline Integration
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-08
Priority	Desirable
Rationale	Enhances learning depth and breadth

Information	Specifications
Related KIO	KIO13
Requirement Name	Progress Monitoring
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-09
Priority	Mandatory
Rationale	Community feature for later phase

Information	Specifications
Related KIO	KIO13
Requirement Name	Impact Reporting
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-10
Priority	Desirable
Rationale	Enhances learning depth and breadth

Information	Specifications
Related KIO	KIO13
Requirement Name	Interactive Learning Modules

Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-11
Priority	Mandatory
Rationale	Support interactive simulations for hands-on prompt engineering practice

Information	Specifications
Related KIO	KIO13
Requirement Name	Full user companion guide
Requirement Type	Functional
Requirement ID Nr.	FR-KIO13-12
Priority	Mandatory
Rationale	Provides comprehensive guidance on configuration, integration, and prompt engineering to ensure effective use of the AI-powered training platform, enhancing developer productivity and adoption

Appendix C: Preliminary List of Non-Functional Requirements (NFRs)

KIO1

Information	Specifications
KIO	KIO1
Requirement Name	Modular Architecture Design
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-01
Priority	Mandatory
Rationale	Critical quality attribute for enterprise adoption

Information	Specifications
Related KIO	KIO1
Requirement Name	Flexibility in Workflow Configuration
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-02
Priority	Mandatory
Rationale	Critical quality attribute for enterprise adoption

Information	Specifications
Related KIO	KIO1
Requirement Name	Scalability for Integrated Services
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-03
Priority	Mandatory
Rationale	Critical quality attribute for enterprise adoption

Information	Specifications
Related KIO	KIO1
Requirement Name	Interoperability with External Systems
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-04
Priority	Desirable
Rationale	Critical quality attribute for enterprise adoption

Information	Specifications
Related KIO	KIO1
Requirement Name	Security in Integration Layer
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-05
Priority	Mandatory

Rationale	Ensures secure integration of services to protect sensitive data and maintain trust

Information	Specifications
Related KIO	KIO1
Requirement Name	System Reliability
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-06
Priority	Mandatory
Rationale	System must maintain 99.9% uptime to support continuous DevOps workflows

Information	Specifications
Related KIO	KIO1
Requirement Name	GDPR and Ethics Compliance
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO1-07
Priority	Mandatory
Rationale	Ensures compliance with data protection and ethical AI standards

KIO2

Information	Specifications
Related KIO	KIO2
Requirement Name	Performance: Quick processing for common operations
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO2-01
Priority	Desirable
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO2
Requirement Name	Quality: Better accuracy
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO2-02
Priority	Desirable
Rationale	Quality baseline

Information	Specifications

Related KIO	KIO2
Requirement Name	Modularity: Updatable structure
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO2-03
Priority	Optional
Rationale	Aids adoption and maintenance

KIO3

Information	Specifications
Related KIO	KIO3
Requirement Name	Performance: fast response to request
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO3-06
Priority	Desirable
Rationale	Enhances experience

Information	Specifications
Related KIO	KIO3
Requirement Name	Scalability: handle concurrent requests
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO3-02
Priority	Desirable
Rationale	Enhances experience

Information	Specifications
Related KIO	KIO3
Requirement Name	Accuracy: no ambiguity or contradiction
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO3-04
Priority	Mandatory
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO3
Requirement Name	Security: securely stored data
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO3-05
Priority	Mandatory

Rationale	Trust and confidentiality required

Information	Specifications
Related KIO	KIO3
Requirement Name	Artifact versioning support
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO3-10)
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO3
Requirement Name	Graceful error detection and recovery
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO3-07)
Priority	Mandatory
Rationale	Ensures system stability

Information	Specifications
Related KIO	KIO3
Requirement Name	Usability: intuitive UI
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO3-03
Priority	Desirable
Rationale	Enhances experience

KIO4

Information	Specifications
Related KIO	KIO4
Requirement Name	Performance: fast response to request
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-06
Priority	Desirable
Rationale	Enhances efficiency

Information	Specifications
Related KIO	KIO4
Requirement Name	Scalability: batch processing

Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-02
Priority	Desirable
Rationale	Enhances efficiency

Information	Specifications
Related KIO	KIO4
Requirement Name	Accuracy: high relevance of suggested architecture
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-04
Priority	Mandatory
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO4
Requirement Name	Security: securely stored data
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-05
Priority	Mandatory
Rationale	Required for correctness and integration

Information	Specifications
Related KIO	KIO4
Requirement Name	Usability: intuitive interface
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-03
Priority	Desirable
Rationale	Enhances efficiency

Information	Specifications
Related KIO	KIO4
Requirement Name	Architectural component versioning support
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO4-09)
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO4

