Ref. Ares(2025)10494365 - 30/11/2025

AI Engineering Suite to Support Agile, Efficient Software Engineering

Deliverable D2.6
Functional and Non-Functional Requirements and Use-Case Requirements

Project Number	**101189908**
Project Title	AI Engineering Suite to Support Agile Efficient Software Engineering
Project Acronym	AI4SWEng
Title of Deliverable	Functional and Non-Functional Requirements and Use-Case Requirements
Version	V1.0
Due Date of Delivery	M12
Distribution level	PU

Work Package	**2**
Responsible Partner	INNO
Editor(s)	Marco Tiemann (INNO)
Contributor(s)	Atta Badi (UREAD), Naga Sai Dasari (UREAD), Ryan Faulkner (INNO), Lesley Badii (INNO), all project Partners for requirements, D2.1 and D2.4 Partners for content summarised in Section 2 of the deliverable
Reviewer(s)	ATT, UNIGE, UNIBZ, AI4, ERG
Keywords	Requirements Engineering, System Architecture, Use-Case Definition, Test-Case Definition, Performance Verification

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 deliverable compiles the revised functional and non-functional requirements and use-case specifications for the AI4SWEng project. It builds on the outcomes of tasks T2.1 and T2.4, which inform updates at the Key Innovation and Output (KIO) level. The document introduces the requirements revision process and the technical infrastructure for managing requirements. It presents the updated KIOs and Use-Cases, formally specifies their mapping, and defines test-cases to support verification against Key Performance Indicators. The AI4SWEng framework architecture is outlined, with its building blocks characterised in relation to KIOs. Finally, the deliverable sets out the strategy for performance verification and certification and indicates how subsequent reports will refine and extend it.

Document Revision History

Version	Date	Author	Revision
V0.1	03.10.2025	Marco Tiemann (INNO)	Initial ToC draft
V0.2	07.11.2025	Marco Tiemann, Ryan Faulkner (INNO)	Initial contributions to sections, 1, 2, 3, 4, 5
V0.3	14.11.2025	Marco Tiemann (INNO)	Revision of document structure to reduce the number of main sections
V0.4	24.11.2025	Marco Tiemann (INNO)	Updates to requirements tables using final export from GitHub repository; removal of appendix for requirements due to space considerations
V0.5	26.11.2025	Atta Badii (UREAD)	Addition of Section 2.1
V0.6	26.11.2025	Atta Badii, Naga Sai Dasari (UREAD)	Addition of Sections 5, 6, 7, 8
V1.0	29.11.2025	Marco Tiemann (INNO), Naga Sai Dasari (UREAD), Andrej Aksjonov (AI4)	Revisions to integrate reviewer feedback and preparation of document release version

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 2025 The AI4SWEng Consortium

AI4SWEng Innovation Mission

AI4SWEng

AI Engineering Suite to Support Agile, Efficient Software Engineering

AI4SWEng comprises 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 Systems 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 in 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 reduce worker stress through the pain-points in agile SW development pipelines and higher productivity thus enhanced job satisfaction and creativity. Accordingly, using four Use-Cases the project will develop, deliver and validate 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. A host of tasks will be supported such as code generation, cross-compilation, advanced debugging and dynamic slicing, security-privacy-by-design, model optimisation for reliability, energy efficiency as well as model adaptation and maintenance. This also includes software quality support through data-driven insights for informed design decision-making and strategic software development planning. Additionally, with a commitment to user-centred design, AI4SWEng will provide structured design in advanced prompt engineering to support software developer teams to best exploit the capabilities of the platform.

Table of Contents

1 Introduction 13
1.1 About This Deliverable 13
1.2 Document Structure 14
1.3 Relation to Other Tasks and Deliverables 14
2 Initial Requirements and User Needs Gathering 16
2.1 Methodological Underpinning 16
2.2 Requirements Management Infrastructure 17
2.3 Target User Groups 19
2.4 Types of Actor-System Interactions 20
2.5 KIO Needs and Initial Requirements 21
2.6 Security, Traceability and Privacy-by-Design Requirements 27
3 Revision and Completion of Initial Requirements 29
3.1 Revision and Completion Process 29
3.2 Requirements Format Specification and Revision 30
3.3 Iterative Requirements Refinement 32
3.4 Continuing Iterative Requirements Engineering 33
4 Project KIO and UC Requirements 35
4.1 Key Innovation Outcome Requirements 35
4.2 Use-Case Requirements 63
5 Use-Case Specifications 79
5.1 Introduction 79
5.2 Use-Case Overview and Mapping to KIOs 80
5.3 Detailed Use-Case Scenarios 81
5.4 Cross-Use-Case Observations 100
6 Test-Case Specifications 103
6.1 Test-cases for individual Use-cases 103
6.2 Cross-Use-Case Test Observations 123
7 Framework Architecture Specification 124
7.1 Introduction and High-Level Framework 124
7.2 Framework Architecture Component Diagram 128
7.3 Enterprise Architecture Components 128

7.4 Building Blocks and Key-Components 130
7.5 Cross-Cutting Non-Functional Specifications 133
7.6 Agentic Architecture 135
8 Performance Verification and Certification Strategy 137
8.1 Introduction 137
8.2 Contemporary Landscape of AI Certification and Evaluation Frameworks Relevant to AI-Powered Software Engineering Tools 138
8.3 Certification Frameworks and Implementation Maturity in AI4SWEng 140
8.4 Benchmarking Activities 140
8.5 Certification-by-Design and Standards Compliance Mapping for the AI4SWEng Platform 141
8.6 Stress Testing Strategy 142
8.7 Performance Optimisation 143
8.8 Monitoring and Measurement Framework 144
8.9 Certification Pathways for Classic and Agentic AI 145
9 Conclusions and Future Work 147
9.1 Summary 147
9.2 Upcoming Work and Deliverables 147
References 149

List of Figures

Figure 1: Example task tracking board from AI4SWEng....17
Figure 2: Example requirement entry from KIO 5 task tracking 19
Figure 3: Revision and completion process for requirements 29
Figure 4: Requirements specification in GitHub projects....34
Figure 5: Use-Cases × KIOs Interaction Diagram 80
Figure 6: AI4SWEng Development Methodology: Agile, DevOps, and Responsible AI 125
Figure 8: AI4SWEng System - Framework Architecture....127
Figure 8: GUI & Execution Block....130
Figure 9: AI SysDev Orchestration 131
Figure 10: Requirements Engineering & Planning....131
Figure 12: Core AI Engines....132
Figure 13: Data Management Block....132
Figure 14: Test Automation and DevOps Block 133
Figure 15: Security & Governance Block 133

List of Tables

Table 1: Important types of actor-system interactions 20
Table 2: KIO functional needs identified in D2.4 22
Table 3: KIO non-functional needs identified in D2.4.....24
Table 4: Security, traceability and privacy-by-design requirements from D2.1 27
Table 5: AI4SWeng requirements format 31
Table 6: KIO1 requirements 35
Table 7: KIO2 requirements 37
Table 8: KIO3 requirements 39
Table 9: KIO4 requirements 42
Table 10: KIO5 requirements 46
Table 11: KIO6 requirements 48
Table 12: KIO7 requirements 50
Table 13: KIO8 requirements 54
Table 14: KIO9 requirements 55
Table 15: KIO10 requirements.....57
Table 16: KIO11 requirements.....58
Table 17: KIO12 requirements.....60
Table 18: KIO13 requirements.....62
Table 19: UC1 requirements 65
Table 20: UC2 requirements 71
Table 21: UC3 requirements 73
Table 22: UC4 requirements 74
Table 23: UC5 requirements 75
Table 24: Use-cases to KIOs Mapping.....80
Table 25: UC-UC1-01: Natural-Language Customer Request to Draft Issue and Requirements 81
Table 26: UC-UC1-02: Automated Code and Test Generation from Approved Issue 82
Table 27: UC-UC1-03: Automated Bug Localisation, Fix Suggestion, and Verification 83
Table 28: UC-UC1-04: Full End-to-End Feature Delivery with Compliance and Energy Check 84
Table 29: UC-UC2-01: Medical Report Generation from Patient Notes and Medical Record Data 85
Table 30: UC-UC2-02: Generate Medication Safety Report from Natural Language Patient Query 87
Table 31: UC-UC2-03: Integrate and Harmonise Multiple Open-Source Healthcare Datasets into Unified Master Dataset 88
Table 32: UC-UC3-01: Align the FPGA developer's development task with formal requirements generation of the system 90

Table 33: UC-UC3-02: Generate HLS project 91
Table 34: UC-UC3-03: Human-in-the-loop process to assess and guide code generation 92
Table 35: UC-UC4-01: Perform Exploratory Data Analysis with Early Energy Constraints and HITL Oversight 93
Table 36: UC-UC4-02: Generate Energy-Constrained Federated TinyML Codebase with HITL Refinement 94
Table 37: UC-UC4-03: Deploy and Runtime-Optimise Federated TinyML Firmware with HITL Monitoring 95
Table 38: UC-UC5-01: Generate User Story from Contextual Requirements 96
Table 39: UC-UC5-02: Generate and validate dataset 97
Table 40: UC-UC5-03: Generate Calculation Module codebase 97
Table 41: UC-UC5-04: Generate Unit tests and Functional tests 98
Table 42: UC-UC5-05: Execute pipeline and run functional tests 99
Table 43: TC-UC1-01: Natural-Language Customer Request to Draft Issue and Requirements 103
Table 44: TC-UC1-02: Automated Code and Test Generation from Approved Issue 104
Table 45: TC-UC1-03: Automated Bug Localisation, Fix Suggestion, and Verification 105
Table 46: TC-UC1-04: Full End-to-End Feature Delivery with Compliance and Energy Check 106
Table 47: TC-UC2-01-01: Verify Clinical Entity Extraction and Data Completeness Check 107
Table 48: TC-UC2-01-02: Test Report Generation with Regulatory Compliance 108
Table 49: TC-UC2-02-01: Validate Natural Language Query Parsing and Interaction Analysis 108
Table 50: TC-UC2-03-01: Validate Ingestion and Schema Harmonisation of Heterogeneous Data 109
Table 51: TC-UC2-03-02: Test Synthetic Linkage and Quality Assessment 110
Table 52: TC-UC-UC3-01: Align the FPGA developer's development task with system formal requirements generation. 111
Table 53: TC-UC-UC3-02-01: Generate HLS project - Code generation. 111
Table 54: TC-UC-UC3-02-02: Generate HLS project - Compilation. 112
Table 55: TC-UC-UC3-03: Human-in-the-loop process to assess and guide code generation 113
Table 56: TC-UC-UC4-01: Perform Exploratory Data Analysis with Early Energy Constraints and HITL Oversight 114
Table 57: TC-UC4-01-02: Test Anomaly Detection and HITL Approval in Exploratory Analysis 115
Table 58: TC-UC4-02-01: Verify Architecture Generation and HITL Selection 116
Table 59: TC-UC4-02-02: Test Code Generation, Compliance Validation, and Debugging 116

Table 60: TC-UC4-03-01: Validate Firmware Deployment and Initial Profiling 118
Table 61: TC-UC4-03-02: Test Runtime Optimisation and Anomaly Handling with HITL. 118
Table 62: TC-UC5-01-01: Validate Generated User Story from Contextual Requirements 119
Table 63: TC-UC5-02-01: Test Dataset Generation and Validation 120
Table 64: TC-UC5-03-01: Test Calculation Module Codebase Generation 121
Table 65: TC-UC5-04-01: Test Unit tests and Functional tests Generation 121
Table 66: TC-UC5-05-01: Test pipeline Execution and functional tests run 122
Table 67: Summary of Cross-Cutting Non-Functional Requirements 134
Table 68: Stages of the Agentic Architecture 135

Terms and Abbreviations

Term	Description
ADRA	AI, Data and Robotics Association
AI	Artificial Intelligence
AI4SWEng	AI Engineering Suite to Support Agile Efficient Software Engineering
BPMN	Business Process Model and Notation
CD	Continuous Delivery
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Delivery
CLI	Command Line Interface
DevOps	Development and Operations
DoA	Description of Action
DoS	Denial of Service
E2E	End-to-End
EHR	Electronic Health Records
ESB	Enterprise Service Bus
EV	Electric Vehicle
FPGA	Field Programmable Gate Arrays
FR	Functional Requirement
GDPR	General Data Protection Regulation
GIS	Geographical Information System
GUI	Graphical User Interface
HDL	Hardware Description Language
HIPAA	Health Insurance Portability and Accountability Act (US)
HITL	Human-in-the-Loop
HLS	High-Level Synthesis
IaC	Infrastructure as Code
JDM	Judgement and Decision Making
KGS	Knowledge Graphs
KIO	Key Innovations and Outputs
KPI	Key Performance Indicator
KVKK	Turkish Data Protection Law
LLM	Large Language Model
ML	Machine Learning
MLOps	Machine Learning Operations
NFR	Non-Functional Requirement
NL	Natural Language
PM	Project Manager
RAG	Retrieval-Augmented Generation
RegOps	Regulatory Operations
RESTful	Representational State Transfer (style)
RUL	Remaining Useful Life
SAST	Static Application Security Testing

SDLC	Software Development Life Cycle
RTO	Research and Technology Organisation
SE	Software Engineering
SME	Small and Medium-sized Enterprise
TAT	Test Automation Tool
TC	Test-Case
TinyML	Tiny Machine Learning
UC	Use-Case
UI-REF	User-Intimate Requirements Hierarchy Resolution and Usability Evaluation Framework
V&V	Verification and Validation
VDB	Vector Database
WP	Work Package
XSS	Cross-Site Scripting

Executive Summary

This deliverable, D2.6, reports on the revised functional and non-functional requirements and use-case requirements for the AI4SWEng project. The document summarises the results of T2.1 and T2.4 that are used as input for revising the functional and non-functional requirements at Key Innovations & Outputs (KIO) level. It introduces the process used for revising requirements and describes the technical requirements management infrastructure as implemented in the project. The revised requirements for KIO and Use-Cases are then described.

The project Use-Cases are formally specified and mapped to the project KIOs and corresponding test-cases are defined to facilitate the verification of project results against the defined Key Performance Indicators. The framework architecture for the project is introduced, and the building blocks of the architecture are characterised, underscoring how the KIOs form part of the overall framework architecture. A strategy for performance verification and certification is defined with consideration of relevant standards, regulations and initiatives.

The document concludes by setting out the outlook towards the follow-on deliverables in the project that will update and refine the content presented herewith in this deliverable (D2.6).

1 Introduction

This section introduces the deliverable D2.6, gives a brief overview of the document structure and documents the relation of this document with other tasks and deliverables in AI4SWEng, both as "upstream" input for this deliverable and as a "downstream" resource for ongoing and upcoming activities in the project.

1.1 About This Deliverable

This deliverable constitutes the consolidated outcome of Task T2.5 "Overall Functional and Non-Functional Requirements Specification" and Task T2.6 "Use-Cases, Test-Cases, Performance Verification and Framework Architecture Specification". It delivers the final, conflict-free requirement hierarchy, the formally specified use-cases with associated test criteria, the agentic framework architecture, and the performance verification and certification strategy for the AI4SWEng platform.

All artefacts presented in this document have been produced under the exclusive ontological governance of the User-Intimate Requirements Hierarchy Resolution and Usability Evaluation Framework (UI-REF) [1]. The framework ensures that every requirement, use-case element, architectural decision, and verification criterion remains transparently traceable to the psycho-cognitively rooted and judgement-and-decision-making validated deepest values of all stakeholders and regulatory regimes. By enforcing dynamic conflict resolution at the level of human memory-recall valence and regulatory weight, UI-REF eliminates by design the possibility of latent contradictions or inappropriate emergent behaviour in the resulting system-of-systems platform.

This document reports the systematic refinement and completion of functional and non-functional requirements initiated in the preceding tasks that provided the primary inputs to the present work, namely:

- Task T2.1 Code/Data Provenance-Awareness, Data Generation, Security-Privacy Requirements Engineering.
- Task T2.2 AI-Act and Sustainability and Security by Design Requirements Engineering.
- Task T2.4 AI-Powered Software Development and Testing Tool Functionalities Requirements.

The requirements originating from these tasks, together with the project Description of Action (DoA), have been subjected to the full UI-REF hierarchy-resolution process described in Deliverable D2.4.

Task T2.6 has transformed these requirements into five concrete use-cases that are further elaborated into eighteen detailed use-case scenarios with associated measurable test criteria, an agentic framework architecture, and a performance verification and

certification strategy that jointly guarantee continued compliance with the AI-Act, GDPR, Data-Act, Z-inspection® and emerging European sustainability standards.

1.2 Document Structure

This document is organised as follows:

- **Section 1** (this section) introduces the purpose of this deliverable, its structure and the relationship between this document and other tasks and deliverables in AI4SWEng.
- **Section 2** summarises the requirements engineering activities undertaken in tasks T2.1 and T2.4 of the project which form the initial round of requirements gathering and specification activities. It also presents a concise overview of the project infrastructure used to gather requirements and operationalise them throughout the course of the project.
- **Section 3** details the approach taken for revisions and improvements to arrive at a consolidated and improved set of requirements for use in AI4SWEng.
- **Section 4** presents a summary of the project requirements for each Key Innovation and Output (KIO) and Use-Case (UC).
- **Section 5** details the formal UCs specification for the five use-cases in the project.
- **Section 6** documents the test-cases that have been defined to provide clearly measurable test and validation criteria for the validation of KIOs against criteria defined and measured in the project use-cases.
- **Section 7** presents the framework architecture definition for AI4SWEng including the building blocks and the chosen agentic approach to architectural composition and interaction.
- **Section 8** describes the AI4SWEng performance verification framework that will be applied to measure key performance indicators for the overall platform.
- **Section 9** closes the document with a summary and an outlook towards upcoming work and deliverables.

A references section concludes the document.

1.3 Relation to Other Tasks and Deliverables

Two main groups of relations are important when considering this deliverable: a) tasks and deliverables that serve as input to the present document, and b) tasks and deliverables that will use the output presented in this deliverable as part of their activities in the project.

Tasks T2.1 and T2.4 and the related deliverables D2.1 and D2.4 are the main project tasks that have provided direct input for the work presented in this deliverable: T2.1 addresses important requirements for provenance awareness, data generation, security and privacy. T2.4 completed a full cycle of requirements gathering and formalisation and provided an extensive set of requirements together with initial reviews by representatives of the target

end user group to guide initial refinements. The results from these tasks together with the project Description of Action (DoA) are the main inputs for T2.5 and T2.6 and this deliverable that documents the results of the activities in T2.5 and T2.6.

The requirements, use-case definitions and architecture specification represent the Consortium's best definition of the work that will be performed in the implementation, demonstration and evaluation phases of the project. Given that, the output of T2.5 and T2.6 presented in this deliverable will guide the work that will be carried out in the work packages WP3 "Development of AI-Powered Software Development and Testing Tools (AI-SysDev, TAT)", WP4 "Implementation of AI-Powered Software Development and Testing Automation Framework", WP5 "System Integration & Validation in Use-Cases" and WP6 "Evaluation and Optimisation of Developed AI-Powered Tools Functionalities".

Because the UI-REF-methodology maintains living promotion/demotion pathways with immutable root-rationale traceability, the specifications and descriptions presented in this document are deliberately designed as dynamic, living artefacts rather than immutable prescriptions. AI4SWEng will employ agile methods in the implementation of the project, which means that requirements will be refined and improved as they are investigated in the project. Furthermore, the Consortium endeavours to ensure continued relevance of the work that is carried out considering the background of very fast ongoing research and development on AI in general and for software engineering in particular. It is expected that this will require adjustments in research and development activities as the project progresses, for instance to incorporate and contribute to new standardisation efforts and to apply and test newly developed AI models and techniques.

To manage the requirements engineering process and the execution of the project, the Consortium uses and online requirements and task tracking system. This system and associated processes are briefly introduced in the next section.

Section 2 proceeds by summarising the initial requirements gathering activities and introducing the methodological foundation and infrastructure that enabled the UI-REF hierarchy-resolution process reported in Section 3.

2 Initial Requirements and User Needs Gathering

This section summarises the requirements gathering undertaken in AI4SWeng as part of the work in WP 2. The section introduces the requirements management methodology and infrastructure used and briefly characterises the requirements gathering work undertaken and documented in other deliverables of the project for reference.

2.1 Methodological Underpinning

In AI4SWEng, Requirements Engineering is strictly methodologically guided for socio-ethical and regulatory compliance and user-led use-context-specific requirements, conflicts resolution and dynamic (re)prioritisation.

The User-Intimate Requirements Hierarchy Resolution and Usability Evaluation Framework (UI-REF) [1] has its foundations in the Psycho-Cognitive Science of Human Memory Recall [2], human Judgement and Decision Making (JDM) [3] and User Experience (UX) Design.

Accordingly, UI-REF supports dynamic conflict resolution and prioritisation of requirements hierarchy to support the optimisation of design for socio-ethical and regulatory compliance and optimised user-intimate personalisation and acceptability in interactive systems co-design. UI-REF is established as an ontologically- committed methodology to ensure that the most-deeply-valued needs of the stakeholders are elicited and ranked, and the root rationale for requirements evolution track and trace remains transparent and contextualised to help resolve stakeholder conflicts. UI-REF supports the dynamically evolving requirements of the users to inform online service provisioning and cyber-physical systems of systems usability. Requirements prioritisation in UI-REF is finely resolved while a promotion/demotion path for lower/higher priority requirements is delineated to ensure that as the requirements evolve so will their resolution and prioritisation. As such our methodology is also consistent with the principles of GDPR [4], AI-Act [5], ADRA[1], Data-Act [6] and Z-Inspection® [7] and more broadly with Design-by-Contract to underpin compliance assurance with socio-ethical as well as other societally desirable merits of systems such as safety, security, accuracy, efficiency, scalability robustness, interpretability (transparency explainability) and fairness (lack of bias). This ensures the design, development and deployment of trustworthy solutions with trade-offs dynamically optimised for the situated use-context per stakeholder preferences.

[1] https://adr-association.eu

2.2 Requirements Management Infrastructure

This section concisely introduces the requirements and task management infrastructure used in AI4SWEng. The presented infrastructure has been used throughout WP2 and will continue to be used throughout the overall project.

2.2.1 Requirements Management Using GitHub

The Consortium manages requirements as well as tasks and bugs for the project activities in a unified task management system for the project that has been implemented using the GitHub Projects [8] infrastructure.

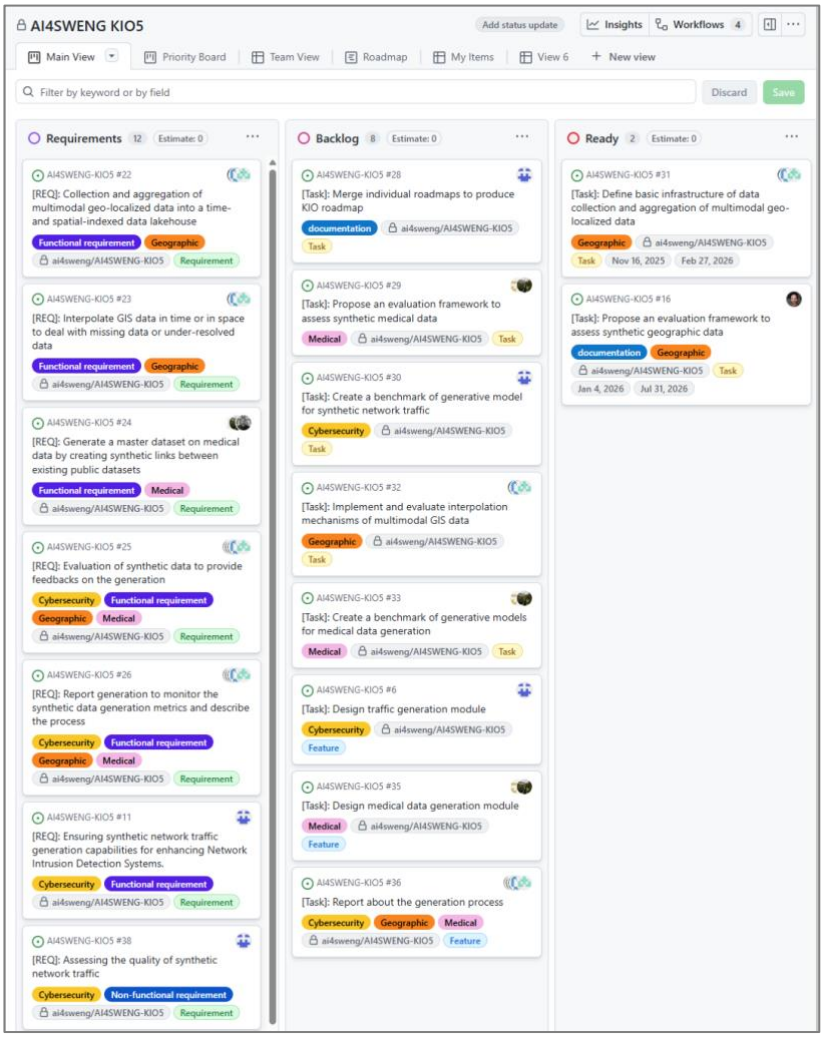


Figure 1: Example task tracking board from AI4SWEng

The environment provides a secure team environment with a flexibly adaptable task tracking mechanism that includes customisable templates for tasks, bugs and other trackable project management elements, agile process boards with functions for backlogs and customisable workflow stages, integration with GitHub repository and DevOps functionalities and API accessibility for automation. Figure 1 shows an example GitHub Projects board (for KIO 5) used in the project for illustration.

2.2.2 Requirements Management Implementation

AI4SWEng tracks requirements, tasks, bugs and other issues using three groups of GitHub project boards:

- Each KIO in the project (13 in total) uses a dedicated board to track progress in achieving the KIO ambitions.
- Each project use-case (5 in total) employs a dedicated board to track formal use-case descriptions and test definitions as well as use-case requirements, tasks and issues to resolve.
- Auxiliary project boards are used to manage the projects infrastructure itself as well as cross-cutting issues including ethics and dissemination activities. The number of these boards is expected to fluctuate over time.

All project Partner organisations contribute to the requirements management board using person-specific and/or functional user accounts assigned to the project requirements, tasks and other issue types.

Custom Issue templates have been created to facilitate the requirements gathering, representation and processing following the UI-REF methodology [9] (see also D2.4, Section 3.5 for a concise summary) as well as the representation of use-case and test-case specifications (see Section 5 and Section 6 in this document for details). Tasks are categorised and tagged with labels to facilitate quick visual inspection and filtering of tasks by type as part of activities. Figure 2 shows an example requirement representation from the KIO 5 task tracking board (please see Section 3.2 for further information on the requirement specification fields used to elaborate AI4SWEng requirements).

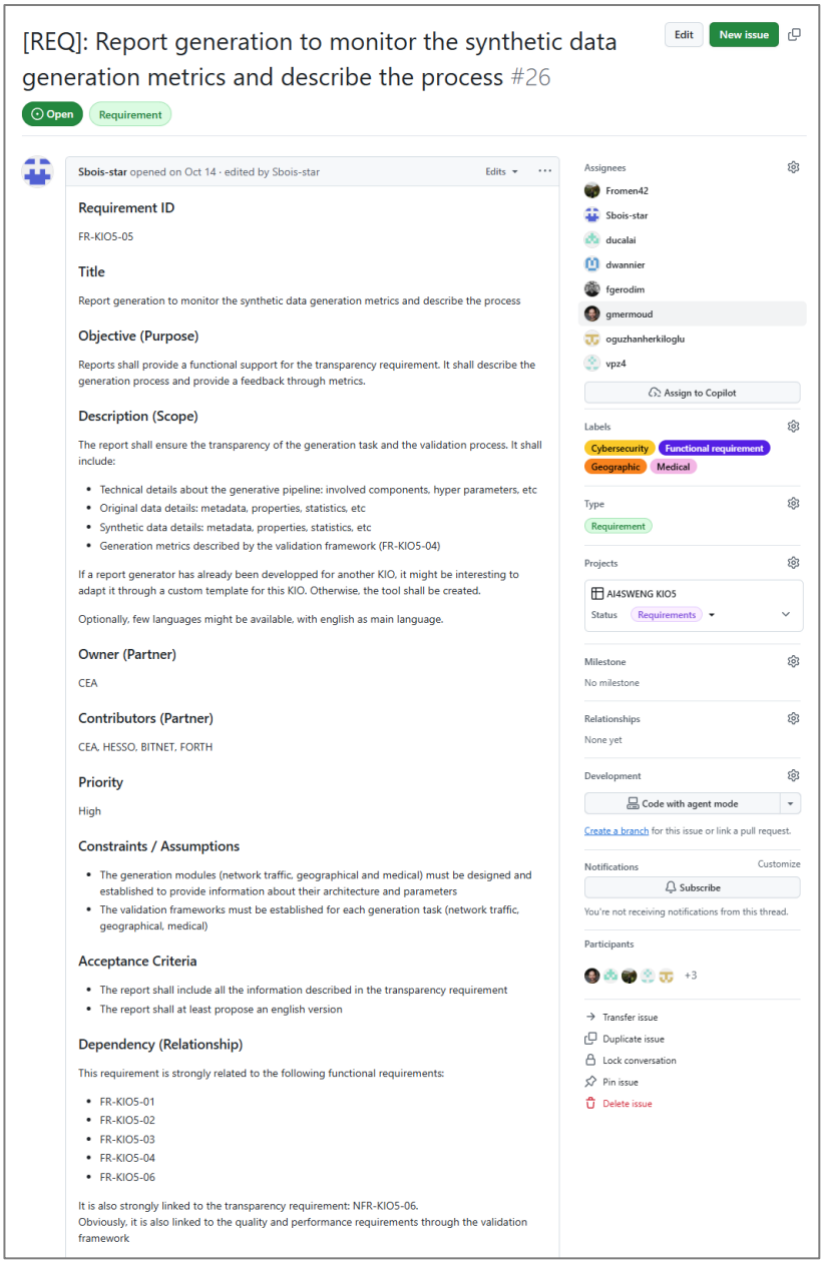


Figure 2: Example requirement entry from KIO 5 task tracking

Following on from the requirements engineering and system architecture phases documented in this deliverable, the project will continue to use GitHub for managing tasks, sharing technical project outcomes between Partners and generating artefacts for use in the project demonstration and evaluation activities for the project use-cases and target user groups.

2.3 Target User Groups

To ensure comprehensive coverage of stakeholder perspectives, five principal user groups were explicitly defined in Deliverable D2.4, Section 3.1. Each group was treated as a distinct stakeholder ontology within the UI-REF process, guaranteeing that the consolidated requirement hierarchy serves all constituencies without compromise to

higher-valence socio-ethical or regulatory merits. The five main user groups in AI4SWEng are as follows:

1. **Developers and Practitioners** working in the software engineering domain in design, coding, testing and similar roles directly engaged in software engineering and DevOps processes,
2. **Integrators and Technology Providers** that deliver solutions to customers as their product or service offering,
3. **Regulators and Certification Authorities** that oversee software engineering, AI, cybersecurity and data use compliance,
4. **End-User Organisations and Application Stakeholders** representing enterprises deploying software systems including AI-driven solutions in application domains such as the use-case domains of AI4SWEng,
5. **Research and Training Communities** such as universities, RTOs and developer communities that research, advance and disseminate AI-assisted software engineering.

These user groups must be considered when defining the project requirements, use-cases and system architecture, especially given the background that many of the requirements in AI4SWEng are most directly applicable to those immediately engaged in DevOps and software development activities.

2.4 Types of Actor-System Interactions

How do the identified user groups interact with the AI4SWEng solutions, potentially both separately and integrated as a solution stack? D2.4 presents ten main activity types that are expected to recur in user-system interactions, often transversal to KIOs and the project use-cases. Table 1 briefly recaptures the identified actor-system interactions presented in D2.4[2]:

Table 1: Important types of actor-system interactions

Title	Purpose	Actors
AI-Driven Software Development Automation	Utilise AI-powered technologies to automate software development processes	Software developers, AI engineers, DevOps teams
AI-Based Software Integration	Ensure that AI-driven software solutions are seamlessly	Software developers, AI solution providers, Corporate IT teams

[2] The table includes minor revisions identified after completion of D2.4. These will be used as input for refinements with the final WP2 deliverable D2.5.

	incorporated into SME operations	
AI-Powered Cybersecurity Analysis	Strengthen cybersecurity protocols for software solutions powered by AI	Cybersecurity experts, AI developers, Corporate IT security teams
Human-Centric AI System Interfaces	Make sure that AI system interfaces are ethical and easy to use	UX/UI designers, AI developers, market researchers
Data security and privacy compliance	Improve privacy compliance and data security in AI-powered solutions	Cybersecurity agencies, Data protection authorities, Compliance officers
AI-Powered Regulatory Compliance	Ensure that software solutions powered by AI adhere to legal and regulatory requirements	Regulatory agencies, compliance officers, Policymakers
Ethics and Fairness Assessment	Ensure that AI solutions adhere to ethical norms	Regulatory bodies, Ethics committees, AI researchers
AI-Driven Monitoring of Changing Security Guidelines	Monitor and evaluate changing security guidelines and legislation pertaining to AI	Government agencies, Legal analysts, Policymakers
Automated AI Training and Upskilling	Assist workers and SMEs in adjusting to AI-driven procedures	Employees, HR departments, AI system providers
AI-Assisted Decision Support	Leverage AI insights to assist enterprises in making data-driven choices	Business owners, Business analysts, Managers

More detailed information on these actor-system interactions can be found in D2.4, Section 3.3, which presents for detailed descriptions and behaviour diagrams for each of the identified actor-system interactions.

2.5 KIO Needs and Initial Requirements

The project KIOs formulate and organise the project ambitions into thirteen topics and purposes that the project will engage with. They play a central role in the definition and organisation of project activities and are also used to organise the everyday collaborations in the project (as can also be seen in the use of KIOs for activity tracking, see Section 2.2).

Consequently, the bulk of project functional and non-functional requirements are organised grouped by the KIOs they are part of.

Deliverable D2.4 reports on the initial process of requirements engineering and provides descriptions for requirements per KIO on two different levels: at the level of high-level needs and at the level of initial functional and non-functional requirements. High-level needs per KIO were first elicited from project Partners and used to kickstart the identification of initial functional and non-functional KIO requirements in Task 2.4. The high-level functional and non-functional needs identified in D2.4 are reproduced below in Table 2 and Table 3 as they provide a useful summary of high-level views on requirements per KIO and the output generated in D2.4.

Table 2: KIO functional needs identified in D2.4

KIO	Functional Needs
KIO1 AI4SWEng AI Engineering Suite	1. Unified KIO Integration Platform 2. Front-End GUI for Lifecycle Interaction 3. DevOps Workflow Orchestration 4. Containerised Service Deployment 5. Horizontal LLM Integration
KIO2 Bug Locate & Fix: Reverse Execution & Dynamic Slicing Component	1. Source code language support 2. Trace Capture & Replay 3. AI-assisted replay simulators 4. Dynamic slicing 5. AI Fault Localization 6. Large Language Model fine-tuned for bug detection 7. Automatic generation of fix suggestions 8. Thread-aware trace capture and replay simulation
KIO3 Natural Language Processing (NLP) to Formal Requirements Component	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
KIO4 Formal Requirements-to-Code Architectural Planning Component	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
KIO5 Synthetic Data Generation Tool	1. Generate compliant synthetic traffic at flow level 2. Generate synthetic traffic representing attacks at flow level 3. Merge GIS layers to create a new synthetic energy consumption map

KIO	Functional Needs
	4. Generate synthetic GIS layers to create new synthetic maps representing non-existing energy consumption scenarios 5. Generate a master medical dataset 6. Propose validation methods for each generation task 7. Report generation for generation tasks
KIO6 Data Validation-Preparation Tool	1. Comparison between input data and properties schema 2. Automated data preprocessing 3. Report generation for data validation, statistical properties, code errors, aggregate 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
KIO7 AI-SysDev Tool incorporating LLM Fine Tuning and Inferencing	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
KIO8 AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components	1. Code optimisation depending on the specified architectures 2. Software reconfiguration recommendations depending on the optimised code and the chosen 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
KIO9 Responsible Innovation and Design-by-Contract Compliant Coding	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)
KIO10 Energy-efficiency with Model-Driven TinyML and LLMs	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 Modelling 7. Address Complex Validation Challenges with Domain-Specific Modelling 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

KIO	Functional Needs
	13. Integrate with Microcontroller Variant of TensorFlow Lite
KIO11 AI-Powered Test Automation Tool (TAT)	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 (prioritisation and redundancy detection, prunin if needed) 4. Automated test script generation 5. Visualisation of TAT results (GUI)
KIO12 AI-based Cybersecurity & Multi-Level Security by Design	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
KIO13 Developer Training for using AI-Powered Software Development Tool	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

Table 3: KIO non-functional needs identified in D2.4

KIO	Non-Functional Needs
KIO1 AI4SWEng AI Engineering Suite	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
KIO2 Bug Locate & Fix: Reverse Execution & Dynamic Slicing Component	1. Common operations in the Reverse Execution & Dynamic Slicing Component shall complete in ≤ 5 seconds 2. Minimum bug-localization accuracy of 95% 3. Updatable structure
KIO3 Natural Language Processing (NLP) to Formal	1. Fast response to request 2. Handle concurrent requests 3. No ambiguity or contradiction 4. Securely stored data 5. Artifact versioning support

KIO	Non-Functional Needs
Requirements Component	6. Graceful error detection and recovery 7. Intuitive UI
KIO4 Formal Requirements-to-Code Architectural Planning Component	1. Fast response to request 2. Batch processing 3. High relevance of suggested architecture 4. Securely stored data 5. Intuitive interface 6. Architectural component versioning support 7. Graceful error detection and recovery 8. Support for compatible output formats
KIO5 Synthetic Data Generation Tool	1. Fast generation and low latency for data interactions 2. Handle wide inputs and generate large synthetic datasets / multithreading for multi-GPU trainings 3. Synthetic data matches input data 4. Data protection, injection protections 5. Intuitive UI
KIO6 Data Validation-Preparation Tool	1. Optimised computation of statistic properties, fast validation and preprocessing of data 2. Support large number of data and schema 3. Error detection, analysis and validation procedures and reports must be accurate 4. Documented and easy to update/debug 5. Able to retrieve logs and observe the different steps of the preparation and validation of data 6. Ergonomic/user-friendly UI 7. Data must be securely stored
KIO7 AI-SysDev Tool incorporating LLM Fine Tuning and Inferencing	1. Fast code generation and debugging 2. High code correctness 3. Parallel inference and fine-tuning tasks 4. Isolated execution for debugging 5. Intuitive UI 6. 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
KIO8 AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components	1. Fast and efficient in terms of resources 2. Able to perform optimisations for large code bases 3. Documented and easy to update/debug 4. Transparency 5. Ergonomic/user-friendly 6. Data must be securely stored
KIO9 Responsible Innovation and Design-by-Contract Compliant Coding	1. Fast authentication and efficient data transmission 2. Support a decentralised network 3. Energy-efficient decentralised networks and tools 4. Inputs and outputs should be transparent 5. Compliance check analysis must be self-explainable 6. End-to-end secure data transmission, storage and multifactor authentication must be provided
KIO10	1. Reduce Energy Consumption by Limiting Cloud Data Transfer

KIO	Non-Functional Needs
Energy-efficiency with Model-Driven TinyML and LLMs	2. Minimise 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
KIO11 AI-Powered Test Automation Tool (TAT)	1. Must be used by huge development teams connecting to the system from different end points 2. Easy documentation, lightweight/heavyweight security depending on the criticality of the development 3. Inputs and outputs should be transparent to the user 4. Ergonomic/user-friendly 5. 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
KIO12 AI-based Cybersecurity & Multi-Level Security by Design	1. Accurate test scenario generation and vulnerability detection 2. Support adding new types of vulnerabilities or attack types depending on the development 3. Easy documentation, lightweight/heavyweight security depending on the criticality of the development 4. Inputs and outputs should be transparent to the user 5. Ergonomic/user-friendly
KIO13 Developer Training for using AI-Powered Software Development Tool	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

The full needs elicitation tables per KIO can be found in D2.4, Section 3.4. Some requirements entries in the tables above have been shortened for readability.

Using these high-level needs as the basis, the project team created an initial set of preliminary functional and non-functional requirements for each KIO (see D2.4, Section 3.5 for a narrative summary per KIO and Appendix B for abbreviated table representations

of each identified requirement). These initial requirements were gathered through the GitHub project infrastructure described earlier in Section 2.2. The preliminary requirements generated in T2.4 served as a main input for the requirements refinement process in T2.5 reported on in this document.

2.6 Security, Traceability and Privacy-by-Design Requirements

In a parallel track to the KIO-driven requirements engineering reported on in D2.4, Task T2.1 performed a focused requirements engineering process for security, traceability and privacy-by-design requirements, documented as part of Deliverable 2.1. The work in T2.1 encompasses the analysis of the relevant regulations and standards, the AI4SWeng use-cases and engagement with all Partners regarding the formulation of requirements. This process identified eight requirements regarding provenance, eight requirements concerning privacy and nine requirements concerning security and transparency, all with a special focus on GDPR and AI-Act requirements. Table 4 below lists the IDs and titles of all D2.1 requirements for reference:

Table 4: Security, traceability and privacy-by-design requirements from D2.1

ID	Requirement	ID	Requirement
RQ-PROV-01	Ingest and persist provenance metadata	RQ-PRIV-06	Controlled data access and role-based segregation
RQ-PROV-02	Expose provenance API/GUI	RQ-PRIV-07	Audit logging for privacy operations
RQ-PROV-03	Store logs in tamper-proof format	RQ-PRIV-08	Explicit consent management
RQ-PROV-04	Interoperable provenance representation	RQ-SEC-01	Strong authentication and RBAC
RQ-PROV-05	Versioning and lineage traceability	RQ-SEC-02	Encryption of data at rest
RQ-PROV-06	Provenance validation on data exchange	RQ-SEC-03	Encryption of data in transit
RQ-PROV-07	Machine-actionable export	RQ-SEC-04	Vulnerability scanning in CI/CD
RQ-PROV-08	Automated data lineage for AI models	RQ-SEC-05	Integrity checks on artefacts
RQ-PRIV-01	GDPR compliance by default	RQ-SEC-06	Anomaly detection and alerts
RQ-PRIV-02	Redact and anonymise sensitive data	RQ-SEC-07	Logging and monitoring
RQ-PRIV-03	Limit personal data collection	RQ-SEC-08	Secure software supply chain (SBOM)
RQ-PRIV-04	Purpose tagging and enforcement	RQ-SEC-09	LLM safety controls
RQ-PRIV-05	Delete intermediate artefacts		

The full tables for these requirements including full descriptions and validation criteria can be found in D2.1. No descriptions of validation criteria from D2.1 were changed as part of the work towards the present deliverable D2.6.

The requirements above were defined independently of specific KIOs as cross-cutting requirements that must be adopted wherever appropriate throughout the project. As the activities were performed in parallel, some requirements specified in D2.1 have duplicates for specific KIOs in D2.4. This duplication was left in place to minimise the need for cross-verification between the two parallel processes.

3 Revision and Completion of Initial Requirements

The aggregation, revision and completion of initial project requirements presented in this section documents the work performed in Task 2.5 of AI4SWEng.

3.1 Revision and Completion Process

The revision and completion process for the AI4SWEng requirements was carried out based on the work carried out in the WP 2 tasks T2.1 and T2.2, the original project DoA and the implementation framework for the DoA documented in Deliverable 1.1. presents the four-step process involving all Partners that was carried out:

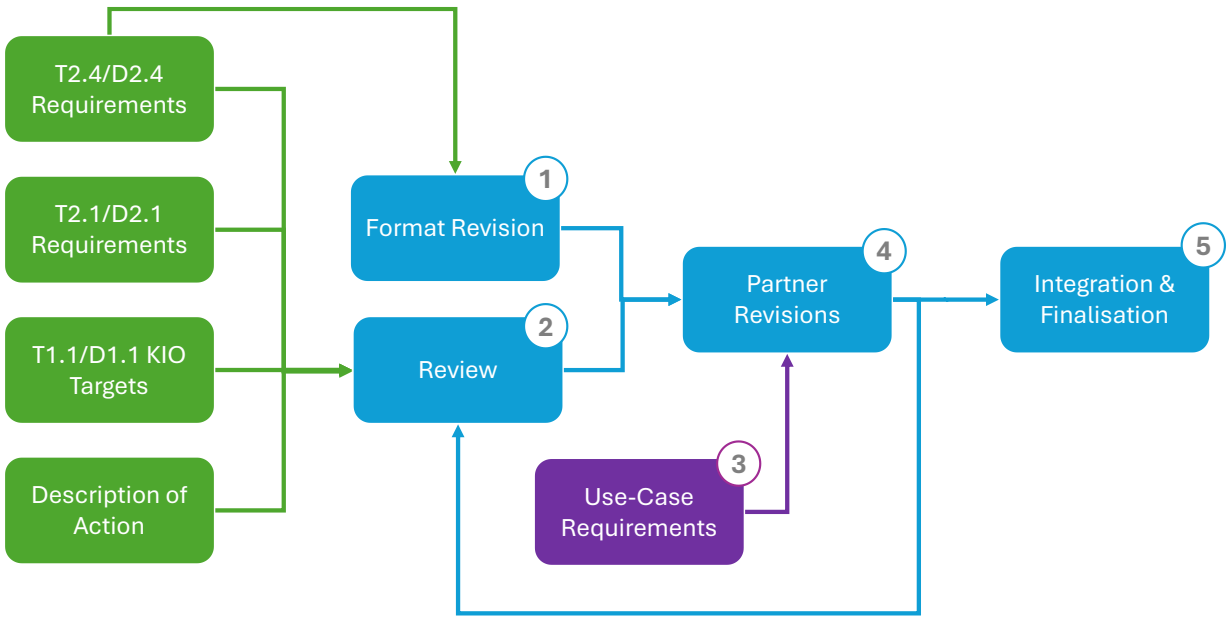


Figure 3: Revision and completion process for requirements

The main inputs into the requirements completion process (highlighted in green) were of course the initial requirements identified in D2.4 and the cross-cutting requirements defined in D2.1. Work undertaken on operationalising the DoA in T1.1 (documented in D1.1) was used as additional input. The project DoA itself was used to ensure that the specified requirements meet the project ambition set out in the Description of Action. Use-case requirements, which were not gathered as part of the initial requirements, were defined as part of WP 2 activities for defining the project use-case activities.

The revision process itself consists of two main strands:

1. Refinement of the format and expected input for defining the project requirements including a revision of the requirements templates used to record and update requirements in the GitHub projects repositories ensuring full support for all UI-REF

fields (Priority, Dependency (Relationships), Invariants, Pre-Conditions, Post-Conditions, Behaviour and Sequences, Embodiment segments)

2. A structured review, revision, and finalisation process comprising:
    a. an iterative review/revision loop in which all Partners contributed corrections, clarifications, priority adjustments, dependency mappings, and embodiment details.
    b. targeted completion of mandatory fields, elimination of ambiguous or contradictory wording, and strengthening of cross-references across KIOs, use-cases, and cross-cutting requirements.
    c. explicit re-evaluation of requirement priorities according to UI-REF psycho-cognitive and regulatory valence criteria (high = mandatory, medium = desirable, low = optional).
    d. specification of measurable acceptance criteria wherever feasible.
    e. final unanimous editorial review and approval by the Work Package 2 requirements process leadership team, with particular attention to duplicate removal, consistent use of normative terminology (shall/must/should), and overall ontological coherence.

The next two subsections describe each of these processes and the main activities performed to arrive at the set of consolidated requirements for AI4SWEng.

3.2 Requirements Format Specification and Revision

The requirements specification format used in AI4SWEng was revised in two stages.

First, the originally envisioned requirements specification format, consisting of User Need and User Story definitions was expanded and integrated into a single requirements format that was a superset of both. This choice was motivated by positive experience of employing a similar requirements specification format in a previous successful research project that also employed the UI-REF methodology for requirements engineering. This decision was made in the initial stages of WP 2 activity and is already reflected in D2.1 and D2.4 requirements output.

Second, the requirements format initially devised for D2.4 was extended and further revised to provide more information on relations between requirements. Additional minor adjustments to the requirements format were made to improve clarity, and parts of the specification format were declared as optional to avoid unnecessary efforts for completing specification fields that were not applicable for some requirements. Table 5 describes the requirements specification format used for D2.6 and shown for instance in Figure 2 on page 19.

Table 5: AI4SWeng requirements format

Field	Description
Requirement ID	Unique requirement identifier handle
Labels	Identification of the type of requirement and optionally other properties (e.g. “security requirement”)
Title	Short descriptive title
Objective (Purpose)	Goal of the requirement
Description (Scope)	Full description of the requirement
Owner (Partner)	Lead Partner responsible for requirement
Contributors (Partner)	Contributors towards achieving requirement
Priority	High, medium or low priority
Constraints/Assumptions	Underlying assumptions that need to be met for the requirement
Acceptance Criteria	Criteria that need to be met for the requirement to be met
Dependency (Relationships)	Dependencies on other requirements
Pre-Condition(s)	Conditions that must be met prior to fulfilling a requirement
Input	Input needed to address the requirement
Invariants	Conditions that must hold true throughout the fulfilment of the requirement
Behaviour and Sequences	Steps that form part of the requirement
Output	Results of successfully addressing the requirement
Post-Condition(s)	Conditions that must be met after fulfilling a requirement

The format consists of the following segments:

- **Identification:** Requirement ID, Labels and Title of the requirement are used for identification, referencing and categorisation in the project.
- **Description:** Objective and Description of the requirement elaborate the goals and scope of the requirement.
- **Execution:** Owner, Contributors, Priority, Constraints/Assumptions, Acceptance Criteria and Dependencies define the project-related properties needed for

assigning, scheduling, completing and verifying that requirements have been addressed.

- **Embodiment:** Input, Invariants, Behaviour and Sequences, Output and Post-Conditions define factors regarding the embodiment of a requirement that are important parts of a particular requirement. Embodiment properties are considered optional and completing them for each requirement is not mandatory.

This requirement format is used for functional and non-functional requirements for KIOs and Use-Cases. The cross-cutting requirements for security, traceability and compliance use a simplified requirements format consisting of Identifier, Title, Description and Acceptance Criteria only. For instances where further specification is useful or required as part of a particular KIO or UC, separate requirements with dependencies on the relevant fundamental security, traceability and compliance requirements are used to record the specific embodiments of the requirements with a KIO or UC.

3.3 Iterative Requirements Refinement

The iterative requirements improvement process used in the project involved iterative review and refinement cycles managed by the WP2 lead Partners across all requirements and the KIO and UC leaders for the topics that they are heading in the project.

The following areas for refinement were addressed as priorities during the refinement process:

1. Completion of requirements specifications to include all mandatory fields.
2. Review of the priorities associated with requirements to appropriately reflect UI-REF priority levels where high priority indicates a mandatory requirement, medium priority signals a desirable requirement and low priority identifies an optional requirement.
3. Clarification and improvements to the specificity of language use in titles and descriptions of requirements.
4. Improvements to cross-referencing within KIOs/UCs as well as across KIOs/UCs and towards security, traceability and compliance requirements.
5. Revisions of acceptance criteria to be as specific as possible at the time of formulation.
6. Inclusion of Embodiment descriptions in requirements specifications when deemed useful.

The process involved 2-3 iterations for revisions over a six-week period. Use-case requirement specifications were reviewed both by UC-responsible Partners and by Partners involved in addressing the UC requirements via their KIO contributions.

As a final refinement step, an editorial review of the overall set of requirements was carried out. The focus of this review was on a holistic revision of the overall set of project

requirements across all requirements defined for the project. The main tasks performed were:

1. Review and identification of duplicate requirements across KIOs, UCs and security, traceability and compliance requirements, and revision of requirements to improve their specificity for their application or removal of unnecessary requirements.
2. Improvement of cross-referencing of requirement dependencies across boundaries.
3. Review of the use of specific signalling language and expressions across all requirement formulations (e.g. for proper use of “shall”, “must” and similar signal terminology for requirements).

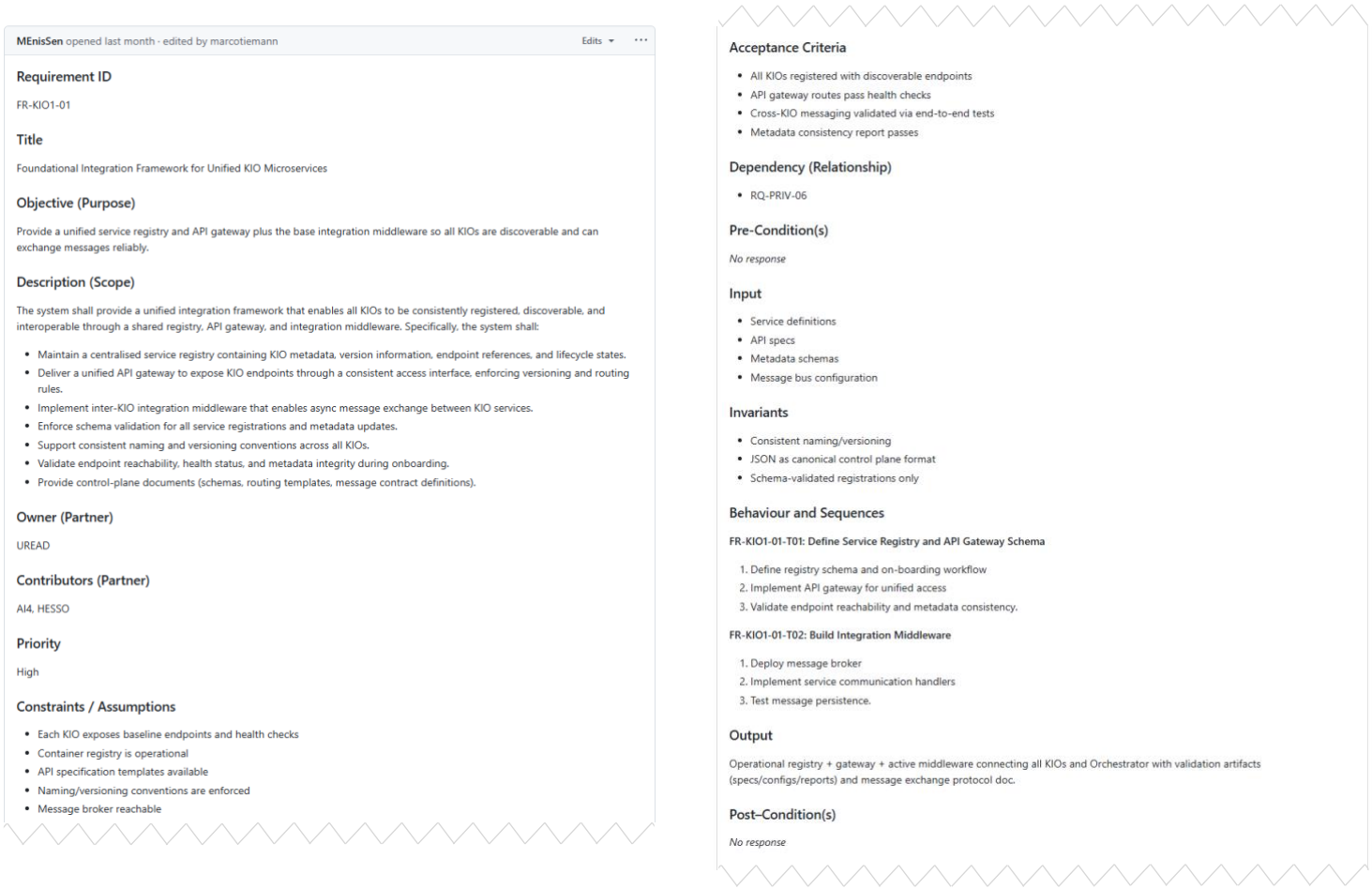
Two independent reviews of the overall set of requirements were undertaken. The reviewers discussed all proposed changes, and unanimity was required for a change to be made.

3.4 Continuing Iterative Requirements Engineering

With the refinement of KIO requirements and the initial definition and refinement of UC requirements, the project has created an overall set of requirements for the full spectrum of activities covering all 13 KIOs and 5 UCs. These requirements will continue to be evolved as dynamic, living artefacts as defined in the UI-REF methodology and will continue to be maintained in the project’s GitHub task management infrastructure. Figure 4 shows the full body of a KIO requirement as defined in AI4SWEng task tracking.