Requirement Name	Graceful error detection and recovery
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO4-08)
Priority	Mandatory
Rationale	Enables interoperability

Information	Specifications
Related KIO	KIO4
Requirement Name	Compliance: support for compatible output formats
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO4-11
Priority	Mandatory
Rationale	Required for correctness and integration

KIO5

Information	Specifications
Related KIO	KIO5
Requirement Name	Performance: fast Generation, low latency for data interactions
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO5-02
Priority	Desirable
Rationale	Essential operational quality

Information	Specifications
Related KIO	KIO5
Requirement Name	Scalability: handle wide inputs and generate large synthetic datasets / multithreading for multi-GPU trainings
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO5-04
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO5
Requirement Name	Accuracy: synthetic data matches input data
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO5-01
Priority	Mandatory
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO5
Requirement Name	Security: data protection, injection protections
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO5-03
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO5
Requirement Name	Usability: intuitive UI
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO5-05
Priority	Desirable
Rationale	Improves usability

KIO6

Information	Specifications
Related KIO	KIO6
Requirement Name	Performance: optimised computation of statistic properties, fast validation and preprocessing of data
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-01
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO6
Requirement Name	Scalability: must support large number of data and schema
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-04
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO6
Requirement Name	Reliability: error detection, analysis and validation procedures and reports must be accurate

Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-02
Priority	Mandatory
Rationale	Quality baseline

Information	Specifications
Related KIO	KIO6
Requirement Name	Sustainability: code must be documented and easy to update/debbug
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-06
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO6
Requirement Name	Transparency: the user must be able to retrieve logs and observe the different steps of the preparation and validation of data
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-05
Priority	Desirable
Rationale	Maintainability/usability

Information	Specifications
Related KIO	KIO6
Requirement Name	Ergonomic/user-friendly: UI must be intuitive and reports easily readable
Requirement Type	Non-Functional
Requirement ID Nr.	FR-KIO6-03
Priority	Desirable
Rationale	Improves usability

Information	Specifications
Related KIO	KIO6
Requirement Name	Security: data must be securely stored
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO6-03
Priority	Mandatory
Rationale	Security baseline

KIO7

Information	Specifications
Related KIO	KIO7
Requirement Name	Performance: fast code generation and debugging
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-06
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO7
Requirement Name	Accuracy: High code correctness
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-09
Priority	Mandatory
Rationale	Operational baseline

Information	Specifications
Related KIO	KIO7
Requirement Name	Scalability: parallel inference and fine-tuning tasks
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-02
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO7
Requirement Name	Security: isolated execution for debugging, no execution of arbitrary code without sandboxing
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-12
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO7
Requirement Name	Usability: Intuitive UI
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-03

Priority	Desirable
Rationale	Improves usability

Information	Specifications
Related KIO	KIO7
Requirement Name	Maintainability: versioning for prompts, fine-tuned models and output code
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO7-13
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO7
Requirement Name	High code coverage test generation
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO7-10)
Priority	Mandatory
Rationale	Guarantees code reliability

Information	Specifications
Related KIO	KIO7
Requirement Name	Graceful error detection and recovery
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO7-07)
Priority	Mandatory
Rationale	Esnsures system stability

Information	Specifications
Related KIO	KIO7
Requirement Name	Coding standard and best practice compliance
Requirement Type	Non-Functional
Requirement ID Nr.	NEW REQUIREMENT (NFR-KIO7-11)
Priority	Desirable
Rationale	General improvement

KIO8

Information	Specifications
Related KIO	KIO8

Requirement Name	Performance : fast and efficient in terms of resources
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO8-01
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO8
Requirement Name	Scalability : the tool must be able to perform optimisations for large code bases
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO8-02
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO8
Requirement Name	Sustainability : the tool must be documented and easy to update/debbug
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO8-03
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO8
Requirement Name	Transparency : the user must be able to retrieve information about the optimisations made by the tool
Requirement Type	Non-Functional
Requirement ID Nr.	FR-KIO8-03
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO8
Requirement Name	Ergonomic/user-friendly : UI must be intuitive and reports easily readable
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO8-04
Priority	Desirable
Rationale	Improves usability

Information	Specifications
Related KIO	KIO8
Requirement Name	Secure: data must be securely stored
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO8-?
Priority	Mandatory
Rationale	Improves usability

KIO9

Information	Specifications
Related KIO	KIO09
Requirement Name	Performance: fast authentication and efficient data transmission, downloading and reporting in terms of resources
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-01
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO09
Requirement Name	Scalability: should support a decentralised network that can be used by huge development teams connecting to the system from different end points
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-02
Priority	Desirable
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO09
Requirement Name	Sustainability: Energy-efficient decentralised networks and tools should be selected, easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-03
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO09
Requirement Name	Transparency: inputs and outputs should be transparent to the user as the compliance check should be conducted at backend
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-04
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO09
Requirement Name	Ergonomic/user-friendly: the compliance check analysis must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-05
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO09
Requirement Name	Security: end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO09-06
Priority	Mandatory
Rationale	Security baseline