Upcoming activities in the requirements engineering process include in particular the following items:

- Creation of initial prototypes in technical work packages and elicitation of feedback from stakeholders, in particular UC stakeholders, as input for revisions of both KIO and UC requirements. These revisions will be used to relate the KPI test-cases defined for the UCs to KIO requirement functional and non-functional performance expectations, refining initial acceptance criteria specified for KIO requirements as well as qualitative performance targets, with careful consideration given to the research and innovation objectives of the project and the target TRL level to avoid large effort expenditures on issues not at the heart of the AI4SWEng vision.
- Incorporation of late-breaking external research and innovation outcomes (including novel algorithms, benchmark datasets, etc.) into the project to ensure continuing relevance of the project in an extremely fast-moving area of research and development.
- Incorporation and reflection of upcoming deliverable D2.3 “Needs Re-Elicitation and Governance Organisation” (M16), which will elaborate further on requirements that specifically relate to the AI-Act, and D2.2 “Organisation-Specific Onsite Requirements” (M18), which will focus on modularity and reusability of the AI4SWEng KIOs, with special consideration given towards standards-compliance.



MEnisSen opened last month · edited by marcotiemann — Edits

Requirement ID

FR-KIO1-01

Title

Foundational Integration Framework for Unified KIO Microservices

Objective (Purpose)

Provide a unified service registry and API gateway plus the base integration middleware so all KIOs are discoverable and can exchange messages reliably.

Description (Scope)

The system shall provide a unified integration framework that enables all KIOs to be consistently registered, discoverable, and interoperable through a shared registry, API gateway, and integration middleware. Specifically, the system shall:

- Maintain a centralised service registry containing KIO metadata, version information, endpoint references, and lifecycle states.
- Deliver a unified API gateway to expose KIO endpoints through a consistent access interface, enforcing versioning and routing rules.
- Implement inter-KIO integration middleware that enables async message exchange between KIO services.
- Enforce schema validation for all service registrations and metadata updates.
- Support consistent naming and versioning conventions across all KIOs.
- Validate endpoint reachability, health status, and metadata integrity during onboarding.
- Provide control-plane documents (schemas, routing templates, message contract definitions).

Owner (Partner)

UREAD

Contributors (Partner)

AI4, HESSO

Priority

High

Constraints / Assumptions

- Each KIO exposes baseline endpoints and health checks
- Container registry is operational
- API specification templates available
- Naming/versioning conventions are enforced
- Message broker reachable

Acceptance Criteria

- All KIOs registered with discoverable endpoints
- API gateway routes pass health checks
- Cross-KIO messaging validated via end-to-end tests
- Metadata consistency report passes

Dependency (Relationship)

- RQ-PRIV-06

Pre-Condition(s)

No response

Input

- Service definitions
- API specs
- Metadata schemas
- Message bus configuration

Invariants

- Consistent naming/versioning
- JSON as canonical control plane format
- Schema-validated registrations only

Behaviour and Sequences

FR-KIO1-01-T01: Define Service Registry and API Gateway Schema

1. Define registry schema and on-boarding workflow
2. Implement API gateway for unified access
3. Validate endpoint reachability and metadata consistency.

FR-KIO1-01-T02: Build Integration Middleware

1. Deploy message broker
2. Implement service communication handlers
3. Test message persistence.

Output

Operational registry + gateway + active middleware connecting all KIOs and Orchestrator with validation artifacts (specs/configs/reports) and message exchange protocol doc.

Post–Condition(s)

No response

Figure 4: Requirements specification in GitHub projects

As part of the process, requirements may be added, removed and revised to improve overall requirements definition for the project. With the submission of D2.5 “Requirements and Report for Data Preparation Activities (final version)” in M28, the major requirements engineering activities are expected to have been completed and requirements are expected to have been finalised with final detailed requirement descriptions, handshakes with the UC stakeholders and the leads of the evaluation activities in WP6, requirements acceptance criteria that contribute to KPI measurements in the UC-driven WP6 validations and final quantified non-functional performance, security, privacy-by-design, usability, reliability, traceability, transparency and compliance specifications with equally strong focus on measuring outcomes with KPI relevance.

The next section, Section 4, will provide more information on the UC requirements and KIO requirements created through this process.

4 Project KIO and UC Requirements

The present section provides a concise overview of the consolidated functional and non-functional requirements for the thirteen Key Innovation Outcomes and the five use-cases, as finalised through the process documented in Section 3.

Each requirement is summarised in tabular form, listing the unique identifier, title, objective (purpose), and assigned priority (high, medium, or low). High-priority requirements reflect those merits that achieved the highest psycho-cognitive and regulatory valence and are therefore mandatory; medium-priority requirements are desirable; low-priority requirements remain optional but retain traceable promotion pathways should future conditions warrant their elevation.

The full, detailed specifications - including description, owner, contributors, constraints/assumptions, acceptance criteria, dependencies, pre-conditions, invariants, behaviour and sequences, output, post-conditions, and embodiment details where applicable - are maintained as living documents in the AI4SWEng GitHub repository. This repository continues to serve as the single source of truth for ongoing requirement evolution under continuous UI-REF governance.

4.1 Key Innovation Outcome Requirements

Each KIO is presented in a separate subsection with a brief introductory recap and a table listing the KIO requirements.

4.1.1 KIO1: AI4SWEng AI Engineering Suite

KIO1 implements the overall AI4SWEng engineering suite including deployment, workflow management and agent orchestration as well as centralised services and the unified engineering suite user interface.

Table 6: KIO1 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO1-01	Foundational Integration Framework for Unified KIO Microservices	Provide a unified service registry and API gateway plus the base integration middleware so all KIOs are discoverable and can exchange messages reliably.	High
FR-KIO1-02	Containerisation and Deployment Blueprint for the Individual KIOs across the entire suite	Establish a uniform, secure, and auditable approach to containerising, validating, and deploying KIOs; supporting both source-based and binary-only containerisation workflows.	High

ID	Title	Objective	Priority
FR-KIO1-03	Inter-KIO Middleware for Async Message-Based Communication	Unify synchronous and asynchronous KIO communication with message contracts.	High
FR-KIO1-04	Workflow Orchestration Layer for Execution, Triggers, and Dependencies operated by advanced LLM-based agentic teams	Establish a unified orchestration layer that manages execution order, triggers, and dependencies among KIOs, evolving from a rule-based orchestrator into an agentic, LLM-based system	High
FR-KIO1-05	Central LM Engine for Generation, Reasoning, and Guard-railed Execution	Offer a single, callable inference service for generation/reasoning/code tasks to all KIOs and orchestration with tuneable parameters and safety hooks	High
FR-KIO1-06	Agentic Interaction Layer for LM-Engine-driven System Reasoning, Actions, and Memory Sharing	Establish a modular Agent Engine capable of interoperating with the Central LM Engine (FR-KIO1-05), enabling intelligent, role-based, and context-aware agent operations across all KIOs and orchestration.	High
FR-KIO1-07	Web-Based GUI for Workflow Monitoring and Control	Provide a responsive web GUI to launch, monitor, and analyse workflows with secure multi-user access	High
FR-KIO1-08	Monitoring and Telemetry: Implementation of an overall system tracking with monitoring, energy, hardware, and time usage indicators	Deliver end-to-end observability across KIOs, LM calls, and infrastructure, including energy, hardware, and time utilisation.	High
FR-KIO-09	Adaptive Multi-Persona Interaction and Guidance Layer	The usability aspect of the framework from both technical and non-technical user perspectives should be addressed and it should be ensured that regardless of profession, users can effectively use the provided capabilities across KIOs by utilizing an adaptive interaction layer. This layer should be able to take shape with respect to the user's needs and expertise level.	High

ID	Title	Objective	Priority
Non-Functional Requirements			
NFR-KIO1-01	Testability: Deterministic Validation and Reproducible Workflows	Ensure every layer (API Gateway, Middleware, Orchestrator, LM Engine, Agent Layer, Memory, GUI) is easily testable, with reproducible results where applicable, and fast feedback in CI.	High
NFR-KIO1-02	Performance: End-to-End Latency and Throughput Assurance	Make sure all operations operate within defined performance budgets for latency, throughput, and energy efficiency under varying workloads and deployment scales	Medium
NFR-KIO1-03	Scalability: Horizontal Expansion and High-Concurrency Operations	Ensure that the suite can scale horizontally across all KIO services, so that multiple workflows and branches can run concurrently without performance degradation, failures, or resource saturation.	Medium
NFR-KIO1-04	Updatability: Zero-Downtime, Backward-Compatible Evolution	Ensure that all KIO services can be updated, patched, or rolled back without interrupting active workflows, by supporting rolling updates, canary releases, schema-versioned migrations, and controlled feature rollout mechanisms.	Low

4.1.2 KIO2: Bug Locate & Fix: Reverse Execution & Dynamic Slicing Component

KIO2 focuses on debugging, bug identification, localisation and remediation and AI-enabled developer assistance for addressing bugs.

Table 7: KIO2 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO2-01	Source code language support (One language for PoC)	Implements foundational support for the selected Python and /or C language, enabling parsing, compilation, and trace generation.	High

ID	Title	Objective	Priority
FR-KIO2-02	Trace Capture & Replay to generate reversible trace data	Presents a debugger-like UI for manipulating execution traces interactively that permits forward/backward navigation and the inspection of state snapshots; enables exploration of recorded data and visualisation of def-use chains; allows for post-mortem interactive debugging sessions.	High
FR-KIO2-03	AI-assisted replay simulators that reconstruct previous states from logged diffs	Enables the manipulation of multiple execution traces within a single interactive session, thereby enabling comparisons across multiple runs; Implements heuristics to align execution traces, taking inspiration from sequence alignment technologies developed in the context of bioinformatics; Lays down foundation to curate sets of execution traces.	High
FR-KIO2-04	Post-mortem expression evaluator	Enables the (partial) reuse of the original source code during post-mortem interactive debugging; expressions requiring data not recorded in the trace query data from mocks; Enables the speculative exploration of data and behaviours not that were not captured in the execution traces, starting from arbitrary program points.	High
FR-KIO2-05	AI Fault Localisation to train from replayed traces and slice context	Uses machine learning to help detect anomalies in sets of execution traces. These anomalies are statistical hints of potential bugs that can be used to guide the developer upon debugging a given issue.	High
FR-KIO2-06	AI-assisted mocking	Uses generative AI to mock interactions with external services for evaluating expressions during post-mortem debugging.	Medium
FR-KIO2-07	Trace recorder	Records runtime traces of Python program executions including variable states and control flow information necessary to support post-mortem interactive debugging.	High

ID	Title	Objective	Priority
Non-Functional Requirements			
NFR-KIO2-01	Performance: Quick processing for common operations	Ensure acceptable latency and throughput for trace, slicing, replay, and inference tasks under common scenarios.	Medium
NFR-KIO2-02	Accuracy Evaluation and Validation	Measures accuracy of bug detection and fault localisation models and tunes parameters for better accuracy.	High
NFR-KIO2-03	Modularity: Updatable Structure	Enable independent updates, replacements, and scaling of KIO2 components without system-wide rebuilds.	Medium

4.1.3 KIO3: Natural Language Processing to Formal Requirements Component

KIO3 investigates the AI-driven translation of natural-language input into formal requirements specifications and the creation of related artefacts such as requirements specification documents[3].

Table 8: KIO3 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO3-01	Requirement Input Processing in Natural Language	To enable the system to understand and process user input provided in natural language (human language, NL).	High
FR-KIO3-02	Conversion of Natural Language Descriptions to User Stories	To automate the creation of user stories from natural language descriptions, standardising project requirements.	High
FR-KIO3-03	Automated Generation of Functional and Non-Functional Requirement Documents	To automatically generate comprehensive functional and non-functional requirement documents, providing a clear and structured outline of system features,	High

[3] Indeed, documents such as the one you are reading right now.

ID	Title	Objective	Priority
		performance criteria, security measures, and usability requirements, thereby centralising and standardising documentation.	
FR-KIO3-04	Conversion of Natural Language Descriptions to Software Wireframes	To automatically generate wireframes from natural language descriptions, enabling early visualisation and iterative design of software layouts and designs based on textual requirements.	High
FR-KIO3-05	Generation of Acceptance Criteria from Requirements	To automatically define clear and measurable acceptance criteria for software features, ensuring that all aspects of the software are considered complete and ready for delivery, thereby standardising quality assurance and project sign-off.	High
FR-KIO3-06	Extraction of Assumptions, Limitations, and Constraints from Requirement Inputs	To enhance project transparency and mitigate risks by automatically identifying and documenting key assumptions made during requirement gathering. This also aims to capture any limitations or constraints that could impact the software development process, fostering more realistic planning.	High
FR-KIO3-07	Export of Project Artifacts from the System	To enable seamless sharing and collaboration by enabling the system to export generated user stories, requirement documents, wireframes and acceptance criteria into widely recognised standard formats, ensuring broad compatibility and accessibility.	High
FR-KIO3-08	Interactive Review and Refinement of Generated Artefacts	To ensure the accuracy and completeness of all generated project artifacts by providing users with the ability to review, modify and refine them within the system, thus enhancing the quality of documentation and design outputs.	High

ID	Title	Objective	Priority
Non-Functional Requirements			
NFR-KIO3-01	Automated Clarity and Ambiguity Checks for Generated Artefacts	To guarantee the quality and usability of all generated project artefacts by ensuring they are consistently clear, concise and unambiguous. This prevents misinterpretations and streamlines subsequent development phases.	High
NFR-KIO3-02	System Performance and Stability under Concurrent User Load	To ensure system reliability and responsiveness under heavy load by enabling it to handle simultaneous requests from multiple users without any degradation in performance. This guarantees a consistent user experience even during peak usage.	Medium
NFR-KIO3-03	User-Friendly Interface for Natural Language Input and Artifact Output	To maximise user adoption and efficiency by providing an intuitive and easy-to-use interface. This will enable users to seamlessly input natural language descriptions and effectively view all generated project outputs.	High
NFR-KIO3-04	Accuracy and Completeness of Generated Artifacts	To ensure the high quality and reliability of all system outputs by guaranteeing that generated artefacts are consistently accurate and complete. This minimises errors and ensures they fully meet specified requirements.	High
NFR-KIO3-05	Encryption of User Inputs and Artefact Outputs in Transit	To safeguard user inputs and generated outputs by encrypting all data in transit. This ensures the security and privacy of sensitive information as it moves across networks.	High
NFR-KIO3-06	Performance and Responsiveness of NL Processing and Artefact Generation	To ensure a fluid and efficient user experience by guaranteeing that the system processes natural language descriptions and generates all outputs within a reasonable and responsive timeframe, avoiding user wait times.	High
NFR-KIO3-07	Ensuring System Stability and Positive UX with Error Detection, Logging, and Recovery Mechanisms	To ensure system stability and a positive user experience by implementing robust mechanisms for detecting and gracefully recovering	High

ID	Title	Objective	Priority
		from errors. This minimises disruption and prevents data loss.	
NFR-KIO3-08	Ensuring High Availability and Reliability for Artefact Generation Services	To guarantee continuous service access and consistent operational performance by ensuring the system maintains a high level of reliability and minimal downtime, maximising user access and business continuity.	High
NFR-KIO3-09	Logging of Interactions and Artifacts for Auditing and Performance Analysis	To enable robust auditing and continuous system enhancement by securely storing comprehensive logs of user interactions and generated outputs. This fosters accountability and supports data-driven optimisation efforts.	High
NFR-KIO3-10	Generated Artifact Versioning for Historical Tracking and Auditability	To enable robust tracking of changes and preserve historical context for all generated project artefacts. This aims to facilitate auditability, collaboration and simplified rollback to previous states.	High

4.1.4 KIO4: Formal Requirements-to-Code Architectural Planning Component

KIO4 targets the automated translation from formal requirements specifications to architecture specification using AI tools including generating related documentation and architectural code.

Table 9: KIO4 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO4-01	Formal Requirements Ingestion	To enable the system to efficiently process and utilise formal software engineering requirements generated from the KIO3 task, laying the groundwork for subsequent automated analysis and artefact generation.	High

ID	Title	Objective	Priority
FR-KIO4-02	Architectural Component Identification from Formal Requirements	To automatically extract and map architectural components and their interdependencies from formal requirements specifications, streamlining system design and development processes.	High
FR-KIO4-03	Architectural Component-Requirement Traceability Mapping	To establish and maintain a clear, automated traceability link between identified architectural components and their corresponding formal requirements, enhancing project understanding, impact analysis and verification processes.	High
FR-KIO4-04	Formalisation of Architectural Design Elements	To support robust software design by formalising architectural component properties, interfaces and relationships. This aims to ensure precision, reduce ambiguity, and facilitate analysis and validation of design decisions.	High
FR-KIO4-05	Interactive Visualisation of Architectural Components	To enhance understanding and communication of the software architecture by providing a mechanism for visually representing architectural components and their relationships, supporting clearer insights into system structure and interactions.	Medium
FR-KIO4-06	Providing Real-time Architectural Design Suggestions	To enhance the architectural planning phase by implementing a real-time suggestion mechanism. This aims to provide immediate feedback and actionable recommendations, thereby guiding designers toward optimal architectural decisions and proactively addressing potential issues.	High
FR-KIO4-07	Documentation and Attachment of Architectural Design Rationale	To enhance transparency and facilitate future maintenance by ensuring the system outputs and attaches the rationale behind architectural choices. This will provide clear reasoning for design decisions.	High
FR-KIO4-08	Export of Architectural Components for External Modelling Tools	To facilitate seamless integration with external visualisation and modelling tools by enabling the system to export architectural	High

ID	Title	Objective	Priority
		components and their relationships in widely adopted standard formats, enhancing collaborative design workflows.	
FR-KIO4-09	Bidirectional Traceability Between Architectural Components and Requirements	To ensure complete transparency and robust impact analysis by establishing and maintaining a clear, automated bidirectional traceability mapping between architectural components and their original formal requirements.	High
FR-KIO4-10	Export Project Scaffolding/Code Skeletons based on Architecture	To accelerate the software development lifecycle by automatically generating and exporting foundational project structures (scaffolding) and code skeletons directly from the defined architectural components, ensuring a consistent and rapid transition from design to implementation.	High
Non-Functional Requirements			
NFR-KIO4-01	Clarity and Unambiguity for Requirements-to-Architecture Mapping	To enhance the reliability and maintainability of the software system by guaranteeing that the mapping from requirements to architectural components is consistently clear, concise and unambiguous. This prevents misinterpretations and ensures accurate implementation.	High
NFR-KIO4-02	Performance Stability of Architectural Analysis Functions	To ensure system reliability and responsiveness under heavy load by enabling it to handle simultaneous requests from multiple users without any degradation in performance. This guarantees a consistent user experience even during peak usage.	Medium
NFR-KIO4-03	User-Friendly Interfacing with Code Developers for Architectural Interaction	To maximise user adoption and efficiency by providing an intuitive and easy-to-use interface. This will enable users to seamlessly input formal requirements and effectively view the generated architectural components.	Medium

ID	Title	Objective	Priority
NFR-KIO4-04	Valid Architectural Components that Minimise Requirement Discrepancies	To ensure the high quality and reliability of the software architecture by guaranteeing that generated architectural components are consistently accurate and complete. This minimises design flaws and ensures they fully meet specified requirements.	High
NFR-KIO4-05	Encryption of Formal Requirements and Architectural Data in Transit	To safeguard user inputs and generated outputs by encrypting all data in transit. This ensures the security and privacy of sensitive information as it moves across networks.	High
NFR-KIO4-06	Performance and Responsiveness of Requirements Processing and Architecture Generation	To ensure a fluid and efficient user experience by guaranteeing the system processes formal requirements and generates architectural components within a reasonable timeframe. This avoids user wait times and maintains responsiveness to inputs.	Med
NFR-KIO4-07	Error Detection, Logging, and Recovery for Architectural Functions	To ensure system stability and a positive user experience by implementing robust mechanisms for detecting and gracefully recovering from errors. This minimises disruption and prevents data loss.	High
NFR-KIO4-08	Ensuring High Availability of Architectural Analysis Functions	To guarantee continuous service access and consistent operational performance by ensuring the system maintains a high level of reliability and minimal downtime, maximising user access and business continuity.	Medium
NFR-KIO4-09	Architectural Component Versioning for Historical Tracking and Auditability	To enable robust tracking of changes and preserve historical context for architectural components. This aims to facilitate auditability, collaboration and simplifies rollback to previous design states.	Low
NFR-KIO4-10	Logging of Architectural Interactions and Artefacts for Auditing and Process Analysis	To enable comprehensive auditing and continuous system enhancement by securely storing detailed logs of user interactions, generated architectural outputs and their rationales. This fosters accountability and supports data-driven optimisation of architectural processes.	High

ID	Title	Objective	Priority
NFR-KIO4-11	Compliance and Compatibility of Standard Export Formats (UML, PlantUML)	To ensure the architectural artefacts generated by the system are interoperable with third-party tools by adhering to widely adopted, standard formats. This guarantees compliance and facilitates seamless integration into existing development workflows.	Medium

4.1.5 KIO5: Synthetic Data Generation Tool

KIO5 focuses on the generation of synthetic data for use in software engineering scenarios where the development and integration of an AI system is part of the software engineering challenge.

Table 10: KIO5 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO5-01	Collection and aggregation of multimodal geo-localised data into a time- and spatial-indexed Data Lakehouse	The system shall organise geospatial data in a time- and spatial-indexed repository for use in synthetic data generation	High
FR-KIO5-02	Interpolate GIS data in time or in space to deal with missing data or under-resolved data	To analyse the data stored in the Data Lakehouse to interpolate missing or under-resolved data in a way that are consistent with the underlying process.	Low
FR-KIO5-03	Generate a master dataset for medical data	To create a master dataset containing several medical data features by creating synthetic links between atomic public datasets	High
FR-KIO5-04	Evaluation of synthetic data to provide feedback on the generation	KIO6 shall validate the outputs of the generation modules to identify pros and cons.	High
FR-KIO5-05	Report generation to monitor the synthetic data generation metrics and describe the process	Reports shall provide a functional support for the transparency requirement. It shall describe the generation process and provide feedback through metrics.	High

ID	Title	Objective	Priority
FR-KIO5-06	Synthetic network traffic generation capabilities for enhancing Network Intrusion Detection Systems.	KIO5 shall provide network generation capabilities to enhance AI-based anomaly detection/classification/response systems by providing synthetic data for benign and malicious network flows.	High
Non-Functional Requirements			
NFR-KIO5-01	Suitable quality of synthetic network traffic	The network traffic quality must be evaluated to provide benchmarking capabilities, enabling the selection of the most suitable model and transparency to the customer	High
NFR-KIO5-02	Synthetic data generation performance logging for monitoring and benchmarking	This requirement aims to provide information about the generation performance for benchmarking purposes and transparency for the customer	High
NFR-KIO5-04	Synthetic data generation process scalability	The scalability assessment of the generative tasks shall provide information for benchmarking purposes, transparency for the customer	Medium
NFR-KIO5-06	Ensuring synthetic data generation process transparency	The KIO shall provide to any customer the information related to the synthetic data generation process to fully understand how the synthetic dataset is crafted.	High
NFR-KIO5-07	Benchmarking of data collection and aggregation process	The KIO shall provide a benchmark of the data collection and aggregation process performed in FR-KIO5-01.	High
NFR-KIO5-08	Validation of data quality of geographical layers	Evaluate the quality of the data collected by FR-KIO5-1 for benchmarking purpose and to provide transparency about the generated GIS layers to the customer.	High
NFR-KIO5-09	Assessing the quality of generated medical data	Provide an assessment of medical data quality for benchmarking purposes and transparency over the generated data for the customer	High

4.1.6 KIO6: Data Validation-Preparation Tool

KIO6 investigates tool for the validation and training data preparation for AI algorithm training as part of software development tasks in which AI tools need to be customised and adapted for a particular application scenario.

Table 11: KIO6 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO6-01	Orchestration of atomic preparation tools from the validation of the abstract preparation pipeline instructions	Execute the pipeline once it has been validated to perform the automatic data preparation.	High
FR-KIO6-02	Validation of the abstract pipeline for automatic data preparation	Once the abstract pipeline is designed, the instructions must be validated to ensure that the steps describing the pipeline are matching the task requirements and dataset specifications.	High
FR-KIO6-03	Abstract pipeline for automatic data preparation based on NL instructions	The Automated Data Preparation tool must be able to create an abstract preparation pipeline from the input dataset and the ML/DL task description.	High
FR-KIO6-04	Exploratory data analysis for providing metadata about input dataset to the preparation orchestrator	Perform exploratory data analysis (EDA) to provide metadata on the dataset and features to the LLM responsible for preparation pipeline instructions.	High
FR-KIO6-05	Validation of the automatically prepared data	Provide validation capabilities for the prepared data depending on the UC requirements.	High
FR-KIO6-06	Model unit tests to validate the automatically prepared data in terms of ML/DL model fitting	The Model Unit Tester shall verify that the prepared data can fit ML/DL models related to the target task for which it was prepared.	High
FR-KIO6-07	Report generation for preparation, analysis, validation and unit tests	The reports shall provide information about the preparation pipeline which has been used, the validations process and the unit tests. It aims to provide transparency over the data transformation.	Medium

ID	Title	Objective	Priority
FR-KIO6-08	Data warehouse agent for storing raw and prepared datasets according to the use-case requirements	This requirement aims to provide a storage facility compatible with all the project's use-cases.	Medium
FR-KIO6-09	Atomic data preparation tools used by the preparation orchestrator	The toolbox shall provide the preparation capabilities required by the preparation orchestrator to execute the pipeline.	High
FR-KIO6-10	Atomic data preparation tooling specialising in unstructured data for LLMs	This set of atomic preparation tools specialising in data preparation for LLMs shall provide capabilities for the preparation steps targeting LLM's tasks required by every use-case.	Medium
Non-Functional Requirements			
NFR-KIO6-01	Ensure fast preparation process, quantified via performance evaluation	The preparation and validation processes shall be as fast as possible to facilitate the adoption of the tool. Performance evaluation can help for identifying components that would require some optimisation.	Medium
NFR-KIO6-02	Assess the preparation relevance for the targeted task to provide transparency over the preparation quality	This requirement shall verify that the dataset has been prepared accordingly to the targeted dataset.	High
NFR-KIO6-04	Ensure the scalability of preparation and validation processes	The scalability of the preparation and validation processes shall provide information to the customer about transparency over the performance evolution based on the dataset's sizes.	Medium
NFR-KIO6-05	Ensure transparency over the data analysis, preparation and validation processes	Through the report, every detail about data provenance, destination, analysis, preparation and validation shall be provided to the customer.	Medium
NFR-KIO6-06	Ensure the tool's maintainability through a modular structure and open-science practice	A modular structure shall provide the capabilities for efficient modification and enhancements of the preparation/validation framework. Open-science practices shall make the community contribution to the tool possible, to shape it depending on the new emerging requirements.	Medium

4.1.7 KIO7: AI SysDev Tool

KIO7 investigates LLM-based methods for software development, software maintenance and software testing. The aim of the task is to improve automation and AI-supported interactive software development, maintenance and testing.

Table 12: KIO7 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO7-01	Ingestion of Prepared Training Data for ML Model Training	To enable the system to efficiently train data generated by the Data Preparation Tool (KIO6), which is crucial for subsequent machine learning model training and system calibration.	High
FR-KIO7-02	LLM Fine-tuning for Software Development Tasks.	To adapt open-source Large Language Models (LLMs) for specialised software development tasks by fine-tuning them with tailored examples, ensuring high performance and relevance in areas like requirements engineering, architectural planning, code generation and automated testing.	High
FR-KIO7-03	External Knowledge Integration	To enable the system to process extensive input data that exceeds LLM context windows, a Retrieval-Augmented Generation (RAG) strategy will be implemented. This ensures comprehensive data understanding and accurate response generation by integrating external knowledge.	High
FR-KIO7-04	Indexing Multiple Data Types (Code, Text, Snapshots) for External Knowledge Integration	To enable the Retrieval-Augmented Generation (RAG) system to process diverse information sources. This will ensure relevant and comprehensive output generation by supporting multiple data types, including code, text, web page snapshots and user actions.	High
FR-KIO7-05	Generation of Code Snippets and Modules	To accelerate software development by providing code generation capabilities based on fine-tuned LLMs. This will enable the automatic generation of code snippets of entire	High

ID	Title	Objective	Priority
		modules, reducing manual coding effort and improving consistency.	
FR-KIO7-06	AI-based Code Understanding and Debugging Support	To enhance code understanding and streamline the debugging process by providing explanations, identifying issues and suggesting fixes for generated code, ultimately improving developer productivity and code quality.	High
FR-KIO7-07	AI-based Proactive Coding Assistance	To significantly boost developer productivity and elevate code quality by implementing a proactive coding suggestion mechanism. This will deliver real-time recommendations directly within coding sessions, preventing errors and promoting best practices.	High
FR-KIO7-08	Exporting Fine-tuned Models to Standard Formats (ONNX, PyTorch)	To enable broad reuse and seamless integration with other tools by enabling the system to export fine-tuned LLMs in widely adopted standard formats.	High
FR-KIO7-09	Management of Fine-tuned Model Checkpoints for Versioning and Incremental Training	To enable robust development and deployment of LLMs by supporting the storage and retrieval of fine-tuned model checkpoints, facilitating incremental training, efficient versioning and easier recovery or experimentation.	Medium
FR-KIO7-10	Logging of AI System Artefacts for Auditing and Performance Analysis	To enable comprehensive auditing and continuous system improvement by securely storing detailed logs of all user interactions, generated outputs and fine-tuned models, fostering accountability and data-driven optimisation.	Medium
FR-KIO7-11	Generation of Automated Test Suites for Generated Code	To ensure the quality and correctness of generated code by automatically creating a comprehensive test suite. This will enable immediate validation and reduce manual testing effort.	High

ID	Title	Objective	Priority
Non-Functional Requirements			
NFR-KIO7-01	Fine-Tuned Generative Models for code and test generation	To ensure broad applicability and future-proofing, the system must adapt effectively to various software development tasks and domains. This is achieved by robustly fine-tuning its underlying LLMs, enabling relevance across diverse project needs.	High
NFR-KIO7-02	Performance Stability of AI Model Operations Under Concurrent Load	To ensure system reliability and responsiveness under heavy load by enabling it to handle simultaneous requests from multiple users without any degradation in performance. This guarantees a consistent user experience even during peak usage.	Low
NFR-KIO7-03	User-Friendly Interface for AI Model Data and Output Management	To maximise user adoption and efficiency by providing an intuitive and easy-to-use interface. This will enable users to seamlessly input training data and effectively view the generated outputs.	High
NFR-KIO7-04	Modular Integration for Models and Retrieval Methods	To ensure the system's adaptability and future performance enhancement by implementing a plug-in-based architecture. This will enable the seamless integration of new embedding models and RAG strategies, keeping the LLMs at the cutting edge of capability.	High
NFR-KIO7-05	Energy-Efficiency Optimisation for LLM Operations	To significantly reduce the computational cost of fine-tuning and inference for LLMs. This will be achieved by implementing energy-efficient algorithms and techniques (Quantization, Pruning), leading to lower operational expenses and a reduced environmental footprint.	High
NFR-KIO7-06	Inference and Fine-Tuning Time Performance	To ensure system responsiveness and efficiency, the system should complete the computationally intensive fine-tuning and inference processes within a reasonable timeframe, providing a smooth user experience.	Medium

ID	Title	Objective	Priority
NFR-KIO7-07	Error Detection, Logging, and Recovery for AI Operations	To ensure system stability and a positive user experience by implementing robust mechanisms for detecting and gracefully recovering from errors. This minimises disruption and prevents data loss.	Medium
NFR-KIO7-08	Ensuring High Availability and Reliability for AI Model Operations	To guarantee continuous service access and consistent operational performance by ensuring the system maintains a high level of reliability and minimal downtime, maximising user access and business continuity.	Medium
NFR-KIO7-09	High-Accuracy LLM Output	To guarantee the quality and reliability of all LLM-generated outputs by ensuring high accuracy. This is achieved through rigorous benchmarking and the use of comprehensive evaluation metrics to validate the performance of fine-tuned models.	High
NFR-KIO7-10	High Code Coverage Test Generation.	To significantly enhance the quality and reliability of generated code by ensuring the system produces tests with high code coverage. This aims to maximise the validated portions of the code and minimise undetected defects.	High
NFR-KIO7-11	Compliance of Generated Code with Coding Standards and Best Practices	To uphold high code quality and readability, the system must ensure that all generated code is compliant with established coding standards and best practices. This promotes consistency and simplifies future maintenance.	High
NFR-KIO7-12	Secure Code Execution and Sandboxing	To mitigate security risks associated with executing or debugging untrusted code by implementing a sandboxed, isolated environment. This ensures the integrity and security of the host system and prevents arbitrary code execution vulnerabilities.	Medium
NFR-KIO7-13	Artefact Versioning for Maintainability	To enhance the maintainability, reproducibility and auditability of the AI development lifecycle by implementing a comprehensive versioning system for all critical artefacts, including prompts, fine-tuned models and generated code.	Medium

4.1.8 KIO8: AI Tools for Cross-Compilation & Reconfiguration using AI DevOps Components

KIO8 focuses on techniques for cross-compilation and reconfiguration of software to facilitate automated porting of software systems between different types of deployment platforms including, for example, deployment of code to FPGA hardware systems or embedded systems.

Table 13: KIO8 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO8-01	Cross-compilation toolchain configuration for environment adaptability	The cross-compilation toolchain shall be configured according to the target environment.	High
FR-KIO8-02	Software builds through the configured cross-compilation toolchain	From the configured cross-compilation toolchain, the KIO shall build the software according to the target environment specifications.	High
FR-KIO8-03	Report generation to provide transparency about cross-compilation and software reconfiguration tasks	The report shall provide all information concerning the cross-compilation toolchain configuration and execution.	Medium
FR-KIO8-04	Enabling human-in-the-loop to make manual change or request modification on specific parts of the toolchain configuration	Human-in-the-loop shall help to refine the toolchain configuration by providing additional information to the AI for bug fixing.	High
Non-Functional Requirements			
NFR-KIO8-01	Build performance measurement for cross-compilation and toolchain configuration	KIO8 shall monitor the build and configuration performances to provide transparency to the customer about the tool's resource consumption	Medium
NFR-KIO8-02	Ensure transparency of cross-compilation and toolchain configuration processes	Transparency shall be ensured through a report providing information about the toolchain configuration and build process, and through open-science practices.	High

ID	Title	Objective	Priority
NFR-KIO8-05	Ensure security in the software reconfiguration and cross-compilation through input/output sanitization and Least Privilege Principle	Since the tool interacts directly with code, configuration and script generation, particular attention must be paid to the tool in terms of model poisoning, code injection, least privilege principle for code, scripts and BASH commands and vulnerable libraries & functions.	Medium

4.1.9 KIO9: Responsible Innovation and Design-by-Contract Compliant Coding

KIO9 investigates methods to ensure that state-of-the-art AI models including LLMs can follow ethical guidelines in data handling and for use in software engineering tasks that require the system to adhere to ethical guidelines and other types of compliance guidelines.

Table 14: KIO9 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO9-01	Hash-Based Decentralised Code Integrity Verification of the Uploaded Codebase	Verify code integrity and authenticity to detect manipulation or untrusted origins	High
FR-KIO9-02	Enforce Multi-Factor Authentication for Identity Management	Prevent unauthorised access by enforcing strong identity verification	High
FR-KIO9-03	Generate encrypted End-to-End Data Transmission and Storage for Protection Against Threats such as Man-in-the-Middle Attack	Protect all user data and code from interception or tampering during transfer or storage	High
FR-KIO9-04	Input Handling for Codebase and Regulatory Constraint Models	Enable the system to accept and process code and rule-based compliance inputs	High
FR-KIO9-05	Ethical and Legal Compliance Report Generation with Alerts and Recommendations	Provide developers with a summary of compliance results and guidance for remediation	High

ID	Title	Objective	Priority
FR-KIO9-06	User Interface for Code Input, Ethical and Legal Compliance Visualisation, and Secure Access	Provide a graphical user interface to support code input, compliance analysis feedback, and secure collaboration	High
FR-KIO9-07	Orchestrated Ethical and Legal Compliance Flow from Input to Verified Output to Ensure Traceability	Ensure all compliance checks follow a traceable, authenticated, and rule-bound workflow	High
Non-Functional Requirements			
NFR-KIO9-01	Ensuring Efficient Ethical and Legal Compliance Checking and Reporting with Minimal Latency	Ensure timely validation and feedback for AI code compliance	High
NFR-KIO9-02	Ensuring Synchronised Ethical and Legal Compliance Checks Across Distributed Endpoints with Reliable Result Aggregation	Enable use of the tool by large, distributed developer teams	Low
NFR-KIO9-03	Ensuring Energy-Efficient Infrastructure of Decentralised Technologies and Adaptive Security for Data Protection and Authorised Access	Support long-term maintainability and secure usage across project types	Medium
NFR-KIO9-04	Transparent and Traceable Ethical and Legal Compliance Check Results	Ensure users understand how compliance decisions are made without compromising system security as part of disclosures	High
NFR-KIO9-05	Ensure Developer-Oriented Ethical and Legal Compliance Feedback is in Understandable and Actionable Manner	Present results in a format understandable and actionable for developers	Medium
NFR-KIO9-06	Ensuring Security of Data in Transit and Rest, and Authorised Access for Trustworthiness	Ensure secure code validation and result delivery	High

4.1.10 KIO10: Energy-Efficiency with Model-Driven TinyML and LLMs

KIO10 addresses energy-efficient implementation and distribution of AI models for IoT edge devices. Targeted deployment of TinyML-driven AI models to edge devices is implemented with the aim of reducing the overall power consumption of operations by reducing communication and centralised computing overheads in favour of energy-efficient edge device implementations.

Table 15: KIO10 requirements

ID	Title	Objectives	Priority
Functional Requirements			
FR-KIO10-01	Initialise Project from User Intent and Data with Guided Recovery	Enable the system to start a new TinyML project when the user describes the goal and optionally provides sensor data, with automatic error correction and user guidance.	High
FR-KIO10-02	Validate and Auto-Optimise Model for Edge Constraints	Ensure the model fits within memory, energy, and timing limits of edge devices, with automatic adjustment if needed.	High
FR-KIO10-03	Assign Target edge device with Automatic Fallback	Bind the model to a specific edge device, with automatic switch to a compatible alternative if the chosen one cannot support it.	High
FR-KIO10-04	Generate Firmware with Build Error Auto-Repair	Produce complete, working firmware code with automatic correction of common build issues.	High
FR-KIO10-05	Deploy Firmware with Retry, Recovery, and Live Diagnostics	Load firmware onto the device (edge-device) and run it, with automatic retry and debugging on failure.	Medium
Non-Functional Requirements			
NFR-KIO10-01	Enforce On-Device (edge-device) Data Processing with Active Safeguards	Guarantee that no raw sensor data ever leaves the edge device, with real-time monitoring and response.	Medium
NFR-KIO10-02	Continuously Optimise Energy Usage with Adaptive Profiling	Measure energy per inference and automatically suggest or apply power-saving improvements over time.	High

ID	Title	Objectives	Priority
NFR-KIO10-03	Resilient Multi-Device Deployment with Graceful Degradation	Ensure the pipeline works across different edge devices, with automatic adaptation or simulation if hardware fails.	Medium

4.1.11 KIO11: AI-Powered Test Automation Tool

KIO11 works on AI tools to power automated software test creation and execution. Test automation will integrate state-of-the-art open-source testing tools and enable them for use with AI-driven test orchestration.

Table 16: KIO11 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO11-01	AI-Driven Test-case Generation to Enable Dynamic Adaptation to Evolving System Logic	Automatically generate test-cases based on code or model input using AI/LLMs.	High
FR-KIO11-02	Test-case Refinement and Optimisation to Ensure Test Suite Efficiency and Reduce Execution Overhead	Improve test-case quality through prioritisation, pruning, and redundancy removal	Medium
FR-KIO11-03	Automated Test Script Generation to Support Full Automation of Regression and Functional Testing Pipelines	Translate validated test-cases into executable test scripts	High
FR-KIO11-04	Automated Self-Healing Testing to Improve Context Awareness via Contextual AI models, LLMs	Detect and adapt to test failures automatically using contextual AI	Low
FR-KIO11-05	GUI for Test Analysis and Results to Enable Intuitive Review of the Testing Process and Outcomes	Provide a user interface for reviewing test results and analysis	High

ID	Title	Objective	Priority
FR-KIO11-06	Input Readiness for AI-Driven Test Orchestration	To define and formalise the inputs required by each subsystem of the AI-Powered Test Automation Tool (TAT) for automated test generation, execution, and optimisation.	High
FR-KIO11-07	Output Artifacts and Dataflows for Automated Test Execution and Validation	To define and formalise the outputs generated by the AI-Powered Test Automation Tool (TAT) during and after automated test execution, ensuring traceability, visualisation readiness, and reusability across KIO11, KIO13, and WP5 validation workflows.	High
Non-Functional Requirements			
NFR-KIO11-01	Ensuring Low-Latency Execution and Analysis for AI-Driven Test Automation	Ensure fast execution and result delivery for test automation tasks	Medium
NFR-KIO11-02	Ensuring Scalable Multi-Team Access and Distributed Test Execution for AI-Driven Test Automation	Enable reliable access and execution for large, distributed teams	Low
NFR-KIO11-03	Ensuring Long-Term Maintainability through Lightweight Documentation and Adaptive Security	Enable long-term usability and adaptability across varying project criticality levels	Medium
NFR-KIO11-04	Ensuring Transparent and Traceable AI-Driven Test Processes and Audit Trails	Make backend processes and decisions observable to users	High
NFR-KIO11-05	Ensuring User-Centric Visual Feedback and Guidance across All Interfaces	Improve usability and interpretation of test results for all user roles	Medium
NFR-KIO11-06	Ensuring Secure and Configurable End-to-End Protection for Test Automation Environments	Ensure secure execution, storage, and access within the TAT environment	Medium

4.1.12 KIO12: AI-Assisted Development of AI-Based Cybersecurity & Multi-Level Security by Design

KIO12 embeds Ai techniques to incorporate AI-supported security-by-design into the software engineering process. The KIO investigates AI-enabled cybersecurity measures across the full DevOps lifecycle.

Table 17: KIO12 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO12-01	Generate Structured and Standardised Reports After Test Execution	To ensure that system outputs, including security test reports, attack detection results, and training materials, are consistently structured, verifiable, and usable for validation, auditing, and knowledge transfer activities, including as input for downstream toolchains like KIO13. Test reports should be user-friendly and use standard machine- and human-readable output formats.	High
FR-KIO12-02	AI-driven Intrusion Detection for Monitoring Network-level and Application-level Intrusions to Enhance Cyber Resilience	To enhance the system cyber resilience by detecting and responding to cybersecurity threats in real time.	High
FR-KIO12-03	Security Test Scenario Generation for Exposed APIs to Strengthen the Cyber Resilience	To strengthen the system's cyber resilience by enabling automated and consistent security testing of exposed APIs, as part of a proactive vulnerability mitigation strategy.	High
FR-KIO12-04	Vulnerability Assessment of API Endpoints and Backend Components to Maintain Cybersecurity and System Integrity	To ensure that the system is systematically scanned and evaluated for known vulnerabilities, enabling early detection and remediation to maintain cybersecurity and system integrity throughout the development lifecycle.	High
FR-KIO12-05	Real-world Cyber Attack Scenario Analysis to Validate System Robustness	To enhance the system's resilience by modelling, simulating, and analysing real-world cyberattack scenarios that target software and hardware layers, enabling proactive risk mitigation and robust design.	High

ID	Title	Objective	Priority
FR-KIO12-06	End-to-End Secure Data Handling and Multi-Factor Authentication to Ensure Protection of Sensitive Data and Identity Verification	Ensure robust end-to-end protection of sensitive data through encryption and identity verification	High
FR-KIO12-08	UI for Vulnerability Assessment and Developer Guidance	Provide an intuitive and interactive interface for vulnerability analysis, rule visibility, and actionable developer support.	High
FR-KIO12-09	Input Data Integration for Vulnerability Assessment and Scenario Generation	Ensure the system accepts and integrates all required technical inputs for AI-based vulnerability detection and secure test generation	High
Non-Functional Requirements			
NFR-KIO12-01	Accurate Test Scenario Generation and Vulnerability Detection	Ensure accurate generation of test scenarios and detection of security vulnerabilities	High
NFR-KIO12-02	Ensuring Scalability of the Platform to Support Evolving Threat Models and Distributed Teams	Ensure scalability of the platform to support evolving security needs and large user bases	Medium
NFR-KIO12-03	Supporting Long-term Maintainability and Adaptability Across Configurable Security Modes	Support long-term maintainability and adaptability across diverse development contexts	Medium
NFR-KIO12-04	Ensuring Transparency of Backend Logic for User Awareness	Make the testing and detection process visible and interpretable to users	High

4.1.13 KIO13: Developer Training for Using AI-Powered Software Development Tool

KIO13 designs and implements the developer training tool for AI4SWEng. The tool will teach software engineers how to use modern AI tools and the AI4SWEng tool suite in particular to improve their output in terms of quality and time to completion.

Table 18: KIO13 requirements

ID	Title	Objective	Priority
Functional Requirements			
FR-KIO13-01	Handle User Inputs for Ontological Commitments in Prompt Engineering	To enable trainees to define ontological elements that underpin effective prompt design in AI-supported development.	High
FR-KIO13-02	Teach Deriving Formal Requirements from NLP-Processed Inputs	To help learners convert everyday language into solid specs using NLP, tying it all back to ontological basics	High
FR-KIO13-03	Provide Tools for Building, Editing, and Visualising Ontologies	To give learners easy-to-use tools for creating and seeing ontologies, helping with requirement setup and timing	High
FR-KIO13-04	Deliver Structured Multimedia and Interactive Training for Prompt Engineering	To offer solid training resources and practice sessions for getting good at prompt engineering.	High
FR-KIO13-05	Assess LLM Outputs and User Prompt Variants	To let learners review and compare what LLMs produce, improving their skills through feedback.	High
FR-KIO13-06	Teach End-to-End AI Tool Integration in Agile Pipelines	To guide learners through connecting AI tools for the whole dev process in agile setups.	High
FR-KIO13-07	Track User Training Progress and Skill Application	To keep an eye on how learners are advancing and applying what they've learned.	Medium
FR-KIO13-08	Supply Comprehensive Step-by-Step User Guides in Multiple Formats	To provide handy guides for getting the most out of the system and its integrations.	Medium
FR-KIO13-11	Provide Hands-On Training Simulations	Enable hands-on practice for prompt engineering and ontological training	High
FR-KIO13-12	Provide Comprehensive User Companion Guide	Deliver a detailed guide to support developers in using the AI-powered training platform	High

ID	Title	Objective	Priority
Non-Functional Requirements			
NFR-KIO13-01	Ensure Intuitive Usability and Accessibility	To provide an inclusive and user-friendly training experience.	High
NFR-KIO13-02	Guarantee High Availability with Data Backup for User Ontologies and Prompts	To keep the training platform reliable with safe data handling.	Medium
NFR-KIO13-03	Enforce GDPR-Compliant Data Protection for User Inputs	To safeguard learner info while following privacy rules.	High
NFR-KIO13-04	Achieve Scalability and Resource Optimisation for Concurrent Users	To manage more users smoothly without wasting resources.	Medium
NFR-KIO13-05	Maintain Compatibility and Easy Updates for AI Tools and Content	To keep things working with other tools and simple to refresh.	Medium
NFR-KIO13-06	Deliver Real-Time Performance for Multimedia and Visualisations	To make sure interactions feel quick for agile learning.	Medium
NFR-KIO13-11	Enhance Global Accessibility with Multi-Language Support	Support global adoption of the training platform	Low

4.1.14 Summary

In total, 185 KIO requirements have been defined across the 13 KIOs of the project, of which 101 are functional requirements and 84 are non-functional requirements. Every requirement has been updated as part of the revision process. 63 cross-references between the KIO requirements and the security, traceability and privacy-by-design requirements introduced in Section 2.6 have been introduced.

4.2 Use-Case Requirements

As shown in Figure 3, use-case requirements were elicited separately from KIO requirements and after the initial KIO requirement specification was concluded. The

rationale for this was to be able to focus the requirements for use-cases so that they and the resulting use-case specifications can be employed to validate the KIO outcomes in realistic scenarios.

UC requirements are specified in the same format as KIO requirements, but it is important to remain aware that the UC requirements are requirements for the output that is to be created using the AI4SWEng tool suite or parts of it: the specified requirements for use-cases are to be met by the related product created using the AI4SWEng tool suite.

4.2.1 UC1: Optimising Software Development Lifecycle Using Large Language Models

Use-Case 1 investigates the overall software development lifecycle and applies the AI4SWEng tool suite to optimise a standard software development project from end to end.

Table 19: UC1 requirements

ID	Title	Objective	Priority
Functional Requirements[4]			
FR-UC1-01	Suggestion of Energy-Efficient Frameworks and Languages for New Projects	To enable the system to suggest low-consumption frameworks and languages for new projects, considering project needs. This ensures the foundation of the project is built with energy efficiency in mind, aligning with the project's goal of energy saving.	High
FR-UC1-02	Comparative Analysis of Technology Trade-offs (Performance vs. Memory)	To enable the system to perform a trade-off analysis between technologies. This ensures that technology selection is based on a clear understanding of the compromises between performance and memory, aligning with the project's energy-saving goals.	High
FR-UC1-03	Ingestion of Historical Project Artifacts (Code, Tickets, Tests)	To enable the system to efficiently process and use all relevant information from an existing project as context at the beginning of each phase. This ensures the LLM generates artefacts that are consistent with the existing codebase and project history.	High
FR-UC1-04	Organisation and Storage of Ingested Project Context	To enable the system to efficiently organise and store project context for use in subsequent analysis and generation steps. This ensures that the LLM has all the necessary information readily available to produce relevant and consistent outputs.	High
FR-UC1-05	Validation of Ingested Project Context (Tickets, Code)	To ensure system reliability by validating that the gathered context data is complete and usable before it is used in subsequent analysis and generation steps. This prevents errors and ensures that the LLM has all the necessary information to produce accurate outputs.	High

[4] UC1 has not specified non-functional requirements at this time. Additional non-functional requirements are expected to be provided with a later deliverable.

ID	Title	Objective	Priority
FR-UC1-06	Anonymisation of Sensitive and Proprietary Context Data	To ensure system compliance with security and privacy policies by anonymising or masking any sensitive or proprietary information found in the gathered context data. This protects user privacy and project confidentiality while still allowing the data to be used for analysis.	High
FR-UC1-07	Secure Storage (Encryption) and Retention Policy for Context Data	To ensure the system securely stores all gathered context data and defines a clear retention policy for historical data. This protects against unauthorised access and ensures compliance with data privacy regulations.	High
FR-UC1-08	Natural Language Task Ingestion from PM	To enable the system to ingest and process a high-level task in natural language provided by a Project Manager (PM). This serves as the primary input for the automated generation of project artefacts, streamlining the initial specification phase.	High
FR-UC1-09	Optional Design Stage Trigger for a Given Requirement	To provide the PM with the option to trigger a design stage for complex or specific requirements before proceeding with issue creation. This ensures that good code practices are considered and a more detailed plan is in place for challenging tasks.	Medium
FR-UC1-10	Provision of Task Metadata (Importance, Tags) by PM	To enable the system to accept additional metadata from the PM for a task. This included information such as importance, preferred programming language, or relevant tags, which helps the LLM generate more precise and well-organised issues.	High
FR-UC1-11	LLM-based Issue Generation	To enable the system to use an LLM to analyse user input and generate a draft issue based on predefined templates, guidelines and similar issues. This automates the creation of project tasks, ensuring consistency and adherence to project standards from the outset.	High

ID	Title	Objective	Priority
FR-UC1-12	Suggestion of Automated Tests (Unit, Integration, E2E)	To enable the system to suggest tests that should be implemented to verify that the code meets the requirements. This automates the initial phase of quality assurance, ensuring that a plan for validating the code is in place from the beginning.	High
FR-UC1-13	Optional Generation of Code/Pseudocode Starting Points	To provide developers with an optional first draft of a code solution or pseudocode, enabling faster initial development and providing a starting point for implementation.	Medium
FR-UC1-14	Issue Review, Editing and Approval	To ensure the accuracy and completeness of generated issues by allowing PMs to review, edit and approve them. This provides a crucial human feedback loop, ensuring the final issue is correct and ready for development.	High
FR-UC1-15	Centralised Management of Customised Issue Templates	To enable the system to have a central place to manage and update issue templates. This ensures consistency in the structure and content of generated issues across different projects, enhancing standardisation and collaboration.	Low
FR-UC1-16	Feedback Integration for Improved Issue Generation	To enable the system to incorporate feedback from PM edits into future issue generation, thereby improving accuracy and consistency over time. This ensures the LLM learns from human corrections, leading to more precise and higher-quality outputs that align better with project-specific needs.	High
FR-UC1-17	Development Environment Setup and Verification	To ensure system reliability by verifying that the development environment is ready and up to date before code generation or other tasks. This includes checking framework versions, dependencies, and configurations to prevent errors and ensure a smooth development process.	High
FR-UC1-18	LLM-based Code Generation from Approved Issues	To enable the system to use the LLM to generate code for the approved issue, including functions, classes, and supporting components. This	High

ID	Title	Objective	Priority
		automates the coding process, improving efficiency and consistency.	
FR-UC1-19	Applying Energy-Efficiency Guidelines during Code Generation	To ensure the system follows predefined energy-efficiency guidelines during code generation, such as avoiding heavy loops or unnecessary UI refreshes. This minimises computational cost and aligns with the project's goal of reducing energy consumption.	High
FR-UC1-20	Review and Validation of Code against Requirements and Standards	To provide a mechanism for developers to review the generated code and ensure it meets requirements, adheres to coding standards, and is functionally correct. This step is crucial for quality control and for catching bugs or deviations from the approved issue.	High
FR-UC1-21	Optional LLM Self-Review of Generated Code	To provide an optional "self-review" by the LLM on the generated code, acting as a second pass to flag potential issues. This process helps to detect problems such as high complexity, security vulnerabilities, or licence conflicts before the code is reviewed by a human team.	Medium
FR-UC1-22	Enforcing Code Quality using Static Analysis (Linters, SAST)	To integrate static analysis tools such as linters and security scanners into the system. This helps enforce minimum quality thresholds and ensures the code meets predefined standards before it's approved.	High
FR-UC1-23	LLM-based Generation and Update of Automated Tests	To enable the system to use the LLM to generate or update automated tests based on the approved issue and the generated code. This automates the quality assurance process by ensuring the code has comprehensive test coverage, reducing manual effort for the development team.	High
FR-UC1-24	LLM-based Generation of Parallel Code Documentation	To enable the system to use the LLM to generate or update code documentation, API documentation, and inline comments in parallel with code creation. This ensures that the documentation is up-to-date and	High

ID	Title	Objective	Priority
		consistent with the code, reducing manual effort and improving maintainability.	
FR-UC1-25	Automated Verification of Documentation Consistency (against Code)	To ensure the generated documentation is consistent with the final code through automated checks. This guarantees that documentation is accurate and reflects the current state of the code, improving maintainability and reducing manual effort.	High
FR-UC1-26	Centralised Guideline Management to Ensure LLM Consistency	To provide a central place to manage and update code templates, coding standards, and test guidelines across projects. This ensures consistency, standardisation, and adherence to best practices throughout the entire development lifecycle.	Medium
FR-UC1-27	Automated Code Compliance Verification (Linters, SAST, License)	To ensure the system performs an automated compliance verification process on the completed code to guarantee adherence to standards. This helps to reduce bugs and human errors by checking the code against predefined guidelines before it is deployed.	High
FR-UC1-28	Execution of Automated and Existing Tests to Verify Code Correctness	To enable the system to execute all generated and existing tests to verify the correctness and functionality of the code. This step is crucial for ensuring code quality and reducing bugs before deployment.	High
FR-UC1-29	Blocking Code Merge to Main Branch on Compliance/Test Failure	To prevent the introduction of bugs and non-compliant code into the main codebase by blocking merges until all compliance checks and tests pass successfully. This enforces a high standard of quality before deployment.	Medium
FR-UC1-30	Automated Code Merging (to Main) and Deployment (to Production)	To enable the system to automatically merge and deploy code once all gates are cleared. This streamlines the final steps of the development lifecycle, ensuring a rapid and reliable transition to the production environment.	Medium

ID	Title	Objective	Priority
FR-UC1-31	Post-Merge Rollback Mechanism for Performance Regression	To provide a rollback mechanism if post-merge gate criteria fail. This ensures that any accidental degradation in runtime, memory, or energy is quickly reversed, maintaining system stability and performance.	Medium
FR-UC1-32	Profiling of Code for Runtime, Memory, and Energy Consumption	To enable the system to profile runtime, memory, and energy consumption of the generated code. This ensures that these metrics have not worsened significantly or unintentionally, maintaining the project's focus on efficiency and energy savings.	High
FR-UC1-33	Management of Configurable Compliance Rule Sets (GDPR, ISO)	To enable the system to maintain a configurable compliance rule set for different projects or regulatory requirements (e.g., GDPR, ISO standards). This ensures that the generated code and project artefacts adhere to a variety of standards and guidelines, enhancing security and reducing legal risks.	High
FR-UC1-34	Version Control for Compliance and Test Reports	To enable the system to store and version control all compliance reports and test results for auditing purposes. This ensures a complete and traceable history of quality and compliance, which is critical for accountability and future analysis.	Medium
FR-UC1-35	Post-Release Monitoring for Performance Regressions (Runtime, Memory, Energy)	To enable the system to monitor the deployed code post-release to detect any performance regressions that were missed during pre-deployment testing. This ensures continuous performance and helps in maintaining the quality of the software in the production environment.	Medium

4.2.2 UC2: Automating Medical Report Generation Using Large Language Models

Use-Case 2 challenges the project with a task in the medical domain and focuses on AI4SWEng capabilities for developing AI models as key component of software solutions. The target system to be created is intended to automatically generate medical reports.