KIO10

Information	Specifications
Related KIO	KIO10
Requirement Name	Reduce Energy Consumption by Limiting Cloud Data Transfer
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-01
Priority	Mandatory
Rationale	Core functionality for energy efficiency

Information	Specifications
Related KIO	KIO10
Requirement Name	Minimize Energy Needs via On-Device ML Tasks
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-02
Priority	Desirable
Rationale	Improves speed or convenience

Information	Specifications
Related KIO	KIO10
Requirement Name	Mitigate AI's Contribution to Global Electricity Usage
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-03
Priority	Optional
Rationale	Reduces AI's environmental impact, aligning with global sustainability goals

Information	Specifications
Related KIO	KIO10
Requirement Name	Mitigate AI's Contribution to Global CO2 Emissions
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-04
Priority	Mandatory
Rationale	Core functionality for sustainability

Information	Specifications
Related KIO	KIO10
Requirement Name	Promote Sustainability in Low-Power AI Development
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-05
Priority	Mandatory
Rationale	Important for future expansion and regulatory compliance

Information	Specifications
Related KIO	KIO10
Requirement Name	Uphold Privacy Through On-Device Processing
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-06
Priority	Mandatory
Rationale	Core functionality for data protection

Information	Specifications
Related KIO	KIO10
Requirement Name	Uphold Security Through On-Device Processing
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-07
Priority	Mandatory
Rationale	Core functionality for secure operations

Information	Specifications
Related KIO	KIO10
Requirement Name	Enhance ML Performance Across Network Conditions
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-08
Priority	Mandatory
Rationale	Ensures robust ML execution

Information	Specifications
Related KIO	KIO10
Requirement Name	Enhance Service Quality Across Network Conditions
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-09
Priority	Mandatory
Rationale	Ensures consistent service quality across diverse network conditions

Information	Specifications
Related KIO	KIO10
Requirement Name	Provide Multi-Level Abstraction in Software Models
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-10
Priority	Mandatory
Rationale	Enables flexible development through multi-level abstraction

Information	Specifications
Related KIO	KIO10
Requirement Name	Support Abstraction from Platform-Independent Models
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-11

Priority	Mandatory
Rationale	Supports platform-agnostic development

Information	Specifications
Related KIO	KIO10
Requirement Name	Support Abstraction to Platform-Specific Models
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-12
Priority	Mandatory
Rationale	Enables tailored deployment for specific platforms

Information	Specifications
Related KIO	KIO10
Requirement Name	Facilitate Robust Development Through Multi-Level Abstraction
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-13
Priority	Mandatory
Rationale	Ensures robust development processes

Information	Specifications
Related KIO	KIO10
Requirement Name	Facilitate Robust Validation Through Multi-Level Abstraction
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-14
Priority	Optional
Rationale	Complement sustainability by ensuring robust validation

Information	Specifications
Related KIO	KIO10
Requirement Name	Energy Consumption Metrics
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO10-15
Priority	Mandatory
Rationale	System must reduce energy consumption by at least 30-50% compared to cloud-based ML solutions

KIO11

Information	Specifications
Related KIO	KIO11

Requirement Name	Performance
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-01
Priority	Mandatory
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO11
Requirement Name	Scalability: solution must be used by huge development teams connecting to the system from different end points
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-02
Priority	Mandatory
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO11
Requirement Name	Sustainability: easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-03
Priority	Desirable
Rationale	Security baseline

Information	Specifications
Related KIO	KIO11
Requirement Name	Transparency: inputs and outputs should be transparent to the user as the TAT should be conducted at backend
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-04
Priority	Mandatory
Rationale	General improvement

Information	Specifications
Related KIO	KIO11
Requirement Name	Ergonomic/user-friendly: the TAT must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-05

Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO11
Requirement Name	Security: Cyber coverage including MitM, DoS, SQL-injection, XSS, phishing and authentication related attacks; end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO11-06
Priority	Mandatory
Rationale	Security baseline

KIO12

Information	Specifications
Related KIO	KIO12
Requirement Name	Performance: accurate test scenario generation and vulnerability detection
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO12-01
Priority	Mandatory
Rationale	High precision essential

Information	Specifications
Related KIO	KIO12
Requirement Name	Scalability: should support adding new types of vulenrabiltiies or attack types depending on the development; solution must be used by huge development teams connecting to the system from different end points
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO12-02
Priority	Mandatory
Rationale	Performance or scale attribute

Information	Specifications
Related KIO	KIO12
Requirement Name	Sustainability: easy documentation, lightweight/heavyweight security depending on the criticality of the development (e.g., top-secret, commercial, or toy projects)

Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO12-03
Priority	Desirable
Rationale	Security baseline

Information	Specifications
Related KIO	KIO12
Requirement Name	Transparency: inputs and outputs should be transparent to the user as the vulnerability check and test scenario generations should be conducted at backend
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO12-04
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO12
Requirement Name	Ergonomic/user-friendly: the test generation and vulnerability analysis must be self-explainable, highlights with color codes and short recommendations/alerts orienting the user
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO12-05
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO12
Requirement Name	Security: Cyber coverage including MitM, DoS, SQL-injection, XSS, phishing and uathentication related attacks; N13end-to-end secure data transmission, storage and multifactor authentication must be provided; lightweight/heavyweight security depending on the criticality of the development (e.g., topsecret, commercial, or toy projects)
Requirement Type	Non-Functional
Requirement ID Nr.	FR-KIO12-01, FR-KIO12-04, FR-KIO12-05
Priority	Mandatory
Rationale	Security baseline

KIO13

Information	Specifications
Related KIO	KIO13

Requirement Name	Interface Usability
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-01
Priority	Mandatory
Rationale	Service baseline & legal compliance

Information	Specifications
Related KIO	KIO13
Requirement Name	Accessibility Standards
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-02
Priority	Mandatory
Rationale	Ensures accessibility for diverse users, complying with legal standards (e.g., WCAG 2.1)

Information	Specifications
Related KIO	KIO13
Requirement Name	Response Time
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-03
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO13
Requirement Name	User Capacity Handling
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-04
Priority	Mandatory
Rationale	Security baseline

Information	Specifications
Related KIO	KIO13
Requirement Name	System Availability
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-05
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO13
Requirement Name	Data Protection
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-06
Priority	Mandatory
Rationale	Protects user data to maintain trust and comply with GDPR

Information	Specifications
Related KIO	KIO13
Requirement Name	Tool Compatibility
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-07
Priority	Desirble
Rationale	Additional capability

Information	Specifications
Related KIO	KIO13
Requirement Name	Content Update Ease
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-08
Priority	Desirable
Rationale	General improvement

Information	Specifications
Related KIO	KIO13
Requirement Name	Resource Optimization
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-09
Priority	Optional
Rationale	Edge-case support

Information	Specifications
Related KIO	KIO13
Requirement Name	Outcome Tracking
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-10
Priority	Mandatory
Rationale	For explainability

Information	Specifications
Related KIO	KIO13
Requirement Name	Multi-Language Support
Requirement Type	Non-Functional
Requirement ID Nr.	NFR-KIO13-11
Priority	Mandatory
Rationale	Support multiple languages to enhance global accessibility

Appendix D: Interview Form

Interview on AI in Software Development

This short questionnaire is part of the **AI4SWENG project**, designed to gather insights on the role of AI in software development from experienced software enInterview on AI in Software Developineers. Your responses will help us understand the real-world impact of AI on current practices. Your detailed feedback is highly valuable.

1. Years of Experience in Software Development*

- ○ Less than 7
- ○ More than 7

2. What is your level of experience with AI technologies?*

- ○ weak: I have no knowledge of AI at all. I have heard of it, but cannot assess it at all.
- ○ average: I have already got to know AI in the media and have basically dealt with the possibilities.
- ○ good: I have already dealt with AI privately and/or professionally and I know the concepts and possibilities.
- ○ excellent: I can apply the concepts and technologies of AI and have already developed solutions based on AI.

3. Based on your experience, what is the core role of a software developer in an environment that utilizes AI?*

- ○ Proactively championing innovation and change

- ○ Managing and integrating AI-driven tools (programming assistants like GitHub, Copilot, etc)
- ○ Building, monitoring and maintenance of AI-driven systems
- ○ Ensuring the ethical and secure use of AI related to data privacy and bias
- ○ Driving the creation of more sophisticated, human-like software
- ○ Other:

4. In what specific ways does AI influence software development practitioners towards improved software development?*

Type your response here...

5. To what extent do you believe AI improves efficiency in the following phases of the Software Development Life Cycle (SDLC)?*

	NOT AT ALL	MINOR IMPROVEMENT	MODERATE IMPROVEMENT
Requirements gathering and analysis	○	○	○
Design and architecture	○	○	○
Coding and development	○	○	○
Testing and Quality Assurance (QA)	○	○	○
Deployment and CI/CD	○	○	○

	NOT AT ALL	MINOR IMPROVEMENT	MODERATE IMPROVEMENT
Maintenance and support	○	○	○

6. Describe a specific example of how AI has improved your team's software delivery process (e.g., in timeline, quality, or automation). Please be as specific as possible regarding the tools or methods used.*

Type your response here...

7. What are the most significant consequences of enhanced, AI-driven data management on software development?*

Type your response here...