Table 20: UC2 requirements

ID	Title	Objectives	Priority
Functional Requirements			
FR-UC2-01	Role-Based Access Control (RBAC) System to Enforce Permissions for "Patient" and "Clinician" Roles	To ensure system security and data privacy by restricting access to features and data based on clearly defined user roles (e.g., Patient, Clinician).	High
FR-UC2-02	Role-Specific User Interface Provisioning	To provide distinct, role-appropriate user interfaces for different user types, specifically a "Patient Portal" for patients and a "Clinical Workstation/Dashboard" for clinicians.	High
FR-UC2-03	Natural Language Processing for Medical Data Extraction	To automatically parse and extract structured medical information (e.g., symptoms, diagnoses, medications, procedures) from unstructured plain-text inputs such as clinical notes.	High
FR-UC2-04	Input Data Validation and Ambiguity Resolution	To ensure the accuracy, completeness, and integrity of all data entered into the system, and to proactively seek clarification for ambiguous, invalid, or incomplete information.	High
FR-UC2-05	Dynamic Report Architecture Planning and Template Selection	To flexibly configure the system's data processing and report generation architecture based on a user's selection of a report template and the available data feature set for a specific case.	High
FR-UC2-06	AI-Assisted Report Generation via Large Language Model	To leverage a Large Language Model (LLM) to assist clinicians in generating comprehensive medical reports efficiently.	High
FR-UC2-07	Automated Compliance Auditing System Adhering to Privacy Laws and Healthcare Regulations	To continuously and automatically ensure that all system operations, data handling, and generated reports adhere to relevant privacy laws (GDPR, KVKK, HIPAA), medical standards, and healthcare regulations.	High
FR-UC2-08	Human-in-the-Loop Report Approval Pipeline	To ensure that 100% of AI-assisted reports are reviewed, edited, and explicitly approved by a qualified a clinician before they are considered finalised and stored.	High

ID	Title	Objectives	Priority
FR-UC2-09	Secure Storage and Transmission of Medical Reports and Data	To protect the confidentiality, integrity, and availability of all patient data and generated reports, both when at rest and when being transmitted.	High
Non-Functional Requirements			
NFR-UC2-01	System Usability and Interoperability: Ease of Use for User Roles and Compatibility with Relevant Medical Systems	To ensure the system is intuitive and easy to use for all defined user roles, regardless of their technical skill set, minimising the need for extensive education or training. The system must also be capable of exchanging data with other relevant medical systems.	Medium
NFR-UC2-02	Information Security: Data Confidentiality, Integrity, and Compliance for Personal and Medical Information	To maintain the highest standards of data confidentiality, integrity, and compliance for all personal and medical information, complementing the functional security requirements.	High
NFR-UC2-03	AI Governance and Auditability for Report Generation Processes	To establish clear governance, transparency, and auditability for all AI-assisted report generation processes.	High
NFR-UC2-04	System Performance and Reliability: Responsive, Available, and Reliable AI-assisted Report Generation	To ensure the system is responsive, available, and reliable, especially during critical clinical-use periods, and that AI-assisted generation is completed in a timely manner.	Medium

4.2.3 UC3: Large Language Models for Advanced Software Generation for FPGAs

Use-Case 3 focuses on an example of a particular type of software development activities with the use of the AI4SWEng toolset for specialist deployment platforms. In this use-case, software development for FPGA platform targets is investigated.

Table 21: UC3 requirements

ID	Title	Objectives	Priority
Functional Requirements			
FR-UC3-01	Iterative code generation	The system should be able to continuously generate code until certain conditions are met, based on the toolchain response	High
FR-UC3-02	HLS-Script creation for toolchain configuration	The system should be able to manage a compilation script that manages the toolchain necessary for HLS related processes.	High
FR-UC3-03	Comprehensive final report generation	The system should have a final output documenting the various aspects of the HLS designed system (occupation, latency, throughput, etc.)	Medium
FR-UC3-04	Interaction with toolchain/script	The system must handle a compilation script that governs the HLS toolchain. Then it must receive its outputs and correct the already generated code.	High
FR-UC3-05	High-Level Synthesis (HLS) Program Generation and Analysis	Support the generation and comprehension of HLS-based source code	High
FR-UC3-07	Generating summary of toolchain outputs to avoid wasting the code generation LLM context	To save up code generating LLM context by reducing toolchain outputs.	Medium
FR-UC3-08	Stepwise expert-in-the-loop in code generation	To allow an expert to view the whole process of code generation and stop it or steer the process via prompt injection.	High
FR-UC3-09	Convergence criteria definition	There should be a way of defining convergence criteria. Having functional HLS code is not sufficient, we may want a design that is "as small as possible" or that "minimises power consumption".	High

ID	Title	Objectives	Priority
Non-Functional Requirements			
NFR-UC3-10	Final report is transparent and shows the execution details	The goal is for the final report to clearly present which decisions were made during execution regarding architecture and code design, as well as a detailed summary of the HLS project that was generated	Medium

4.2.4 UC4: Next-Generation Battery Management in EVs with AI-Driven TinyML Solutions

Use-Case 4 specifically addresses the development and deployment of systems that combine machine learning capabilities with specialised hardware environments, using the example of battery management in EVs. The UC will focus on AI-driven software engineering to achieve these tasks and on incorporating approaches that incorporate the energy consumption targets into the software engineering process.

Table 22: UC4 requirements

ID	Title	Objectives	Priority
Functional Requirements			
FR-UC4-01	Initialise Battery RUL Project from Multi-Source Data	Enable engineers to define a battery RUL prediction task using time-series sensor data and unstructured maintenance logs.	High
FR-UC4-02	Construct Multi-Modal AI Model	Build a hybrid model integrating graph, convolutional, recurrent, and language processing for battery health analysis.	High
FR-UC4-03	Configure Federated Learning with Privacy Safeguards	Enable multiple vehicles to collaboratively improve the model while keeping all raw data local.	High
FR-UC4-04	Embed Anomaly Detection and Self-Repair	Detect battery anomalies and automatically initiate model correction in real time.	High

ID	Title	Objectives	Priority
FR-UC4-05	Deploy Self-Adaptive RUL System	Run the full multi-modal, federated, self-healing model on the vehicle with continuous adaptation.	High
Non-Functional Requirements			
NFR-UC4-01	Zero Raw Data Transmission with Federated Learning	Ensure no voltage, current, or temperature samples ever leave the vehicle.	High
NFR-UC4-02	Achieve Ultra-Low-Power Inference	Minimise energy per RUL prediction to enable always-on monitoring.	High
NFR-UC4-03	Cross-BMS Deployment Compatibility	Support deployment on different EV battery management systems with consistent behaviour.	High

4.2.5 UC5: AI-Powered GIS Integration for Smart Energy Management

Use-Case 5 investigates the development of energy-efficient solutions that involve data fusion and predictive modelling using the AI4SWEng tool suite.

Table 23: UC5 requirements

ID	Title	Objectives	Priority
Functional Requirements			
FR-UC5-01	Optimal GIS and framework selection for new developments	We shall leverage AI4SWeng to suggest optimal frameworks for new developments, taking into account the user's high-level requirements and the pre-existing codebase.	High
FR-UC5-02	Natural Language Inputs from Product Manager for Requirement Specification	To enable the system to ingest and process a high-level requirement in natural language provided by a Product Manager or from user interviews. This serves as the primary input for the automated generation of project artifacts, streamlining the initial specification phase.	High

ID	Title	Objectives	Priority
FR-UC5-03	Interface for High-level contextual development constraints provided by the Product Manager	To enable the system to accept high-level contextual constraints for a requirement, often provided by the Product Manager. This includes metadata such as requirement priority, preferred programming language, security constraints, performance objectives, and relevant tags, which helps the LLM generate more precise, well-organised issues.	High
FR-UC5-04	LLM-based User Story Generation	To enable the system to use an LLM to analyse user input and generate a draft user story based on natural language descriptions of requirements for tasks like demand prediction, grid impact analysis, or hydro/solar/wind integration support, including predictive capabilities such as production/demand forecasts for various installations across the territory.	High
FR-UC5-05	Automated Generation of Documentation on Functional Requirements	To automatically generate comprehensive functional requirement documents for generated user stories.	High
FR-UC5-06	Automated Generation of Acceptance Criteria for Generated User Stories	To automatically define clear and measurable acceptance criteria for generated user stories.	High
FR-UC5-07	Automated Identification of Necessary GIS Datasets for User Story Implementation	To identify necessary geospatial datasets and mechanisms to collect, preprocess, and aggregate those in a time- and spatial-indexed data lake.	High
FR-UC5-08	Interface for User Story Review, Editing, and Approval	To ensure the accuracy and completeness of generated user stories by enabling the Product Managers (PMs) to review, edit, and approve them. This provides a crucial human feedback loop, ensuring the final user story is correct and ready for development.	High
FR-UC5-09	Feedback Integration for Improved User Story Generation	To enable the system to incorporate feedback from Product Manager (PM) edits into future user story generation, thereby improving accuracy and consistency over time.	High

ID	Title	Objectives	Priority
		This ensures the LLM learns from human corrections, leading to more precise, higher-quality outputs that better align with project-specific needs.	
FR-UC5-10	Setup and Verification of the Development Environment	To ensure the system's reliability by verifying that the development environment is ready and up to date before code generation or other tasks. This includes checking framework versions, dependencies, and configurations to prevent errors and ensure a smooth development process.	High
FR-UC5-11	LLM-based Code Generation Aligned with User Story Specifications	To enable the system to use the LLM to generate code for the approved user story, including functions, classes, and supporting components. This automates the coding process, improving efficiency and consistency.	High
FR-UC5-12	Integrated Interface for Code Validation and Peer Review	To ensure the system enables the development team to review and validate the code generated by the LLM.	High
FR-UC5-13	Automated Test Generation for Code Verification and Functional Validation	To enable the system to use the LLM to generate automated tests based on the approved user story and the generated code. This automates the quality assurance process by ensuring the code has comprehensive test coverage, reducing manual effort for the development team.	High
FR-UC5-14	Test Execution and Verification	To enable the system to execute all generated and existing tests to verify the code's correctness and functionality. This step is crucial for ensuring code quality and reducing bugs before deployment.	High
FR-UC5-15	Report Generation for Compliance Verification and Testing Results	To enable the system to use the LLM to generate a detailed report on the test outcomes, indicating which tests passed, failed, or were skipped.	High

ID	Title	Objectives	Priority
Non-Functional Requirements			
NFR-UC5-01	Intuitive Input and Output User Interaction	To maximise user adoption and efficiency by providing an intuitive and easy-to-use interface as an integrated VS Code plugin/extension. This will enable the users to seamlessly input natural language descriptions and effectively view all generated project outputs.	High

4.2.6 Summary

In total, 81 use-case requirements have been identified, 72 of which are functional requirements and 9 of which are non-functional requirements.

Section 5 continues the definition of the project use-cases and presents the detailed use-case specifications for each of the five use-cases.

5 Use-Case Specifications

5.1 Introduction

Building upon the consolidated, conflict-free requirement hierarchy established in Chapter 4 through the rigorous application of the UI-REF Framework described in Chapters 2 and 3, and in full alignment with the overarching project scope and methodological foundations presented in Chapter 1, the present chapter delineates the foundational validation strategy for the AI4SWEng project, centring on five diverse Use-Cases (UCs) that operationalise the project mission to develop and validate AI-powered tools for efficient, resilient, and ethical software engineering across the complete development lifecycle. These use-cases are derived directly from the project problem statement and methodology, underscoring agile, energy-efficient practice with integrated support for requirements engineering, architecture planning, code generation, testing automation, and continuous optimisation. By harnessing the KIOs, notably the unifying AI Engineering Suite (KIO1), the UCs illustrate how AI4SWEng tackles real-world challenges in multi-architecture systems, encompassing security-by-design, privacy protection, and compliance with regulations such as the Artificial Intelligence Act (AI-Act) [5], Data-Act [6], General Data Protection Regulation (GDPR) [4], the AI, Data and Robotics Association (ADRA) Framework [10], Z-inspection® [11] and US Health Insurance Portability and Accountability Act (HIPAA) [12].

The use-cases are categorised as follows to ensure broad applicability and domain-specific profundity:

- **Horizontal use-cases** (UC1 and UC3) furnish cross-industry demonstrations of the platform core capabilities in general software development and high-performance computing environments, such as optimising the Software Development Lifecycle (SDLC) and generating hardware-aware code for Field Programmable Gate Arrays (FPGAs). These emphasise scalability, reusability, and integration into standard DevOps pipelines, showcasing the platform versatility under various domain-specific constraints.
- **Vertical use-cases** (UC2, UC4, and UC5) target specialised sectors-healthcare, electric vehicle (EV) edge computing, and smart energy management-wherein stringent requirements for accuracy, regulatory compliance, energy efficiency, and data privacy prevail. These use-cases incorporate federated learning, TinyML optimisations, and Geographical Information System (GIS) integrations to validate the platform robustness in regulated, resource-constrained settings.

Each UC is subdivided into 2–5 executable use-case scenarios, adhering to a standardised template informed by the project agile methodology. This decomposition ensures traceability to functional requirements, explicit mapping to KIOs, and measurable Key

Performance Indicators (KPIs) aligned with the DevOps evaluation framework. Scenarios are initiated by natural-language inputs via KIO1 Graphical User Interface (GUI) or Command-Line Interface (CLI), incorporating Human-in-the-Loop (HITL) oversight at pivotal decision points to support compliance with Responsible-AI principles (e.g., via KIO9) from the outset. The interdependencies amongst use-cases and KIOs are visualised in Figure 5, highlighting primary flows for orchestration and feedback loops.

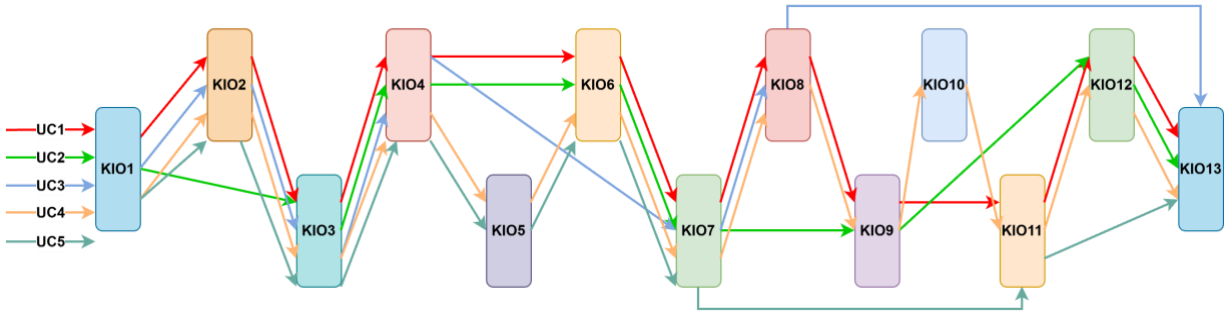


Figure 5: Use-Cases × KIOs Interaction Diagram

5.2 Use-Case Overview and Mapping to KIOs

Table 24 below summarises the use-cases (UCs), their alignment to project objectives, and primary KIO involvements, verified against the KIO descriptions, DevOps methodology, and the provided interaction diagram. KIO10 is confined to UC4, as it pertains specifically to energy-efficient TinyML in resource-constrained EV environments.

Table 24: Use-cases to KIOs Mapping

Use-case	Type	Lead Partner	Primary Domain	Key Challenges Addressed	Primary KIOs Involved (Verified Mapping)
UC1	Horizontal	IDE	General SDLC optimisation	Effort-intensive requirements, code/test generation, energy-efficient practices	KIO1→ KIO3→ KIO6→ KIO5→ KIO4→ KIO7→ KIO12 → KIO9→ KIO11→ KIO2→ KIO8→ KIO1
UC2	Vertical	BIT	Healthcare report automation	Time-consuming documentation, HIPAA/GDPR compliance, data accuracy in electronic health records (EHRs)	KIO1→ KIO3→ KIO6→ KIO4→ KIO7→ KIO9 → KIO6→ KIO12→ KIO1

Use-case	Type	Lead Partner	Primary Domain	Key Challenges Addressed	Primary KIOs Involved (Verified Mapping)
UC3	Horizontal	ITCL	FPGA software generation (C++/HLS)	Iterative synthesis/timing closure, hardware resource optimisation	KIO1→ KIO3→ KIO4→ KIO7→ KIO8→ KIO2→ KIO1
UC4	Vertical	UREAD	EV battery Remaining Useful Life (RUL) prediction (TinyML)	Edge privacy/federation, low-power inference, anomaly detection in multi-sensor data	KIO1→ KIO3→ KIO10→ KIO4→ KIO5→ KIO6→ KIO7→ KIO10→ KIO11→ KIO2→ KIO9→ KIO12→ KIO8→ KIO1
UC5	Vertical	HESSO	GIS-based smart energy optimisation	Real-time network balancing, renewable forecasting, end-to-end validation in digital twins	KIO1→ KIO3→ KIO5→ KIO6→ KIO4→ KIO7→ KIO2→ KIO1

5.3 Detailed Use-Case Scenarios

This subsection presents the detailed UC scenarios for the five use-cases. Each of the scenarios is coupled with a corresponding test-case, which is presented in Section 5.

5.3.1 UC1 - Optimising Software Development Lifecycle Using Large Language Models (Horizontal)

This horizontal use-case streamlines the end-to-end SDLC for software professionals, reducing manual effort in requirements-to-deployment whilst enforcing consistency and energy savings. It integrates historical data from tracking systems (e.g., Jira, GitHub) and promotes Agile System Development practices such as automated issue labelling and test generation.

Table 25: UC-UC1-01: Natural-Language Customer Request to Draft Issue and Requirements

ID	UC-UC1-01
Name	Natural-Language Customer Request to Draft Issue and Requirements
Description (short)	This scenario transforms a natural-language customer or stakeholder request into a formalised set of requirements and a fully structured draft development issue. Using KIO1 and KIO3, the system extracts functional and

	non-functional requirements, acceptance criteria, tags, priority, story points, and initial test outlines. HITL refinement is supported through the KIO1 GUI.
Actor(s)	Project managers, product owners, software architects, developers
Trigger	A natural-language customer request is submitted via the KIO1 GUI or directly to KIO3.
Pre-condition(s)	• Historical project context ingested (tickets, code, tests) • Issue templates and tagging guidelines configured. • Requirements schema available • Label taxonomies, priority rules, and estimation heuristics loaded
Input	Natural-language customer request
Output	• Structured functional and non-functional requirements. • Acceptance criteria • Draft development issue with metadata (priority, labels, size/story points) • Initial test ideas
Post-condition(s)	• Requirements and issue draft stored in project context. • Artefacts ready for validation (UC-UC1-02)
Sequence	1. User submits a natural-language customer request via KIO1. 2. KIO3 extracts intents, functional requirements, and constraints. 3. KIO3 generates acceptance criteria following the project schema. 4. KIO3 creates a structured issue (priority, tags, story points). 5. The draft is displayed in KIO1 GUI for HITL feedback. 6. User approves or refines the issue and requirements. 7. KIO1 stores the result in the requirements repository.
Related KPIs	KPI 1.2 - Faster Issue Resolution KPI 3.2 - Review Score Increase KPI 5.1 - Time-to-Market KPI 8.1 - Adoption Rate KPI 8.2 - Active Usage & Satisfaction
Related KIOs	KIO1, KIO3
Related Requirements	FR-UC1-04, FR-UC1-05, FR-UC1-06, FR-UC1-09, FR-UC1-10, FR-UC1-27
Validation metrics	• Completeness of requirements (fields present) • Number of HITL corrections • Time from request submission to draft generation
Interdependencies	Successor: UC-UC1-02

Table 26: UC-UC1-02: Automated Code and Test Generation from Approved Issue

ID	**UC-UC1-02**
Name	Automated Code and Test Generation from Approved Issue
Description (short)	This scenario generates production-ready code, unit tests, integration tests, and related documentation directly from an approved issue. KIO7 is responsible for LLM-based code and test generation, KIO4 for architectural scaffolding, KIO6 for validating artefact-requirement consistency, and KIO9/KIO12 for ethical/compliance and security checks.
Actor(s)	Developers, technical leads, QA engineers
Trigger	An issue approved in UC-UC1-01 is selected for code generation.
Pre-condition(s)	• Approved issues with linked requirements • Architectural templates and guidelines available • Project language and framework configured. • Validation schema for code/tests available
Input	• Approved issue ID • Linked requirements and acceptance criteria

Output	• Generated source code (aligned with architecture) • Unit and integration tests • Documentation, comments, and design notes • Compliance and security validation reports
Post-condition(s)	• Codebase ready for testing and integration • All artefacts traceable to requirements
Sequence	1. Developer selects an approved issue in KIO1. 2. KIO4 synthesises project architecture and required modules. 3. KIO7 generates code, tests, docstrings, and API documentation. 4. KIO6 validates requirement -> code alignment. 5. KIO12 performs security scans; KIO9 checks compliance. 6. User reviews generated artefacts and approved via HITL. 7. KIO1 commits files to the project repository.
Related KPIs	KPI 1.1 - Code Generation Speed KPI 3.1 - Code Quality Improvement KPI 5.1 - Time-to-Market KPI 8.1 - Adoption Rate
Related KIOs	KIO1, KIO3, KIO4, KIO6, KIO7, KIO9, KIO12
Related Requirements	FR-UC1-12, FR-UC1-13, FR-UC1-15, FR-UC1-16, FR-UC1-17, FR-UC1-21
Validation metrics	• Compilation success rate • Requirement-to-code traceability score • Test coverage of generated tests
Interdependencies	Prerequisite: UC-UC1-01 Successor: UC-UC1-03

Table 27: UC-UC1-03: Automated Bug Localisation, Fix Suggestion, and Verification

ID	**UC-UC1-03**
Name	Automated Bug Localisation, Fix Suggestion, and Verification
Description (short)	This scenario uses KIO2, KIO11, and KIO7 to diagnose failing tests or runtime anomalies. The platform identifies faulty code regions, proposes a fix, regenerates tests, and verifies correctness and energy impact before committing the final fix.
Actor(s)	Developers, testers, CI/CD pipelines
Trigger	A failing unit test, integration test, or runtime exception is detected.
Pre-condition(s)	• Test suite available. • Code generated or manually written • KIO2 debugging configuration available
Input	• Failing test results or runtime logs
Output	• Bug localisation trace • Proposed fix patch. • Updated verifying tests • Energy impact assessment • Fix verification report
Post-condition(s)	• Fix validated and committed. • Updated test suite aligned with patch. • Traceability maintained
Sequence	1. Failing test reported by CI or tester. 2. KIO11 performs symbolic/dynamic analysis to confirm failure. 3. KIO2 uses forward & reverse execution to localise faulty lines. 4. KIO7 proposes a fix patch and regenerates verifying tests. 5. KIO10 evaluates energy impact of alternative fixes. 6. KIO12 validates security and KIO9 checks compliance.

	7. Developer approves fix through HITL.
Related KPIs	KPI 6.1 - Bug-Fix Time KPI 6.2 - Customer Reported Issues KPI 9.2 - Technical Debt Reduction
Related KIOs	KIO1, KIO2, KIO7, KIO11, KIO12, KIO10
Related Requirements	FR-UC1-12, FR-UC1-21, FR-UC1-27
Validation metrics	• Time from failure detection to validated fix. • Accuracy of bug localisation • Improvement in test pass rate
Interdependencies	Prerequisite: UC-UC1-02 Successor: UC-UC1-04

Table 28: UC-UC1-04: Full End-to-End Feature Delivery with Compliance and Energy Check

ID	**UC-UC1-04**
Name	Full End-to-End Feature Delivery with Compliance and Energy Check
Description (short)	This scenario spans the entire horizontal value chain through customer request draft issue, requirements, code, tests, debugging, compliance, security, energy optimisation, deployment. The platform automates the whole SDLC while ensuring traceability, GDPR compliance, secure-by-design, and low-energy execution.
Actor(s)	Project managers, developers, QA engineers, DevOps engineers, compliance officers
Trigger	A feature request triggers the start of automated end-to-end delivery.
Pre-condition(s)	• UC-UC1-01, UC-UC1-02, UC-UC1-03 completed. • Deployment configuration available • CI/CD and energy profiling configurations active
Input	• Feature request, approved issue, generated code, validated tests
Output	• Deployed feature • Compliance & security reports • Energy-optimised build • Full traceability ledger
Post-condition(s)	• Feature live in staging or production • All artefacts versioned with traceability links
Sequence	1. UC-UC1-01 generates requirements and draft issue. 2. UC-UC1-02 generates code and tests. 3. UC-UC1-03 handles any emerging bugs. 4. KIO12 performs final vulnerability scans. 5. KIO9 validates GDPR and responsible-AI compliance. 6. KIO10 assesses energy usage and optimises model/logic if required. 7. KIO8 prepares the build for deployment. 8. KIO1 triggers deployment and stores the full audit trail.
Related KPIs	KPI 2.1 - Lifecycle Energy Reduction KPI 2.2 - Deployment Energy Efficiency KPI 5.1 - Time-to-Market KPI 8.1 - Adoption Rate
Related KIOs	KIO1, KIO3, KIO4, KIO6, KIO7, KIO8, KIO9, KIO10, KIO11, KIO12
Related Requirements	All FR-UC1 requirements
Validation metrics	• Total SDLC duration • Number of defects in final build • Energy consumption per build • Compliance and security pass rate

Interdependencies	Completes the full UC1 pipeline

5.3.2 UC2 - Automating Medical Report Generation Using Large Language Models (Vertical)

This vertical use-case targets healthcare documentation, automating Electronic Health Records (EHR)-based report creation whilst upholding HIPAA/GDPR via privacy-preserving techniques and HITL clinical oversight.

Table 29: UC-UC2-01: Medical Report Generation from Patient Notes and Medical Record Data

ID	UC-UC2-01
Name	Medical Report Generation from Patient Notes and Medical Record Data
Description (short)	A primary care physician accesses the healthcare portal during or after patient consultation and initiates clinical report generation. The system synthesises multi-source patient data, validates clinical accuracy, ensures GDPR, KVKK and HIPAA compliance, and produces a professional report suitable for clinical decision-making and documentation.
Actor(s)	Physician user completing patient consultation, desktop/tablet-based interface for physician workflow, electronic health records (EHR) with patient history.
Trigger	Physician completes a patient consultation and initiates clinical report generation to prepare comprehensive documentation for a case review meeting scheduled for the same day. Physician selects "Generate Clinical Report" action in the clinical portal after finalising patient visit notes.
Pre-condition(s)	• Physician has valid credentials and authorised access to clinical portal (physician role authentication) • Patient's complete medical record is available in EHR system with visit notes, imaging, lab results. • Physician has completed clinical assessment and documented patient visit notes in EHR. • System has access to relevant clinical literature and evidence-based resources. • HIPAA/GDPR/KVKK compliance protocols are active for clinical documentation

Input	• **Patient Visit Notes**: Physician's clinical documentation from consultation (chief complaint, present illness, physical examination findings, assessment, plan)
	• **Patient Identifier**: Unique patient ID linking to complete medical record.
	• **Patient Medical History (Accessed from the EHR database)**:
	  ○ Previous diagnoses and treatment outcomes
	  ○ Current active medications and allergies
	  ○ Vital signs and anthropometric data
	  ○ Laboratory and imaging results
	  ○ Surgical history and procedures
	  ○ Vaccination records

Output	• **Comprehensive Clinical Report**: Professional medical document including:
	  ○ Chief complaint and history of present illness summary
	  ○ Relevant medical history synthesis
	  ○ Physical examination findings
	  ○ Current medication list with interactions noted.
	  ○ Problem list with active diagnoses
	  ○ Clinical assessment and diagnostic reasoning
	  ○ Recommended treatment plan with evidence-based justification.

	○ References to clinical guidelines and literature • **Evidence-Based Recommendations:** AI-synthesized clinical suggestions based on patient-specific factors and guidelines. • **Physician Signature Block:** Area for physician digital signature or authentication • **Audit Trail:** Complete documentation of report generation process, data accessed, and modifications made. • **Exportable Report:** PDF or secure format suitable for case review meeting distribution
Post-condition(s)	• Clinical report is generated and available for physician review. • Physician reviews report makes any necessary edits or additions to finalise documentation. • Final report is signed/authenticated by physician and locked against further modifications (immutable version created) • Report is documented in patient's permanent medical record with timestamp and clinician attribution. • Audit trail shows all report access and clinical decision support references used. • Report meets all regulatory documentation standards for clinical medical records. • Patient's clinical care plan and records are updated in the EHR database based on approved reports.
Sequence	1. Physician logs into clinical portal with physician credentials, selects "Generate Clinical Report" option. 2. KIO1 receives report generation request with patient ID and visit context. 3. KIO1 routes to KIO3 (NLP module) to extract clinical entities from physician notes. KIO3 identifies diagnoses, medications, procedures, risk factors, and relevant clinical findings. Extracted data is displayed for verification and correction of any misinterpreted clinical information. 4. KIO1 activates KIO6 to verify all essential clinical data for report generation is available: patient history, current medications, lab results, imaging, vital signs. KIO6 identifies any missing or stale data and prompts physician to address critical gaps. 5. KIO1 launches KIO7 with all validated clinical data and context. KIO7 synthesises comprehensive clinical report incorporating: ○ Patient history narrative synthesis ○ Differential diagnosis analysis ○ Evidence-based treatment recommendations ○ Drug interaction and contraindication cross-checks ○ Clinical decision support references 6. KIO1 engages KIO9 to verify report meets clinical documentation standards and regulatory compliance: ○ Appropriate clinical terminology and terminology consistency ○ HIPAA/GDPR/KVKK compliance and proper patient identifiers ○ No inappropriate clinical recommendations detected. ○ Proper disclaimers and scope limitations included. 7. AI-generated report is shown to the physician for review and final edits as needed to ensure clinical accuracy and appropriate recommendations. Physician adds digital signature/authentication, finalising the immutable record. Report is submitted to the case review system and locked to prevent any modifications.
Related KPIs	KPI 1.1, KPI 1.2, KPI 3.1, KPI 4.1, KPI 6.1, KPI 6.2, KPI 8.2
Related KIOs	KIO1, KIO3, KIO4, KIO6, KIO7, KIO9
Related Requirements	FR-KIO1-04, FR-KIO3-02, FR-KIO6-02, FR-KIO7-03, FR-KIO9-05

Validation metrics	• Extracted clinical findings verified accurate by physician during review. • Medical data validation passed with all critical fields complete. • Compliance checks confirmed HIPAA/GDPR/KVKK standards met. • Physician successfully authenticates and signs report, creating immutable record. • No unauthorised data access detected in audit trail
Interdependencies	Prerequisite: All UC2 requirements

Table 30: UC-UC2-02: Generate Medication Safety Report from Natural Language Patient Query

ID	**UC-UC2-02**
Name	Generate Medication Safety Report from Natural Language Patient Query
Description (short)	A patient accesses the healthcare portal, queries medication interactions using natural language, and receives a personalised safety analysis. The system validates the patient's data completeness, performs AI-driven interaction analysis, ensures regulatory compliance, and delivers a patient-readable report.
Actor(s)	Primary user accessing the patient portal via web browser, web-based frontend for patient interaction, EHR database containing patient medical data, reference database for medication interactions
Trigger	Patient clicks on the chatbot interface in the patient portal and submits a natural language query regarding medication interactions or medical concerns (e.g., "I need to check if the ibuprofen 400mg I take for my back pain interacts with any of my current prescriptions").
Pre-condition(s)	• Patient has a valid account with credentials verified and securely logged into the portal. • Patient's complete medical records are available in the EHR system (prescriptions, allergies, medical conditions) • Drug interaction database is current and accessible. • HIPAA/GDPR/KVKK compliance protocols are active

Input	• **Natural Language Query**: Patient's textual request describing medication or health concern.
	• **Patient Identifier**: Unique patient ID
	• **Patient Medical Records (Accessed from the EHR Database)**:
	  ○ Active prescriptions list.
	  ○ Medication history (past 2 years minimum)
	  ○ Documented allergies
	  ○ Current medical conditions
	  ○ Recent laboratory results
	  ○ Previous adverse drug reactions
	• **Optional**: Patient confirmation of medication frequency/dosage if ambiguous

Output	• **Structured Query Requirements:** Formally extracted and validated patient request
	• **Data Validation Report:** Completeness assessment of medical records
	• **Medical Analysis Report:** AI-generated medication interaction analysis with:
	  ○ Identified interactions (contraindicated, major, moderate, minor)
	  ○ Risk severity levels.
	  ○ Patient-specific considerations
	  ○ Evidence-based recommendations
	• **Compliance Certification**: Validation that report meets HIPAA/GDPR/KVKK standards.

	• **Patient-Readable Summary**: Formatted report displayed in patient portal with clear warnings and actionable guidance. • **Audit Log**: Timestamped record of data access and processing steps
Post-condition(s)	• Patient receives personalised medication safety report displayed in patient portal. • Medical analysis is documented in patient's EHR with audit trail. • System maintains complete audit log of data access and compliance checks. • No patient data is retained in model training logs or unencrypted temporary storage. • All interactions logged per HIPAA/GDPR/KVKK requirements for potential regulatory review
Sequence	1. Patient logs into the patient portal with secured credentials; system verifies authentication status. 2. Patient clicks chatbot interface, enters natural language query about medication interactions, and submits request to KIO1. 3. KIO1 receives query and routes to KIO3 (NLP module). KIO3 extracts and formalises requirements, identifying specific medications, dosages, and analysis scope. System displays extracted requirements to patient for confirmation. 4. KIO1 activates KIO6 to verify all necessary patient medical data is complete and current. If data gaps exist, system prompts patient to provide missing information (e.g., medication frequency confirmation). 5. KIO1 launches KIO7 with validated patient data. KIO7 performs comprehensive interaction analysis using AI models, drug database queries, and patient-specific factors, generating detailed analysis report with severity levels and recommendations. 6. KIO1 engages KIO9 to verify report meets all regulatory standards (HIPAA, GDPR, KVKK), appropriate disclaimers are included, and no data leakage risks exist. KIO9 validates audit trail completeness. 7. KIO1 displays final patient-readable report in portal with clear formatting, warnings highlighted, and recommended actions summarised. Patient can review, share with physician, or seek additional guidance.
Related KPIs	KPI 1.1, KPI 1.2, KPI 3.1, KPI 6.2, KPI 8.1, KPI 8.2
Related KIOs	KIO1, KIO3, KIO6, KIO7, KIO9
Related Requirements	FR-KIO1-04, FR-KIO3-02, FR-KIO6-02, FR-KIO7-03, FR-KIO9-05
Validation metrics	The system successfully identifies and extracts the medication name from a standard patient query. The generated report includes warnings for known contra-indications that exist in the reference database. The system prompts the patient for clarification if the dosage or frequency is missing from the input. The displayed report contains only data relevant to the specific query and the logged-in user.
Interdependencies	Prerequisite: All UC2 requirements

Table 31: UC-UC2-03: Integrate and Harmonise Multiple Open-Source Healthcare Datasets into Unified Master Dataset

ID	UC-UC2-03
Name	Integrate and Harmonise Multiple Open-Source Healthcare Datasets into Unified Master Dataset
Description (short)	Ingests multiple heterogeneous open-source healthcare datasets (MIMIC-IV, eICU, MIMIC-CXR, BIMCV-COVID19, synthetic patient records), validates

	data completeness and quality, applies schema harmonisation and standardisation, and produces a unified master dataset with consistent data structures and comprehensive clinical coverage for use as a synthetic EHR database.
Actor(s)	Data engineer/pipeline architect, Open-Source Data Sources
Trigger	Healthcare organisation or research institution initiates data preparation project to create comprehensive, harmonised EHR dataset for AI/ML model training. Data engineering team receives project charter to consolidate multiple open-source healthcare datasets into single production-ready master dataset with synthetic data enhancement.
Pre-condition(s)	• Data engineering team has credentialed access to required repositories (e.g., PhysioNet for MIMIC-IV/eICU). • Target data schema (e.g., FHIR or OMOP Common Data Model) is defined and approved.
Input	**Raw Datasets:** • MIMIC-IV (Clinical data) • eICU (ICU data) • MIMIC-CXR (Chest X-ray metadata/images) • BIMCV-COVID19 (Imaging and records) • Synthetic patient records (supplementary data) **Reference Standards:** • ICD-10, LOINC dictionaries for standardisation.
Output	• A single accessible database harmonised records from all sources. • Documentation defining the structure, types, and sources of the unified data. • Log of excluded records, null values encountered, and transformation errors. • Traceability of the ingestion process.
Post-condition(s)	• The Unified Master Dataset is available for downstream AI training and application querying. • All original raw data is archived securely for audit purposes. • Patient identifiers across different datasets are either linked (if applicable) or distinct ID collisions are resolved. • The dataset enables SQL or API-based retrieval by other system components.
Sequence	1. Pipeline authenticates with source repositories and downloads raw data batches to the staging area. 2. System removes duplicate records and re-verifies that no direct PII from the open-source fields was accidentally retained. 3. Pipeline maps heterogeneous schemas (e.g., different column names for "Blood Pressure") into the single Master Standard (e.g., FHIR). 4. Terminologies are normalised (e.g., converting local drug names to RxNorm codes). 5. Cleaned subsets are merged into the Master Dataset; synthetic links are created to form complete "patient journeys" if combining disparate sources. 6. Dataset is flagged as "Ready" for consumption by UC2-01 and UC2-02.
Related KPIs	N/A
Related KIOs	• The final dataset structure matches the defined Master Data Schema without errors. • All clinical events (labs, vitals) are successfully linked to a valid existing Patient ID (no orphan records). • Duplicate records from overlapping data sources are identified and resolved. • Clinical codes (diagnoses/medications) are successfully mapped to the standard format (e.g., ICD-10) where applicable. • The downstream application service account can successfully query the

	resulting database.
Related Requirements	N/A
Validation metrics	FR-KIO5-03, NFR-KIO5-09
Interdependencies	KIO5

5.3.3 UC3 - Large Language Models for Advanced Software Generation for FPGAs in C++ and HLS (Horizontal)

This horizontal use-case accelerates FPGA design cycles by enabling natural-language specification to bitstream generation, with iterative feedback from synthesis tools.

Table 32: UC-UC3-01: Align the FPGA developer's development task with formal requirements generation of the system

ID	**UC-UC3-01**
Name	Align the FPGA developer's development task with formal requirements generation of the system.
Description (short)	This interaction begins when the user types detailed instruction, in the chat-based interface of KIO1, describing all aspects of the HLS project they want to build. Then, KIO3 generates a first draft of formal requirements and begin a multi-turn conversation between the user and KIO3 until the user is satisfied with the result
Actor(s)	FPGA developer (user) Natural language instructions to formal requirement component (KIO3)
Trigger	User submits their instructions through the chat-based interface of KIO1
Pre-condition(s)	The project is fully initialised by KIO1
Input	User prompt with details about the HLS implementation they want
Output	Document with all the formal requirements KIO3 detects from the initial user instructions and subsequent multi-turn conversation. This document is shown to the user through the chat-based interface
Post-condition(s)	The system has verified that the formal requirements cover High-Level Synthesis (HLS) codebase creation, code generation, code validation and project compilation, among other more specific details, all of which are framed within the user's natural-language specifications. The user sees in the interface all the formal requirements generated based on their specifications.
Sequence	1. The user submits instructions describing HLS project details. 2. KIO3 converts user input into formal requirements. 3. KIO3 starts a multi-turn conversation with the user to gather more details about the user's specifications. 4. KIO1 displays the document with the formal requirements detected from previous interaction with the user.
Related KPIs	• KPI1.1 Measure how much time KIO3 needs to generate all the documents such as user stories, formal requirements, etc., using KIO1 time stamps. • KPI7.1 - Annual cost saving - ~15% of baseline annual cost reduction in requirements elicitation stage in a software development project.
Related KIOs	KIO1, KIO3
Related Requirements	All functional and non-functional requirements of KIO3, FR-KIO1-07
Validation metrics	All artifacts generated are visible through GUI; feedback included in KIO3 response
Interdependencies	**Optional successor:** UC-UC3-02

Table 33: UC-UC3-02: Generate HLS project

ID	UC-UC3-02
Name	Generate HLS project
Description (short)	In this scenario the user triggers the HLS codebase generation based on already well-defined requirements deduced from previous interactions between the user and KIO3. KIO4 is activated and builds the base code structure, after which KIO7 and KIO2 are triggered to perform the code generation and refinement loop. Eventually, the generated project is compiled by KIO8. Depending on the compilation results, the execution either finishes or restarts the workflow. The final outcome is a compiled HSL project
Actor(s)	FPGA developer (user)
Trigger	User reviews formal requirements generated in previous interaction with the system and presses a button such as “Continue building project” to trigger KIO4
Pre-condition(s)	Formal requirements have been generated by KIO3, and they are displayed by KIO1 in its chat-based interface
Input	• User presses a confirmation button to continue with the workflow. • Artifacts generated in KIO3
Output	• Compiled HLS project. • HLS project codebase • Reports
Post-condition(s)	• The system displays a final summary containing technical information about the HLS implementation, along with the generated HLS project and reports. • This is the final stage of the system execution
Sequence	1. The user presses a confirmation button. 2. KIO4 creates the codebase structure. 3. KIO7 makes an execution planning and a strategy to follow, then it starts generating code iteratively. 4. KIO2 validates and refines code generated by KIO7. 5. KIO8 compiles the generated HLS project, executing the toolchain including HDL code generation and -simulation, and sends results to KIO2 which verifies whether performance metrics are reached or need refinement. 6. KIO1 displays a final report with implementation details, metrics and design decisions justifications, as well as the HLS project codebase
Related KPIs	KPIs 1.1, 8.3
Related KIOs	KIO1, KIO2, KIO4, KIO7, KIO8
Related Requirements	KIO1: All functional and non-functional requirements KIO2: FR-KIO2-01, FR-KIO2-02, FR-KIO2-03, FR-KIO2-04, FR-KIO2-06, FR-KIO2-07. All KIO2 non-functional requirements KIO4: All KIO4 requirements EXCEPT NFR-KIO4-11, NFR-KIO4-05, FR-KIO4-08 KIO7: FR-KIO7-03, FR-KIO7- 04, FR-KIO7- 05, FR-KIO7- 06, FR-KIO7-07, FR-KIO7-10, NFR-KIO7-07, NFR-KIO7- 08, NFR-KIO7-11, NFR-KIO7-12 KIO8: All functional and non-functional requirement
Validation metrics	The compilation is successful, and the generated program adheres to the requirements and feedback provided by the user.
Interdependencies	**Prerequisite:** UC-UC3-01 This is the final scenario.

Table 34: UC-UC3-03: Human-in-the-loop process to assess and guide code generation

ID	UC-UC3-03
Name	Improve code quality through human-in-the-loop process to assess and guide code generation
Description (short)	During the execution of processes involving KIO7 for • HLS code generation, • C++ simulation of the HLS code, • HLS synthesis for HDL code generation, and • Co-simulation of the HDL code, it must be possible for the user to provide feedback, guidelines or even code to steer the code generation. The user can see what is being generated and based on that, decide whether to intervene
Actor(s)	FPGA developer (user) Chat-based interface (KIO1)
Trigger	KIO1 either prompts the user for feedback or enables the user to interrupt execution to intervene.
Pre-condition(s)	• The codebase structure has been created. • KIO7 has generated (partial) HLS code OR KIO8 is executing the toolchain. • KIO1 chat-based interface is available because there is no code generation in progress, and is reporting information on code generation and simulations
Input	• Codebase, in case the user modifies files or directories. • Generated HLS code so far, in case the user modifies code. • User feedback prompt
Output	• New HLS code (in practice, HDL code will not be altered directly, but the HDL can trigger changing the HLS) • Modifications to existing files • Logs are created of the user intervention
Post-condition(s)	The state is mostly the same as before in this use-case, what has changed is that the user is prompted to steer the results of previous code generations
Sequence	• The chat-based interface (KIO1) is ready to receive user feedback. • The user provides instructions to guide the generation. • KIO1 propagates this input to downstream KIOs
Related KPIs	KPIs 3.1, 3.2
Related KIOs	KIO1, KIO7, KIO8
Related Requirements	KIO1: All functional and non-functional requirements KIO7: NFR-KIO7-03 KIO8: FR-KIO8-03, FR-KIO8-04
Validation metrics	KIO1 confirms that it has successfully received the user instructions (binary). Subsequent code generations adhere to these instructions.
Interdependencies	UC-UC3-02

5.3.4 UC4 - Next-Generation Battery Management in EVs with AI-Driven TinyML Solutions (Vertical)

This vertical use-case focuses on privacy-preserving RUL prediction at the edge, integrating Graph Neural Networks (GNNs), LLMs, and federated learning for real-time, low-power EV battery management.

Table 35: UC-UC4-01: Perform Exploratory Data Analysis with Early Energy Constraints and HITL Oversight

ID	UC-UC4-01
Name	Perform Exploratory Data Analysis with Early Energy Constraints and HITL Oversight
Description (short)	This scenario begins the UC4 pipeline by exploring and validating uploaded multi-sensor EV battery datasets. Through the KIO1 chat-based GUI and KIO3 requirement generator, engineers trigger containerised KIO6 for cleaning, validation, and exploratory analysis. Human In the Loop (HITL) supervision ensures that feature selection, data balancing, and privacy preservation meet TinyML deployment requirements.
Actor(s)	EV software engineers, TinyML developers, data privacy specialists
Trigger	The user issues a natural-language prompt in the KIO1 GUI, such as "Initiate TinyML-constrained EDA for TinyML/ EV Battery Data set".
Pre-condition(s)	UC4 project initialised in KIO1; requirements parsed using KIO3 and energy constraints embedded using KIO10; raw datasets (sensor logs, cycle data, maintenance reports) uploaded via GUI.
Input	**Mandatory:** Raw battery sensor datasets, operational logs, maintenance reports. **Optional:** User prompts specifying feature or data-volume limits.
Output	Versioned, validated Parquet dataset; synthetic data set (if privacy enabled using KIO5); exploratory analysis notebook; quality and anomaly report; streamed visualisations and HITL decision log.
Post-condition(s)	Dataset tagged as "EDA-TinyML-ready"; quality and privacy checks passed (using KIO9); artefacts available for subsequent architecture (KIO4) and code generation (KIO7).
Sequence	1. The engineer uploads datasets through KIO1 GUI, requirements parsed using KIO3 and initiating orchestration of KIO6 for schema validation and integrity checks. 2. KIO6 analyses missing values, noise, and outliers, flagging quality thresholds in the chat stream for HITL acceptance. 3. KIO5 may be invoked to create synthetic samples ensuring privacy and data balance under constraints. 4. HITL approval is provided for each cleansing and synthesis step through GUI prompts. 5. KIO6 conducts exploratory analysis: correlation matrices, feature distributions, and early anomaly detection using statistical and GNN, RNN/LSTM-based diagnostics. 6. Summary visualisations and insights are streamed for HITL confirmation, ensuring feature selection aligns with TinyML footprint limits. 7. Upon sign-off, KIO1 versions the dataset and attaches all validation logs (input for KIO9 for documentation).
Related KPIs	KPI1.2
Related KIOs	KIO1, KIO3, KIO4, KIO5, KIO6, KIO9 and KIO10
Related Requirements	FR-KIO1-04, all the requirements of KIO10
Validation metrics	Data integrity score; anomaly detection accuracy; dataset size and balance ratio; HITL approval timestamps; privacy audit completeness.
Interdependencies	**Prerequisite:** UC4 project initialisation and constraint definition. **Successor:** UC-UC4-02 (uses validated dataset for model and code generation).

Table 36: UC-UC4-02: Generate Energy-Constrained Federated TinyML Codebase with HITL Refinement

ID	UC-UC4-02
Name	Generate Energy-Constrained Federated TinyML Codebase with HITL Refinement
Description (short)	This scenario synthesises the validated EDA artefacts into an energy-efficient, federated TinyML codebase. The KIO1 GUI displays the generated architecture for LLM-driven coding using KIO7, and cross-compilation (using KIO8) under the constraints defined by KIO10 Platform Capability Model (PCM). HITL checkpoints ensure model feasibility and code compliance before deployment.
Actor(s)	EV software engineers, TinyML developers, data privacy specialists
Trigger	A prompt such as "Generate federated TinyML RUL codebase" is entered into the KIO1 GUI after EDA completion.
Pre-condition(s)	UC-UC4-01 completed and dataset tagged "EDA-TinyML-ready"; KIO10 Platform Capability Model (PCM) available; microcontroller profile configured.
Input	**Mandatory:** Validated dataset ID, PCM constraints. **Optional:** Refinement prompts (e.g., "Apply GRU for sequence prediction" or "Use GNN for sensor fusion").
Output	Git repository with TFLite Micro model; Flower-based federated learning scripts; quantised model graph; KIO2 reverse/forward debugging hooks; build scripts (using KIO7 and KIO8); HITL refinement log.
Post-condition(s)	Repository tagged "Code-TinyML-optimised"; energy and compliance checks passed; deployment unlocked.
Sequence	1. The user requests model generation in the KIO1 GUI. Orchestration loads KIO3 formal requirements and KIO4 architecture options filtered through KIO10 PCM (DSM) limits/ constraints. 2. Architectures are streamed to the GUI-LSTM for sequence prediction, GNN for sensor fusion, CNN for signal analysis-and selected via HITL approval. 3. KIO7 generates federated TinyML code using LLM fine-tuning and retrieval-augmented reasoning, ensuring compatibility with TensorFlow Lite Micro using KIO6 data input. 4. KIO9 validates ethical, privacy, and design-by-contract compliance. Violations or optimisations stream back for HITL sign-off. 5. KIO11 executes automated reliability and fuzz testing; KIO2 previews forward/reverse execution traces for debugging, enabling developers to approve fixes conversationally. 6. KIO12 performs vulnerability and anomaly detection. 7. KIO9 validates ethical, privacy, and design-by-contract compliance. Violations or optimisations stream back for HITL sign-off. 8. KIO8 performs cross-compilation and device-specific optimisation, in a feedback loop with KIO10 for constraint compliance (KIO10 ⇄ KIO8 ⇄ KIO7). 9. Upon HITL confirmation, KIO1 commits the repository, tags artefacts, and records execution logs.
Related KPIs	KPI1.1, KPI2.1, KPI3.1, 3.2, 3.3, KPI6.1, 6.2
Related KIOs	All the KIOs (except KIO5)
Related Requirements	All KIO12, KIO7, KIO2, KIO6 requirements
Validation metrics	Code compilation success; memory footprint and latency estimation; PCM compliance report; privacy and ethics validation; successful HITL review entries.

Interdependencies	**Prerequisite:** UC-UC4-01. **Successor:** UC-UC4-03 (deployment and optimisation).

Table 37: UC-UC4-03: Deploy and Runtime-Optimise Federated TinyML Firmware with HITL Monitoring

ID	UC-UC4-03
Name	Deploy and Runtime-Optimise Federated TinyML Firmware with HITL Monitoring
Description (short)	This scenario deploys the generated TinyML firmware to edge devices and initiates runtime energy optimisation. KIO10 operates in its deployment mode, profiling inference, quantising models, and dynamically adapting them under live conditions. HITL monitoring through the KIO1 GUI governs each optimisation cycle to ensure efficiency and reliability in real-time RUL prediction.
Actor(s)	EV software engineers, TinyML developers, data privacy specialists
Trigger	A natural-language prompt such as "Deploy UC4 firmware to edge" is entered after code generation.
Pre-condition(s)	UC-UC4-02 completed; repository tagged and validated; device or simulator connected via GUI.
Input	**Mandatory:** Repository ID and connection profile. **Optional:** Desired optimisation level or telemetry duration.
Output	Flashed binary; profiling and energy-consumption report; federated learning logs; telemetry dashboard; refined model checkpoint; HITL log.
Post-condition(s)	Devices executing live RUL prediction; telemetry linked; optimisation loop active and tagged "Deployed-and-Monitoring".
Sequence	1. The user initiates deployment via KIO1 GUI, selecting edge targets. KIO8 prepares binaries for the defined MCU. 2. Firmware flashing proceeds, after which KIO10 profiles inference latency and energy consumption. 3. Detected threshold breaches trigger automatic proposals from KIO10 for pruning or quantisation; HITL confirmation required before redeployment. 4. Federated rounds begin, using Flower integration for privacy-preserving model updates. 5. KIO11 executes automated reliability and fuzz testing; KIO12 performs vulnerability and anomaly detection. 6. Any issues stream into KIO2 for reverse execution and trace-based correction; fixes require HITL approval. 7. Final compliance and energy metrics are verified, after which KIO1 activates live monitoring and archives artefacts.
Related KPIs	KPI2.2, KPI3.3
Related KIOs	KIO1, KIO2, KIO7, KIO8, KIO10, KIO11, KIO12
Related Requirements	KIO10, KIO8, KIO12 and KIO9 requirements
Validation metrics	Runtime profiling logs; energy consumption measurements; convergence and latency reports; anomaly and vulnerability counts; HITL verification records.
Interdependencies	**Prerequisite:** UC-UC4-02. **Completes:** UC4 pipeline.

5.3.5 UC5 - AI-Powered GIS Integration for Smart Energy Management (Vertical)

This vertical use-case optimises energy grids using GIS data fusion, predictive modelling, and automated testing for renewable integration.

Table 38: UC-UC5-01: Generate User Story from Contextual Requirements

ID	UC-UC5-01
Name	Generate User Story from Contextual Requirements
Description (short)	This scenario begins the UC5 pipeline. Through the KIO1 VS Code chat-based extension, engineers trigger the KIO3 requirement generator. The User Story should clearly indicate its related tasks and acceptance criteria.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Trigger	The user issues a natural-language prompt in the KIO1 GUI, such as "Deployment scenarios of EVs and of required infrastructure will be developed by merging data on the existing car fleet, electrification scenarios for the respective countries, mobility surveys and considering PV potential and its development as well as the availability of flexible assets".
Pre-condition(s)	• UC5 project initialised in KIO1 • High-level contextual development constraints provided by the Product Manager • The UC5 project exists on GitHub
Input	User prompts specifying a feature with contextual requirements
Output	• User Story in format "As a [type of user], I want [some goal] so that [some reason]." • Implementation steps are defined as a Task list. • Acceptance criteria in format ("Test that [description of test performed] ...")
Post-condition(s)	• User Story, including task list and acceptance criteria (Product Backlog Items), is saved on GitHub. • Product Backlog Items are available for geospatial datasets generation, code generation and test generation
Sequence	1. The energy analyst enters a detailed description of the desired requirement using a natural-language prompt in the KIO1 GUI. 2. The user requests User Story generation. 3. Requirement is parsed using KIO3. 4. KIO3 generates a User Story and clearly indicates development steps as a Tasks list. 5. KIO3 generates Acceptance Criteria in the correct format. 6. Human-In-The-Loop approval is provided when generated Product Backlog Items are aligned with the user's instructions. 7. Product Backlog Items are made available to the geospatial dataset generation module, the code generation module, and the tests generation module.
Related KPIs	KPI3.2, KPI4.1, KPI5.1, KPI7.1
Related KIOs	KIO1, KIO3
Related Requirements	FR-UC5-02, FR-UC5-03, FR-UC5-04, FR-UC5-05, FR-UC5-06, FR-UC5-08, FR-UC5-09, NFR-UC5-01
Validation metrics	• Generation time: the time elapsed between the user's prompt and the user stories generated. • User story quality: a qualitative assessment of the completeness and correctness of the generated user stories.
Interdependencies	Successor: UC-UC5-02

Table 39: UC-UC5-02: Generate and validate dataset

ID	UC-UC5-02
Name	Generate and validate geospatial datasets
Description (short)	The suite identifies necessary geospatial datasets (e.g., weather observations and forecasts, elevation models, infrastructure layers, population density, etc.) and leverages the Data Lakehouse (KIO5) to produce a coherent and validated set of data.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Trigger	A prompt such as "Generate necessary Dataset" is entered into the KIO1 GUI after User Story generation.
Pre-condition(s)	• Data available in the data Lakehouse. • Valid credentials for data access.
Input	High-level instructions and requirements from users, possibly refined through a conversational interface.
Output	Geospatial layers consistent with the user's instructions.
Post-condition(s)	Data are available for code generation, testing, and deployment.
Sequence	1. LLM is prompted with user instructions and a description of available tools. 2. LLM generates a query to fetch relevant layers and filtering criteria. 3. Query is passed to the data Lakehouse, which generates and validates the data. 4. Generated data are made available to the code generation module.
Related KPIs	KPI3.1, KPI4.1
Related KIOs	KIO1, KIO5, KIO6
Related Requirements	FR-UC5-07, NFR-UC5-01
Validation metrics	• Generation time, between input and data being available for downstream components. • Assessment of failure modes: identification of boundary conditions beyond which the system is unable to comply with the user's instructions, either due to the complexity of the scenario or the limitations of the model. • Success rate on a set of queries.
Interdependencies	**Prerequisite:** UC-UC5-01 **Successor:** UC-UC5-03

Table 40: UC-UC5-03: Generate Calculation Module codebase

ID	UC-UC5-03
Name	Generate Calculation Module codebase
Description (short)	This scenario synthesises the validated User Story into a Calculation Module codebase. The KIO1 GUI orchestrates architecture generation and LLM-driven coding using KIO7. Human-in-the-Loop checkpoints ensure algorithm logic and code compliance before testing.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Trigger	A prompt such as "Generate calculation module codebase" is entered into the KIO1 GUI after Geospatial Dataset generation and validation.
Pre-condition(s)	• An optimal GIS and framework have been defined. • Generated User Story with development Tasks list is available. • Necessary geospatial Datasets are available. • Base Calculation Module source code is available on GitHub
Input	• Generated User Story • Base Calculation Module source code and file structure

	• Geospatial Datasets
Output	New Calculation Module codebase
Post-condition(s)	• Code is saved on GitHub. • Generated code is available for testing
Sequence	1. The user requests code Calculation Module code generation in the KIO1 GUI. 2. KIO1 activates KIO4 components to check whether the required Geospatial Datasets are available and accessible. 3. Verification of the Development environment setup with the necessary libraries is performed. 4. KIO7 generates the Calculation Module code using optimal GIS and framework selection. 5. Code generation follows development tasks defined in the generated User Story 6. Human-in-the-Loop checks and inputs ensure the generated code meets the user's algorithm implementation logic. 7. Calculation Module code is made available to the Test generation module
Related KPIs	KPI4.1, KPI8.1, KPI9.1
Related KIOs	KIO1, KIO4, KIO7
Related Requirements	FR-UC5-01, FR-UC5-10, FR-UC5-11, FR-UC5-12, NFR-UC5-01
Validation metrics	• Generation time, between input and generated code • Assessment of failure modes: identification of (qualitative) criteria causing the model to generate incorrect solutions. • Success rate on a curated set of relevant inputs.
Interdependencies	**Prerequisite:** UC-UC5-02 **Successor:** UC-UC5-04

Table 41: UC-UC5-04: Generate Unit tests and Functional tests

ID	**UC-UC5-04**
Name	Generate Unit tests and Functional tests
Description (short)	This scenario generates Unit tests and Functional tests (i.e., Playwright)
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Trigger	A prompt such as "Generate unit tests and functional tests with Playwright" is entered into the KIO1 GUI after the Calculation Module code generation.
Pre-condition(s)	• Generated User Story with development Task list and Acceptance criteria is available. • Necessary geospatial Datasets are available. • Generated Calculation Module code is available
Input	• Generated User Story • Geospatial Datasets • Generated Calculation Module code
Output	• Unit tests • Functional tests in Playwright format • CI/CD pipeline as GitHub Action
Post-condition(s)	• Generated Unit tests and Functional tests are available for testing execution. • The generated CI/CD pipeline, as GitHub Action, is available for testing execution
Sequence	1. The user requests Unit tests and Functional tests generation in the KIO1 GUI 2. KIO1 launches KIO11.

	3. Generated Calculation Module code is analysed, and Unit tests are generated. 4. Acceptance Criteria defined in the generated User Story are analysed, and Functional tests in Playwright format are generated. 5. CI/CD pipeline is generated on GitHub and includes generated Unit tests and generated Functional tests. 6. Unit tests and Functional tests are made available to the Tests execution module. 7. CI/CD pipeline as GitHub Action is made available to the Tests execution module.
Related KPIs	KPI3.1, KPI4.1, KPI6.2, KPI9.1, KPI9.2
Related KIOs	KIO1, KIO11
Related Requirements	FR-UC5-13, NFR-UC5-01
Validation metrics	• Generation time, between input and the production of both unit and functional tests. • Coverage: measurement of the test coverage produced by the system. • Relevance: qualitative assessment of the quality of the tests, in terms of their ability to catch relevant bugs and corner cases.
Interdependencies	**Prerequisite:** UC-UC5-03 **Successor:** UC-UC5-05

Table 42: UC-UC5-05: Execute pipeline and run functional tests

ID	**UC-UC5-05**
Name	Execute pipeline and run functional tests
Description (short)	This scenario executes a CI/CD pipeline with GitHub Actions. Should the test-case fail, KIO1 runs KIO2 to locate bugs and fix the error in the generated code. Test reports are generated for review.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Trigger	A prompt such as "Execute CI/CD" is entered into the KIO1 GUI after Unit tests and functional tests generation.
Pre-condition(s)	• Generated unit tests and generated functional tests are available. • Generated CI/CD pipeline as a GitHub Action available. • Generated Datasets available
Input	• Generated unit tests. • Generated functional tests in Playwright format. • Generated CI/CD pipeline as GitHub Action • Generated Datasets
Output	Tests reports
Post-condition(s)	Test reports are saved on GitHub
Sequence	1. The user requests Unit tests and Functional tests execution in the KIO1 GUI 2. KIO1 launches KIO11. 3. The CI/CD pipeline, as a GitHub Action, is executed. 4. Should the test-case fail KIO1 runs KIO2 to locate and fix error(s) in the generated code. 5. Test reports are generated to list all passed/failed tests
Related KPIs	KPI3.1, KPI4.1, KPI6.2, KPI9.1, KPI9.2
Related KIOs	KIO1, KIO2, KIO11
Related Requirements	FR-UC5-14, FR-UC5-15, NFR-UC5-01
Validation metrics	• Execution time: the time for the test suite to run.

	• Test report quality: the ease of use of the test report, both for the user and KIO2, responsible for bug locating & fixing.
Interdependencies	**Prerequisite:** UC-UC5-04 **Completes:** UC5 pipeline

5.4 Cross-Use-Case Observations

This section provides a structured analysis of the patterns, synergies, divergences, and insights derived from the five use-cases (UC1–UC5). The analysis is based on their classification (horizontal or vertical), lead Partners, primary domains, key challenges, and verified Key Innovation Output (KIO) mappings. It illustrates how these use-cases collectively advance the AI4SWEng project objectives, including the provision of AI-powered tools for efficient, resilient, secure, and energy-efficient software engineering. The observations incorporate elements of the DevOps methodology, responsible AI principles, and proof-of-concept validation, with an emphasis on modularity, iteration, and regulatory compliance.

The use-cases are conceptualised as interconnected scenarios within the project ecosystem, facilitating the examination of interdependencies such as KIO chaining and human-in-the-loop (HITL) interventions. These elements promote a balance between AI autonomy and human oversight. The observations are organised into five categories for clarity.

5.4.1 Patterns in KIO Utilisation and Flows

The KIO mappings reveal consistent structural patterns:

- **Central Role of KIO1:** This component orchestrates all flows (100%), facilitating seamless DevOps integration through continuous integration/continuous delivery (CI/CD) and infrastructure as code (IaC), while supporting modular and scalable deployment.
- **Prevalent KIOs:** KIO7 (AI-driven code generation using large language models and retrieval-augmented generation) and KIO9 (responsible innovation and design-by-contract compliance) are ubiquitous, ensuring generation tasks and ethical oversight in every use-case. KIO10 (energy efficiency via model-driven TinyML) is applied proactively only in UC4, with implicit applications elsewhere.
- **HITL Interventions:** An average of 3–5 HITL gates per pipeline provides oversight at critical junctures, such as draft reviews in UC2 or code refinements in UC4, thereby enhancing trustworthiness and mitigating bias risks.
- **Cyclical and Modular Flows:** Flows exhibit iterative cycles (e.g., UC1 comprehensive SDLC loop from KIO1 to KIO1; UC3 targeted refinement), involving an average of 8–10 KIOs per use-case to foster synergistic complementarity.

5.4.2 Thematic Synergies and Divergences

Synergies and divergences emerge from the thematic alignment of the use-cases:

- **Synergies:**
  - **AI-Driven Automation:** All use-cases mitigate manual effort, with examples including code and test generation in UC1 and UC3, report automation in UC2, and predictive modelling in UC4 and UC5.
  - **Data Management:** KIO5 and KIO6 support privacy-preserving data synthesis and validation, evident in UC4 federated learning and UC5 GIS data fusion.
  - **Energy and Sustainability:** Energy efficiency is implicit across all (e.g., optimised practices in UC1), but explicit in UC4 and UC5 through TinyML and resource allocation.
- **Divergences:**
  - **Horizontal Use-cases (UC1 and UC3):** These emphasise iteration and generalisability, such as SDLC optimisation in UC1 and FPGA scalability in UC3, promoting reusability across multi-architecture systems.
  - **Vertical Use-cases (UC2, UC4, and UC5):** These prioritise domain-specific compliance and real-time adaptation, including HIPAA/GDPR adherence in UC2 and privacy in UC4.

5.4.3 Patterns in Challenge Mitigation

Cross-use-case challenges are addressed through systematic patterns:

- **Common Challenges:** Effort-intensive requirements are alleviated by KIO3 natural language processing; compliance and privacy issues by KIO9 and KIO12; and resource optimisation by KIO8 and KIO10.
- **Domain-Specific Challenges:** Vertical use-cases manage multi-sensor data and anomalies via KIO2 self-adaptive mechanisms, while horizontal ones address iterative synthesis through LLM fine-tuning.
- **Mitigation Strategies:** DevOps tools enable monitoring and automated testing via KIO11, with inter-scenario chaining resolving integration gaps in multi-architecture environments. Infrastructure as code ensures reproducibility and scalability.

5.4.4 Alignment with Innovation and Project Objectives

The use-cases constitute a cohesive solution stack, with KIO synergies validating core components such as the AI-SysDev platform and Testing Automation Tool (TAT). The complementarity between horizontal (foundational tools) and vertical (specialised

applications) use-cases ensures broad applicability. KIO13 (developer training) is implicitly supported through prompt engineering in LLM interactions, addressing the identified skills shortage.

5.4.5 Implications for Validation and Future Directions

Inter-scenario chaining provides comprehensive pipeline coverage, encompassing eighteen scenarios (excluding sub-variations from alternative flows). This facilitates robust validation in Work Package 5, including pilot implementations, performance benchmarking (e.g., stress testing), and certification for efficiency and reliability. Proof-of-concept demonstrations in real-world contexts (e.g., EV battery management in UC4, GIS systems in UC5) will substantiate transformative impacts, such as reduced errors and enhanced productivity.

6 Test-Case Specifications

The test-case specifications chapter presents a comprehensive and systematic validation strategy for the AI4SWEng project. All test-cases are derived directly from the use-case scenarios defined in the previous section, maintaining explicit traceability to the originating scenarios, functional and non-functional requirements, Key Innovation Outputs (KIOs), and measurable Key Performance Indicators (KPIs). This approach guarantees full coverage of the specified behaviours while enabling objective demonstration of conformance to project objectives, including energy efficiency, regulatory compliance, responsible AI principles, and DevOps practices.

The test-cases are organised by use-case to preserve contextual coherence and facilitate targeted execution during Work Package 5 pilots. Vertical use-cases (UC2, UC4, UC5), which operate in regulated or resource-constrained domains, receive the deepest coverage, incorporating positive and negative paths, boundary conditions, exception handling, and dedicated compliance verification. Horizontal use-cases (UC1, UC3) emphasise orchestration robustness, iterative refinement loops, and cross-architecture scalability.

6.1 Test-cases for individual Use-cases

6.1.1 UC1 - Optimising Software Development Lifecycle Using Large Language Models (Horizontal)

This horizontal use-case exercises the foundational end-to-end automation capabilities of the AI Engineering Suite in unconstrained software development contexts. Particular emphasis is placed on the accuracy and traceability of requirements elicitation from natural language, the effectiveness of multi-turn Human-in-the-Loop refinement, and the generation of high-coverage test suites with automated self-healing. These test-cases validate the platform ability to substantially reduce manual effort while maintaining artefact quality and DevOps pipeline integration.

Table 43: TC-UC1-01: Natural-Language Customer Request to Draft Issue and Requirements

ID	TC-UC1-01-01
Name	Natural-Language Customer Request to Draft Issue and Requirements
Description (short)	This test-case verifies that KIO3 correctly transforms a natural-language customer request into a structured draft issue containing functional and non-functional requirements, acceptance criteria, metadata (priority, story points, tags), and initial test ideas. HITL refinement is performed through the KIO1 GUI.
Actor(s)	Project managers, product owners, software architects
Preconditions	• UC1 project initialised in KIO1 • Requirement schema, templates, and taxonomies configured. • Historical project context ingested

Test Steps	1. Submit a natural-language customer request through the KIO1 GUI (e.g., "Customers should be able to reset their password securely"). 2. Observe KIO3's extraction of requirement candidates and classification into functional and non-functional items. 3. Inspect generated acceptance criteria and metadata (priority, tags, effort estimation). 4. Review the automatically created draft issue in KIO1. 5. Provide a HITL refinement (e.g., edit acceptance criteria). 6. Approve the final requirements and draft issue.
Expected Results	• Requirements generated without parser or schema failures. • Acceptance criteria automatically created and complete • Priority, tags, and story points assigned following guidelines. • HITL edits correctly integrated. • Draft issue stored as a versioned artefact
Postconditions	A complete, validated requirement & issue draft is available for UC-UC1-02.
Related KPIs	• KPI 1.2 - Issue Resolution Speed - Measured via time from natural-language input to approved draft issue; target ≤70% of baseline manual drafting time. • KPI 3.2 - Review Score Increase - Measured using HITL quality ratings of generated requirements; target ≥120% of baseline clarity/completeness. • KPI 5.1 - Time-to-Market - Measured as reduction in early SDLC phase duration; target ≥15% faster. • KPI 8.1 - Adoption Rate - Measured via percentage of requirement-drafting sessions using KIO3; target ≥80%. • KPI 8.2 - Active Usage & Satisfaction - Measured via KIO1 surveys and usage logs; target ≥4/5 satisfaction.
Related KIOs	KIO1, KIO3
Related Requirements	FR-UC1-04, FR-UC1-05, FR-UC1-06, FR-UC1-09, FR-UC1-10, FR-UC1-27
Validation Metrics	• Completeness ratio of requirement fields • Number of HITL refinements • Time from request submission to final draft issue
Interdependencies	Prerequisite: UC-UC1-01 Successor: TC-UC1-02-01

Table 44: TC-UC1-02: Automated Code and Test Generation from Approved Issue

ID	**TC-UC1-02-01**
Name	Automated Code and Test Generation from Approved Issue
Description (short)	This test-case evaluates the system's ability to generate code, unit tests, integration tests, and documentation from an approved issue. It also validates requirement-code consistency, compliance checks, and security scanning through connected KIO modules.
Actor(s)	Software developers, technical leads, QA engineers
Preconditions	• Approved draft issue from TC-UC1-01-01 • Architecture templates configured (KIO4) • Code generation capabilities active (KIO7) • Compliance and security rules configured (KIO9, KIO12)
Test Steps	1. Select an approved issue in the KIO1 GUI and request code generation. 2. KIO4 prepares architectural scaffolding and required module structures.

	3. Observe KIO7 generating code, unit tests, and integration tests. 4. Inspect documentation and code comments. 5. Verify requirement -> code -> test traceability in KIO6. 6. Execute KIO9 compliance checks and KIO12 security scans. 7. Approve artefacts via HITL workflow.
Expected Results	• Code compiles without errors. • Unit and integration tests are syntactically and logically valid. • Traceability links correctly established. • Compliance and security checks pass • HITL approval stored in project history
Postconditions	Generated code and tests are ready for debugging and execution in UC-UC1-03
Related KPIs	• KPI 1.1 - Code Generation Speed- Time from issue selection → generated code; target ≥30% faster than baseline. • KPI 3.1 - Code Quality Improvement - Measured through static analysis; target ≥30% improvement. • KPI 5.1 - Time-to-Market - Reduction in coding and testing phase duration; target ≥15%. • KPI 8.1 - Platform Adoption Rate - ≥75% automated generation usage vs manual coding.
Related KIOs	KIO1, KIO4, KIO6, KIO7, KIO9, KIO12
Related Requirements	FR-UC1-12, FR-UC1-13, FR-UC1-15, FR-UC1-16, FR-UC1-17, FR-UC1-21
Validation Metrics	• Compilation success rate • Test coverage and correctness. • Requirement-to-code consistency score
Interdependencies	Prerequisite: UC-UC1-01 Successor: TC-UC1-03-01

Table 45: TC-UC1-03: Automated Bug Localisation, Fix Suggestion, and Verification

ID	TC-UC1-03-01
Name	Automated Bug Localisation, Fix Suggestion, and Verification
Description (short)	This test-case verifies the full debugging loop: detection of a failing test, automatic bug localisation via KIO2, generation of a fix patch via KIO7, validation through test regeneration, and assessment of energy and compliance impacts.
Actor(s)	Developers, testers, CI/CD systems
Preconditions	• Generated code and tests (TC-UC1-02-01 passed) • KIO2 debugging mode enabled. • Energy impact rules configured (KIO10)
Test Steps	1. Trigger a failing unit or integration test in CI. 2. KIO11 confirms the failure and extracts diagnostic data. 3. KIO2 performs forward and reverse execution to identify root cause lines. 4. KIO7 proposes a fix patch and regenerates test-cases validating the fix. 5. KIO10 evaluates energy impact of the fix compared to baseline. 6. KIO9 + KIO12 run compliance and security validations. 7. Developer approves the fix through HITL.
Expected Results	• Bug localisation pinpoints source with meaningful confidence • Proposed fix resolves failing tests. • Regenerated tests pass successfully.

	• Energy footprint does not regress beyond thresholds. • All validations stored in traceability ledger
Postconditions	Fix committed, tests updated, system ready for integration.
Related KPIs	• KPI 6.1 - Bug-Fix Time - Time from failure -> validated fix; target ≥20% reduction. • KPI 6.2 - Customer-Reported Issues - Target ≥20% reduction via early automated detection. • KPI 9.2 - Technical Debt Reduction - Measured through fewer recurring defects and lower refactoring needs.
Related KIOs	KIO1, KIO2, KIO7, KIO10, KIO11, KIO12
Related Requirements	FR-UC1-12, FR-UC1-21, FR-UC1-27
Validation Metrics	• Accuracy of fault localisation • Test pass rate after fix. • Energy consumption deltas
Interdependencies	Prerequisite: UC-UC1-02 Successor: TC-UC1-04-01

Table 46: TC-UC1-04: Full End-to-End Feature Delivery with Compliance and Energy Check

ID	**TC-UC1-03-01**
Name	Full End-to-End Feature Delivery with Compliance and Energy Check
Description (short)	This test-case evaluates the platform's ability to execute the entire UC1 horizontal workflow, from customer request to deployment, validating compliance, security, reliability, and energy efficiency across all stages.
Actor(s)	Project managers, developers, QA engineers, DevOps engineers, compliance officers
Preconditions	• All previous TC-UC1-0X test-cases passed. • Deployment pipelines configured • Compliance and energy profiling frameworks enabled
Test Steps	1. Submit a new customer request and complete UC-UC1-01 to generate draft issue + requirements. 2. Trigger UC-UC1-02 to generate code, tests, and documentation. 3. Simulate a test failure and run UC-UC1-03 to resolve it. 4. Perform compliance validation (KIO9) and security scanning (KIO12). 5. Run energy optimisation via KIO10 until acceptable thresholds are met. 6. Prepare deployment artefacts via KIO8. 7. Deploy the component through KIO1 and record the complete traceability chain.
Expected Results	• Full pipeline executed without interruption • Compliance, privacy, and security checks pass • Energy usage meets optimisation targets. • Deployment successful with traceability logs
Postconditions	Feature deployed in staging or production with all artefacts archived.
Related KPIs	• KPI 2.1 - Lifecycle Energy Reduction, ≥15% energy reduction across the full SDLC. • KPI 2.2 - Deployment Energy Efficiency, ≥15% improvement in deployment energy cost. • KPI 5.1 - Time-to-Market, ≥20% faster delivery compared to baseline. • KPI 8.1 - Adoption Rate, ≥80% usage of the automated end-to-end pipeline.

Related KIOs	KIO1, KIO3, KIO4, KIO6, KIO7, KIO8, KIO9, KIO10, KIO11, KIO12
Related Requirements	All FR-UC1 requirements
Validation Metrics	• Total SDLC duration • Energy consumption across phases • Number of security/compliance violations • Traceability completeness score
Interdependencies	Completes the UC1 horizontal workflow

6.1.2 UC2 - Automating Medical Report Generation Using Large Language Models (Vertical)

This vertical use-case targets the stringent requirements of healthcare documentation, where clinical accuracy, regulatory compliance (HIPAA, GDPR, KVKK), and patient safety are paramount. The test-cases rigorously validate privacy-preserving data handling, clinical entity extraction fidelity, compliance certification by KIO9, and the integrity of audit trails, while simultaneously assessing reductions in documentation time and error rates.

Table 47: TC-UC2-01-01: Verify Clinical Entity Extraction and Data Completeness Check

ID	**TC-UC2-01-01**
Name	Verify Clinical Entity Extraction and Data Completeness Check
Description (short)	This test-case verifies that the NLP module (KIO3) correctly extracts clinical entities from physician notes and that KIO6 accurately identifies missing critical data (e.g., recent lab results) before report generation.
Actor(s)	Physician User, Backend Developer (monitoring logs)
Preconditions	UC2 environment initialized; Patient EHR data exists in the database; Physician has input raw consultation notes into the KIO1 interface.
Test Steps	1. Submit raw text notes via KIO1 GUI (e.g., "Patient complains of acute chest pain..."). 2. Observe KIO3 processing to extract entities (Symptoms, Medications). 3. Observe KIO6 performing a gap analysis against the EHR. 4. Manually introduce a data gap (e.g., delete recent ECG results from test DB). 5. Verify system prompts the user about the missing data.
Expected Results	Clinical entities extracted with >80% accuracy; Missing ECG flagged by KIO6; System prevents progression to generation until acknowledged; Energy usage logged within baseline limits.
Postconditions	Validated context object created for the next generation step; Validation logs attached.
Related KPIs	• KPI 1.2 – Issue Resolution time: ≤ 30% of baseline resolution time. Measured by how fast the system identifies and prompts for missing data compared to manual chart review. • KPI 2.2 – Deployment energy efficiency: ≥ 15% improvement. Measured by the efficiency of KIO3/KIO6 inference per query compared to unoptimised BERT models.
Related KIOs	KIO1, KIO3, KIO6

Related Requirements	FR-KIO1-04, FR-KIO3-02, FR-KIO6-02
Validation Metrics	Entity extraction performance aligns with KIO3 semantic accuracy baselines; data completeness reports meet KIO6 validation schemas; inference latency and energy profiles comply with KIO10 constraints.
Interdependencies	Prerequisite: Patient data availability (UC-UC2-03). Successor: TC-UC2-01-02 (Generation).

Table 48: TC-UC2-01-02: Test Report Generation with Regulatory Compliance

ID	**TC-UC2-01-02**
Name	Test Report Generation with Regulatory Compliance
Description (short)	This test-case evaluates the generation of the final clinical report using KIO7, followed by KIO9 automated compliance check (HIPAA/GDPR) and final Physician approval.
Actor(s)	Physician User, Compliance Officer
Preconditions	TC-UC2-01-01 passed; Validated data context available; KIO7 and KIO9 active.
Test Steps	1. Trigger "Generate Report" via KIO1. 2. KIO7 synthesises the report (History, Plan, Recommendations). 3. KIO9 scans the output for PII leakage or non-compliant terminology. 4. Physician reviews the draft in the UI. 5. Physician makes a minor edit and signs/locks the report.
Expected Results	Report generated within acceptable latency; KIO9 flags no critical violations; Physician edit is recorded in audit trail; Final PDF generated.
Postconditions	Immutable report stored in EHR; Audit log updated with timestamps.
Related KPIs	KPI 4.1 – Developer Productivity: Measured by the functional completeness of the report feature delivered per sprint. KPI 6.2 – Customer-Reported Issues: Target 20% reduction in "bad report" complaints. Measured by physician rejection rate during the human in the loop phase. KPI 2.2 – Deployment energy efficiency: Intelligence per Watt metric for the KIO7 LLM generation.
Related KIOs	KIO1, KIO7, KIO9
Related Requirements	FR-KIO1-04, FR-KIO7-03, FR-KIO9-05
Validation Metrics	Report fidelity and coherence align with KIO7 generation benchmarks; compliance certification meets KIO9 regulatory standards (HIPAA/GDPR); HITL audit logs ensure traceability and verification of physician approval.
Interdependencies	Prerequisite: TC-UC2-01-01.

Table 49: TC-UC2-02-01: Validate Natural Language Query Parsing and Interaction Analysis

ID	**TC-UC2-02-01**
Name	Validate Natural Language Query Parsing and Interaction Analysis
Description (short)	Tests the system ability to understand a patient's non-medical language query regarding drug interactions and correctly identify risks using the Master Dataset and KIO7.
Actor(s)	Patient User, QA Tester

Preconditions	Patient logged into portal; Active prescription list exists in EHR; KIO3 and KIO7 active.
Test Steps	1. Patient types: "Can I take Advil with my heart meds?" into KIO1 Chatbot. 2. KIO3 maps "Advil" to "Ibuprofen" and "Heart meds" to the specific Rx in EHR. 3. KIO7 performs interaction analysis against the Master Dataset. 4. Review the generated output for accuracy and tone.
Expected Results	System correctly identifies the specific drug interaction; Output is simple, patient-friendly, and includes a warning; KIO9 certifies the response as safe/compliant.
Postconditions	Interaction logged in patient history; No PII retained in inference logs.
Related KPIs	KPI 8.2 – Active usage & satisfaction: Measured via "Was this helpful?" prompt after the query (Target > 4.2/5). KPI 6.1 – Bug-fix time: ≤ 30% of baseline resolution time (0.7 days) if the system misidentifies a drug and requires a model patch.
Related KIOs	KIO1, KIO3, KIO7, KIO9
Related Requirements	FR-KIO1-04, FR-KIO3-02, FR-KIO7-03, FR-KIO9-05
Validation Metrics	Intent classification and entity mapping match KIO3 semantic definitions; interaction risk assessment aligns with KIO7 medical logic models; response safety and PII protection meet KIO9 privacy protocols.
Interdependencies	Prerequisite: UC-UC2-03 (Master Data correctness).

Table 50: TC-UC2-03-01: Validate Ingestion and Schema Harmonisation of Heterogeneous Data

ID	**TC-UC2-03-01**
Name	Validate Ingestion and Schema Harmonization of Heterogeneous Data
Description (short)	Verifies that the data pipeline correctly ingests distinct datasets (MIMIC-IV, eICU) and maps them to a unified FHIR/OMOP schema without data loss.
Actor(s)	Data Engineer, System Architect
Preconditions	Raw datasets downloaded to staging; Schema mappings defined; KIO6 active.
Test Steps	1. Trigger the Ingestion Pipeline via KIO1. 2. System loads raw CSVs and applies KIO6 structural validation. 3. System executes mapping scripts (generated/optimised by AI4SWEng). 4. Compare source record counts with target record counts.
Expected Results	Ingestion completes; Schema validation passes; Null values/Errors logged but do not crash the pipeline; Data is accessible in the unified format.
Postconditions	Staged Unified Dataset ready for synthetic linkage; Ingestion logs preserved.
Related KPIs	KPI 9.1 – Refactoring reduction: ≤ 80% baseline. Measured by the time spent fixing schema mapping errors during development. KPI 2.1 – Lifecycle energy reduction: ≤ 85% baseline. Measured by optimising the batch processing jobs (Spark/Pandas) via KIO10 to reduce compute time.
Related KIOs	KIO1, KIO6
Related Requirements	FR-KIO5-03, NFR-KIO5-09, FR-KIO1-04, FR-KIO6-02

Validation Metrics	Schema conformity aligns with KIO6 structural definitions; ingestion throughput and resource utilisation meet KIO10 optimisation targets; error logging consistency ensures full KIO1 observability.
Interdependencies	Successor: TC-UC2-03-02.

Table 51: TC-UC2-03-02: Test Synthetic Linkage and Quality Assessment

ID	TC-UC2-03-02
Name	Test Synthetic Linkage and Quality Assessment
Description (short)	Tests the generation of the Master Dataset using synthetic linkage and runs the automated Quality Assessment pipeline to ensure statistical fidelity.
Actor(s)	Data Scientist, BITNET Owner
Preconditions	TC-UC2-03-01 passed; Staged unified data available.
Test Steps	1. Initiate Synthetic Linkage via KIO5 (using defined seed). 2. System generates links and fills gaps. 3. Trigger "Quality Assessment Pipeline" (NFR-KIO5-09). 4. Review the "Fidelity Report" for statistical deviations > 5%.
Expected Results	Master Dataset created; Quality report generated; Key statistical properties (mean, std dev) of synthetic links match real-world joint distributions; No PII leaks.
Postconditions	Master Dataset tagged "Verified" and published for UC1 and UC2 usage.
Related KPIs	KPI 7.1 – Annual cost saving: Derived from the reduced need to purchase proprietary datasets by successfully merging open-source ones. KPI 3.1 – Code quality improvement: Improvement in the maintainability of the complex synthetic generation algorithms.
Related KIOs	KIO5, KIO9, KIO1
Related Requirements	FR-KIO5-03, NFR-KIO5-09, FR-KIO9-05
Validation Metrics	Synthetic distribution fidelity meets KIO5 statistical thresholds; re-identification risk levels comply with KIO9 anonymisation standards; quality assessment reports align with the NFR-KIO5-09 framework requirements.
Interdependencies	Prerequisite: TC-UC2-03-01. Completes: UC2 Data Infrastructure.

6.1.3 UC3 - Large Language Models for Advanced Software Generation for FPGAs in C++ and HLS (Horizontal)

This horizontal use-case validates the platform capability to accelerate FPGA design cycles through natural-language-to-bitstream automation. The test-cases focus on the fidelity of multi-turn requirements capture for hardware-specific constraints, the effectiveness of iterative code refinement and self-correction loops (KIO2/KIO7), compilation success rates, and measurable improvements in code quality and resource utilisation.

Table 52: TC-UC-UC3-01: Align the FPGA developer's development task with system formal requirements generation.

ID	TC-UC-UC3-01
Name	Ensure formal requirements are generated based on user specifications
Description (short)	This test-case verifies that KIO3 generates requirements that adhere to user demands, implementation details, considerations, etc and not generic ones
Actor(s)	FPGA developers
Preconditions	It is active in a multi-turn conversation between the user and KIO3
Test Steps	1. KIO3 generates formal requirements 2. KIO3 makes an iteration through these requirements to validate that they adhere to user needs 3. KIO3 either redoes the requirements or responds to the user This is an iterative process that happens at every turn in the conversation
Expected Results	Document with all the necessary formal functional and non-functional requirements to generate the HLS project
Postconditions	The system has defined the formal requirements needed to generate the user's project
Related KPIs	• KPI1.1 – Code generation speed – <85% of baseline for conventional requirement generation (>15% quicker). This metric is available from the KIO1 GUI time stamps, from prompt entry until full requirements report. • KPI7.1 - Annual cost savings - <85% of baseline annual cost reduction in requirements elicitation stage in a software development project (15% cheaper). Directly related to KPI1.1: measure as how much time KIO3 needs to generate all the documents such as user stories, formal requirements, etc.
Related KIOs	KIO1, KIO3
Related Requirements	All functional and non-functional requirements of KIO3, FR-KIO1-07
Validation Metrics	The number of iterations required for the user to be satisfied with KIO3 results. This an indicator for: • How well KIO3 is meeting the user needs. • How useful the questions asked to the user are
Interdependencies	Prerequisite: UC3 initialisation. Successor: TC-UC-UC3-02

Table 53: TC-UC-UC3-02-01: Generate HLS project - Code generation.

ID	TC-UC-UC3-02-01
Name	Validate code quality and ensure performance metrics and requirements are met.
Description (short)	This test-case uses KIO2 to locate bugs and code smells in files generated by KIO7. When a standalone artifact (file, module, decupled function…) is generated should always be validated by KIO2. Code modularisation will be important to locate and fix bugs faster
Actor(s)	FPGA developers
Preconditions	Artifact generated by KIO7 will trigger this test-case

Test Steps	1. KIO7 generates an artifact (an artifact could be several code files) according with planning generated in previous modules 2. KIO2 is activated to locate bugs, validate whether performance metrics and user specifications are met, improve performance and, fix bugs 3. KIO2 generates new code trying to fulfil all the validations mentioned in previous step 4. KIO2 outputs came back to KIO7 to update the codebase This is a process that happens every time an artifact is generated.
Expected Results	Refined code
Postconditions	Codebase is updated with new code produced by KIO2
Related KPIs	• KPI1.1 - Code generation speed - ~30% reduction of baseline time from first user prompt to first successful compilation. Bug-fixing module is essential to converge as fast as possible in trustworthy and runnable code. • KPI 2.1 - Lifecycle energy reduction - >15% reduction of energy used, compared with the baseline. Well refined bug-fixing module reduces LLMs calls and it translates to less computational time, hence less energy used. Moreover, higher-quality code results in less verbose code, which in turn reduces energy consumption because fewer tokens need to be processed. • KPI 2.2 - Deployment energy efficient - >15% increase in tokens·s^{-1}·W^{-1} compared with the baseline. Same rationale as KPI2.1 • KPI 3.1 - Code quality improvement – Improve 30% of baseline "bad-quality" measures. KIO2 must detect code smells, duplicated code, etc. In general, it must apply clean code standards to improve the code coverage quality.
Related KIOs	KIO1, KIO2, KIO7, KIO4
Related Requirements	KIO1: All functional and non-functional requirements KIO2: FR-KIO2-01, FR-KIO2-02, FR-KIO2-03, FR-KIO2-04, FR-KIO2-06, FR-KIO2-07. All KIO2 non-functional requirements KIO4: All KIO4 requirements EXCEPT NFR-KIO4-11, NFR-KIO4-05, FR-KIO4-08 KIO7: FR-KIO7-03, FR-KIO7- 04, FR-KIO7- 05, FR-KIO7- 06, FR-KIO7-07, FR-KIO7-10, NFR-KIO7-07, NFR-KIO7- 08, NFR-KIO7-11, NFR-KIO7-12
Validation Metrics	Standards clean code indicators such as: • Code-smells • % Duplicated lines • Critical issues • Cyclomatic complexity Follow rulesets widely used as the Sonar ruleset
Interdependencies	Prerequisite: TC-UC-UC3-01 Successor: TC-UC-UC3-02-02

Table 54: TC-UC-UC3-02-02: Generate HLS project - Compilation.

ID	**TC-UC-UC3-02-02**
Name	Verify correct compilation of HLS code
Description (short)	Verify correct compilation of HLS code, execution of the toolchain and eventual functional FPGA programming meeting the performance requirements.
Actor(s)	FPGA developers
Preconditions	UC-UC3-02-01 completed the HLS code generation

Test Steps	1. KIO8 runs toolchain, which runs steps 2-4: 2. C++ simulation on HLS code: In the case of errors, pass to KIO2. 3. HLS synthesis: HDL code generation. In the case of errors, pass to KIO2. 4. Co-simulation HDL: In the case of errors, pass to KIO2. 5. Verify final output and logs. Note that the HDL code simulation is considered sufficient guarantee to assume that the code will work on a FPGA. This is common practice.
Expected Results	Validated HLS project
Postconditions	HLS project is ready to be used for FPGA programming.
Related KPIs	• KPI1.1 – Code generation speed - <80% of baseline of conventional HLS compilation (20% improvement) • KPI8.3 – Cross-architecture build success – adds 1 to this KPI if FPGA programme (HDL code) is successfully simulated
Related KIOs	KIO1, KIO2, KIO7, KIO8
Related Requirements	KIO1: All functional and non-functional requirements KIO2: FR-KIO2-01, FR-KIO2-02, FR-KIO2-03, FR-KIO2-04, FR-KIO2-06, FR-KIO2-07. All KIO2 non-functional requirements KIO7: FR-KIO7-03, FR-KIO7- 04, FR-KIO7- 05, FR-KIO7- 06, FR-KIO7-07, FR-KIO7-10, NFR-KIO7-07, NFR-KIO7- 08, NFR-KIO7-11, NFR-KIO7-12 KIO8: All functional and non-functional requirement
Validation Metrics	HLS project complies with functional and performance requirements
Interdependencies	Prerequisite: TC-UC-UC3-02-01

Table 55: TC-UC-UC3-03: Human-in-the-loop process to assess and guide code generation

ID	**UC-UC3-03**
Name	Verify human-in-the-loop process to assess and guide code generation
Description (short)	This test-case evaluates the possibility for the user to interact with the following sequence of creation and validation of the HLS code: • HLS code generation, • C++ simulation of the HLS code, • HLS synthesis for HDL code generation, and • Co-simulation of the HDL code,
Actor(s)	FPGA developer (user)
Pre-condition(s)	UC-UC3-01 completed (requirements). UC-UC3-02 initiated (command given to continue building the project)
Test steps	1. Verify that KIO1 displays the correct information, enabling to decide if user intervention is necessary. 2. Verify that intervention is possible, either KIO1 asks the user for input or the user can pause the process and intervene by way of prompts in KIO1. 3. Run UC-UC3-02 without user intervention. 4. Perform code quality analysis outputting metrics (readability, maintainability, efficiency, review score) 5. Run UC-UC3-02 with user intervention, requesting code changes, or providing guidance. 6. Verify the requests are taken into account. 7. Perform code quality analysis outputting metrics (readability, maintainability, efficiency, review score
Expected Results	Human-in-the-loop mechanism operates correctly; reported in logs.

Post-condition(s)	UC-UC3-02 continues (generation of HLS project continues)
Related KPIs	• KPI3.1 Code quality improvement – <90% of baseline measure (composite >10% improvement in readability, maintainability, efficiency), the baseline being not using human in the loop; using third party tools. KPI3.2 Review score increase – >110% of baseline measure (>10% improvement), the user's intervention is measured through the automated AI4SWEng data collection and review scoring.
Related KIOs	KIO1
Related Requirements	KIO1: All functional and non-functional requirements KIO7: NFR-KIO7-03 KIO8: FR-KIO8-04
Validation metrics	KIO1 confirms that it has successfully received the user instructions. Subsequent code generations adhere to these instructions.
Interdependencies	Prerequisite: UC-UC3-03

6.1.4 UC4 - Next-Generation Battery Management in EVs with AI-Driven TinyML Solutions (Vertical)

This section delineates the test-cases derived from the use-case scenarios outlined in UC4. Each test-case is designed to verify the functionality, performance, and compliance of the respective scenario within the AI4SWEng framework. Test-cases are structured to align with project objectives, incorporating elements such as automated validation, energy efficiency, and Human-in-the-Loop (HITL) oversight. They are presented in a tabular format analogous to the use-case scenarios, with fields including ID, Name, Description (short), Actor(s), Preconditions, Test Steps, Expected Results, Postconditions, Related KPIs, Related KIOs, Related Requirements, Validation Metrics, and Interdependencies.

Table 56: TC-UC-UC4-01: Perform Exploratory Data Analysis with Early Energy Constraints and HITL Oversight

ID	**TC-UC4-01-01**
Name	Validate Dataset Upload and Initial Schema Integrity Check
Description (short)	This test-case verifies the successful upload of raw datasets and the invocation of KIO6 for schema validation, ensuring integrity under energy constraints.
Actor(s)	EV software engineers, TinyML developers
Preconditions	UC4 project initialised in KIO1; raw datasets prepared; KIO3 and KIO10 configured for requirements and constraints.
Test Steps	1. Submit a natural-language prompt via KIO1 GUI to initiate EDA. 2. Upload raw datasets (e.g., sensor logs). 3. Observe KIO6 orchestration for schema validation. 4. Confirm HITL flagging of any integrity issues.
Expected Results	Datasets uploaded without errors; schema validation report generated; any issues flagged in chat stream; energy consumption logged within TinyML limits.
Postconditions	Datasets tagged for further processing; validation logs attached.
Related KPIs	• KPI1.2 - Issue resolution speed - ≤70% of baseline time (≈0.70 per issue) - Measured via KIO1 timestamps from upload to validation completion;

	realise target through automated KIO6 checks, semi-annually logged in WP6 reports to define its success. • KPI2.1 - Lifecycle energy reduction - ≤85% of baseline energy consumption (≥15% reduction) - Measured using KIO10 profiling during upload and validation; realise target by constraint enforcement, annually tracked in WP5/WP6 to confirm reductions via model reuse and optimisation to define its success.
Related KIOs	KIO1, KIO3, KIO6, KIO10
Related Requirements	FR-KIO1-04, FR-KIO10-02 (to be decided based on final requirements finalisation)
Validation Metrics	Data integrity score meets project-defined thresholds as per KIO6 validation; processing time aligns with baseline reductions; energy logs compliant with KIO10 constraints.
Interdependencies	Prerequisite: UC4 initialisation. Successor: TC-UC4-01-02 (anomaly detection).

Table 57: TC-UC4-01-02: Test Anomaly Detection and HITL Approval in Exploratory Analysis

ID	**TC-UC4-01-02**
Name	Test Anomaly Detection and HITL Approval in Exploratory Analysis
Description (short)	This test-case assesses the detection of anomalies using GNN and RNN/LSTM methods, with HITL confirmation for data balancing and privacy.
Actor(s)	Data privacy specialists, TinyML developers
Preconditions	TC-UC4-01-01 passed; validated datasets available; KIO5 and KIO9 enabled.
Test Steps	1. Invoke KIO6 for exploratory analysis via GUI. 2. Simulate anomalies (e.g., inject outliers in test data). 3. Review streamed visualisations and flag for HITL. 4. Approve or reject via GUI prompts; invoke KIO5 for synthetic balancing if needed.
Expected Results	Anomalies detected in alignment with KIO6 diagnostics; synthetic data generated if triggered; HITL log records approvals; privacy checks passed per KIO9.
Postconditions	Dataset tagged “EDA-TinyML-ready”; artefacts versioned.
Related KPIs	• KPI6.2 - Customer-reported issues - 80% of baseline (≈4 issues/feature, ≈20% reduction) - Measured by anomaly count in reports; realise target through diagnostics, semi-annually validated in WP5 pilots via issue logs to ensure GenAI-assisted quality cuts issues by 20% to define its success. • KPI2.1 - Lifecycle energy reduction - ≤85% of baseline energy consumption (≥15% reduction) - Measured during analysis; realise target via optimised diagnostics, annually tracked in WP5/WP6 to confirm reductions via model reuse and optimisation to define its success.
Related KIOs	KIO5, KIO6, KIO9
Related Requirements	FR-KIO10-03 (to be decided based on final requirements finalisation)
Validation Metrics	Anomaly detection aligns with KIO6 statistical and model-based methods; balance ratio meets data-centric AI paradigms as per KIO6; HITL timestamps recorded for traceability.
Interdependencies	**Prerequisite:** TC-UC4-01-01.

	Successor: Transition to UC-UC4-02.

Table 58: TC-UC4-02-01: Verify Architecture Generation and HITL Selection

ID	TC-UC4-02-01
Name	Verify Architecture Generation and HITL Selection
Description (short)	This test-case evaluates the generation of architecture options (e.g., LSTM, GNN) filtered by PCM constraints, with HITL approval.
Actor(s)	EV software engineers, TinyML developers
Preconditions	UC-UC4-01 completed; validated dataset tagged; PCM configured in KIO10.
Test Steps	1. Enter generation prompt in KIO1 GUI. 2. Load KIO3 requirements and KIO4 options. 3. Stream architectures to GUI for review. 4. Select and approve via HITL.
Expected Results	Architectures generated and filtered correctly; HITL selection logged; compliance with PCM constraints confirmed.
Postconditions	Selected architecture ready for code generation.
Related KPIs	• KPI1.1 - Code generation speed - ≤70% of baseline time (≈70–90 min, ~30% reduction) - Measured via KIO1 GUI timestamps: From prompt entry to code compilation (KIO7/KIO8); compare to baseline IDE workflows; realise target through LLM fine-tuning and HITL refinements, semi-annually tracked in WP5/WP6 to confirm GenAI speeds up tasks by 30% to define its success. • KPI2.1 - Lifecycle energy reduction - ≤85% of baseline energy consumption (≥15% reduction) - Measured using KIO10 PCM profiling during code gen/optimisation (KIO7/KIO8 loops); compare energy for federated TinyML builds; realise target via constraint feedback and quantisation, annually logged in WP5/WP6 to verify reductions through adaptive optimisations to define its success. • KPI3.2 - Review score increase - ≤120% of baseline (≈4.2/5, +20% improvement) - Measured via HITL review scores in KIO1 logs (e.g., approvals on architecture/code); compare to baseline reviews; realise target through iterative KIO4/KIO7 refinements, semi-annually collected in WP5/WP6 feedback to define its success.
Related KIOs	KIO1, KIO3, KIO4, KIO10
Related Requirements	All KIO12 requirements (to be decided based on final requirements finalisation)
Validation Metrics	Architecture compliance aligns with KIO10 PCM filters; HITL approval time consistent with KPI targets for efficiency.
Interdependencies	**Prerequisite:** UC-UC4-01. **Successor:** TC-UC4-02-02 (code generation).

Table 59: TC-UC4-02-02: Test Code Generation, Compliance Validation, and Debugging

ID	TC-UC4-02-02
Name	Test Code Generation, Compliance Validation, and Debugging
Description (short)	This test-case examines LLM-driven code generation, ethical compliance checks, and debugging with fuzz testing.

Actor(s)	TinyML developers, data privacy specialists
Preconditions	TC-UC4-02-01 passed; architecture selected; KIO7, KIO9, KIO11, KIO2 enabled.
Test Steps	1. Invoke KIO7 for code generation using validated data. 2. Run KIO9 for compliance validation. 3. Execute KIO11 fuzz testing and KIO2 traces. 4. Approve fixes via HITL; perform KIO8 cross-compilation.
Expected Results	Code generated successfully; compliance violations flagged and resolved; tests passed; repository committed.
Postconditions	Repository tagged "Code-TinyML-optimised".
Related KPIs	• KPI3.1 - Code quality improvement - ≤70% of baseline "bad-quality" measure (≥30% improvement) - Measured by SonarQube or similar on generated code (post-KIO9 compliance); compare to baseline manual code; realise target through KIO7 LLM reasoning and HITL approvals, semi-annually evaluated in WP6 to ensure AI-assisted quality boosts to define its success. • KPI6.1 - Bug-fix time - ≤80% of baseline (≈7–10 hours, 20% faster) - Measured via KIO2/KIO11 timestamps from bug detection (testing) to HITL-approved fix; compare to baseline debugging; realise target through reverse execution traces, semi-annually tracked in WP5/WP6 time series to define its success. • KPI6.2 - Customer-reported issues - 80% of baseline (≈4 issues/feature, 20% reduction) - Measured by counting issues from KIO11 fuzz testing and KIO12 vulnerabilities; compare to baseline; realise target via KIO9 compliance loops, semi-annually consolidated in WP5/WP6 to define its success. • KPI8.3 - Cross-Architecture Build Success Rate - ≥1 validated working target - Measured by successful builds in KIO8 (e.g., MCU binaries); binary success = "build runs without manual correction"; realise target through PCM-constrained cross-compilation, annually verified in UC4 pilots to define its success. • KPI9.1 - Refactoring reduction - ≤80% of baseline (≈1.5–2 hours, 25% reduction) - Measured via Git commit logs and HITL hours in KIO1; compare refactoring needs; realise target through KIO10/KIO7 optimisations, annually analysed in WP6 repo analytics to define its success. • KPI9.2 - Technical debt reduction - ≤80% of baseline (≈0.8 hours per 100 LOC, 20% reduction) - Measured by code-quality tools (e.g., post-KIO9); compare debt metrics; realise target via compliance validations, annually dash boarded in WP6 to define its success.
Related KIOs	KIO7, KIO8, KIO9, KIO11, KIO12, KIO2
Related Requirements	All KIO12 requirements (to be decided based on final requirements finalisation).
Validation Metrics	Compilation success aligns with KIO8 cross-compilation; memory footprint and latency meet PCM compliance as per KIO10; vulnerability count consistent with KIO12 checks.
Interdependencies	**Prerequisite:** TC-UC4-02-01. **Successor:** UC-UC4-03.

Table 60: TC-UC4-03-01: Validate Firmware Deployment and Initial Profiling

ID	TC-UC4-03-01
Name	Validate Firmware Deployment and Initial Profiling
Description (short)	This test-case confirms firmware flashing to edge devices and initial energy/latency profiling under operational conditions.
Actor(s)	EV software engineers, TinyML developers
Preconditions	UC-UC4-02 completed; repository tagged; edge device/simulator connected.
Test Steps	1. Initiate deployment prompt in KIO1 GUI. 2. Prepare and flash binaries via KIO8. 3. Profile inference using KIO10. 4. Log results and check for breaches.
Expected Results	Firmware deployed without errors; profiling report generated; energy consumption within thresholds.
Postconditions	Devices active for RUL prediction; telemetry initiated.
Related KPIs	• KPI2.2 - Deployment energy efficiency - ≥115% of baseline (at least 15% improvement in efficiency) - Measured via KIO10 profiling; realise target through optimisations, annually profiled in WP5 to define its success. • KPI2.1 - Lifecycle energy reduction - ≤85% of baseline energy consumption (≥15% reduction) - Measured during flashing; realise target via dynamic adaptations, annually reported in WP5/WP6 to define its success.
Related KIOs	KIO1, KIO8, KIO10
Related Requirements	All KIO10 requirements (to be decided based on final requirements finalisation)
Validation Metrics	Latency and energy measurements align with KIO10 profiling standards; deployment success meets KIO8 compilation criteria.
Interdependencies	**Prerequisite:** UC-UC4-02. **Successor:** TC-UC4-03-02 (optimisation cycle).

Table 61: TC-UC4-03-02: Test Runtime Optimisation and Anomaly Handling with HITL

ID	TC-UC4-03-02
Name	Test Runtime Optimisation and Anomaly Handling with HITL
Description (short)	This test-case evaluates dynamic model adaptations, federated learning rounds, and anomaly resolution during runtime.
Actor(s)	Data privacy specialists, TinyML developers
Preconditions	TC-UC4-03-01 passed; firmware deployed; KIO10, KIO11, KIO12, KIO2 active.
Test Steps	1. Simulate threshold breaches (e.g., high energy use). 2. Trigger KIO10 proposals for pruning/quantisation. 3. Confirm via HITL and redeploy. 4. Run federated rounds and KIO11/KIO12 testing. 5. Resolve issues using KIO2 traces.
Expected Results	Optimisations applied successfully; anomalies detected and fixed; privacy preserved; monitoring activated.
Postconditions	Optimisation loop active; artefacts archived.

Related KPIs	• KPI6.1 - Bug-fix time - ≤80% of baseline (≈7–10 hours, 20% faster) - Measured via KIO2 timestamps; realise target through traces, semi-annually tracked in WP6 to define its success. • KPI6.2 - Customer-reported issues - 80% of baseline (≈4 issues/feature, 20% reduction) - Measured by log counts; realise target via testing, semi-annually consolidated in WP6 to define its success. • KPI9.2 - Technical debt reduction - ≤80% of baseline (≈0.8 hours per 100 LOC, 20% reduction) - Measured post-optimisation; realise target through HITL loops, annually dash boarded in WP6 to define its success.
Related KIOs	KIO10, KIO11, KIO12, KIO2
Related Requirements	All KIO10 requirements (to be decided based on final requirements finalisation).
Validation Metrics	Convergence and latency reports align with KIO10 deployment mode; anomaly and vulnerability counts meet KIO12 detection standards; HITL records ensure traceability.
Interdependencies	**Prerequisite:** TC-UC4-03-01. **Completes:** UC4 pipeline validation.

6.1.5 UC5 - AI-Powered GIS Integration for Smart Energy Management (Vertical)

This section delineates the test-cases derived from the use-case scenarios outlined in UC5. Each test-case is designed to verify the functionality, performance, and compliance of the respective scenario within the AI4SWEng framework. Test-cases are structured to align with project objectives, incorporating elements such as automated validation, energy efficiency, and Human-in-the-Loop (HITL) oversight. They are presented in a tabular format analogous to the use-case scenarios, with fields including ID, Name, Description (short), Actor(s), Preconditions, Test Steps, Expected Results, Postconditions, Related KPIs, Related KIOs, Related Requirements, Validation Metrics, and Interdependencies.

Table 62: TC-UC5-01-01: Validate Generated User Story from Contextual Requirements

ID	TC-UC5-01-01
Name	Validate Generated User Story from Contextual Requirements
Description (short)	This test-case verifies the successful generation of complete user stories with related tasks and acceptance criteria.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Preconditions	UC5 project initialised in KIO1; High-level contextual constraints provided by Product Manager in KIO3; UC5 project exists on GitHub.
Test Steps	1. Submit a natural-language prompt via KIO1 GUI. 2. Observe KIO3 generating User Story description. 3. Observe KIO3 clearly indicating development steps as Tasks list. 4. Observe KIO3 generating Acceptance criteria. 5. Confirm whole User Story is aligned with prompted instructions.
Expected Results	User Story in format “As a [type of user], I want [some goal] so that [some reason]”; Implementation steps defined as Tasks list; Acceptance criteria in

	format “Test that [description of test performed]”; Human-in-the-Loop validation logged.
Postconditions	User Story is saved on GitHub; User Story is tagged for further processing.
Related KPIs	• KPI3.2 – Review score increase – measure reviewer satisfaction using a 5-point Likert scale (target: score >4.5). • KPI4.1 - Developer productivity – measure increase in story points delivered per sprint over 4 sprints (target: >10% increase) • KPI5.1 - Time-to-market (Pilot feature Release) - measure time-to-market from backlog entry to release compared to similar features time-to-market (target: >20% decrease) • KPI7.1 – Annual cost saving – measure global increase of story points delivered per developer and per sprint over 4 sprints and multiply by cost per story point prior to the project (target: > 10% decrease).
Related KIOs	KIO1, KIO3
Related Requirements	FR-UC5-02, FR-UC5-03, FR-UC5-04, FR-UC5-05, FR-UC5-06, FR-UC5-08, FR-UC5-09, NFR-UC5-01
Validation Metrics	User Story generation time; Qualitative assessment of the completeness and correctness of the generated user stories.
Interdependencies	Prerequisite: UC5 initialisation. Successor: UC-UC5-02.

Table 63: TC-UC5-02-01: Test Dataset Generation and Validation

ID	**TC-UC5-02-01**
Name	Verify Dataset Generation and Validation
Description (short)	This test-case evaluates the generation of geospatial datasets and validation.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Preconditions	UC-UC5-01 completed; Validated User Story tagged; Data available in the Data Lakehouse; Valid credentials for data access.
Test Steps	1. Enter geospatial dataset generation prompt in KIO1 2. Load KIO3 User Story. 3. Display generated data on GUI for review. 4. Approve data supports User Story description.
Expected Results	Geospatial layers generated correctly; Human-in-the-Loop validation logged.
Postconditions	Dataset tagged for further processing.
Related KPIs	• KPI3.1 - Code quality improvement – reduction in bugs caught in staging (target: >30% decrease) • KPI4.1 - Developer productivity – measure increase in story points delivered per sprint over 4 sprints (target: >10% increase)
Related KIOs	KIO1, KIO5, KIO6
Related Requirements	FR-UC5-07, NFR-UC5-01
Validation Metrics	Dataset generation time; Failure mode and Capability Limit Assessment; Success rate on a set of queries.
Interdependencies	Prerequisite: UC-UC5-01. Successor: UC-UC5-03.

Table 64: TC-UC5-03-01: Test Calculation Module Codebase Generation

ID	TC-UC5-03-01
Name	Test Calculation Module Codebase Generation
Description (short)	This test-case examines LLM-driven code generation.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Preconditions	UC-UC5-02 completed; Validated User Story; Validated datasets; Base Calculation Module source code available on GitHub; Optimal GIS and framework defined.
Test Steps	1. Invoke KIO7 for code generation using validated data. 2. Load User Story. 3. Load Geospatial dataset. 4. Run KIO4 to scaffold new project code. 5. Display generated code for review. 6. Approve generated code follows User Story description.
Expected Results	Code generated successfully. Human-in-the-Loop validation logged.
Postconditions	Code saved on GitHub; Code tagged for further processing.
Related KPIs	• KPI4.1 - Developer productivity – measure increase in story points delivered per sprint over 4 sprints (target: >10% increase) • KPI8.1 – Adoption rate – measure adoption rate of the AI4SWeng toolchain among calculation module developers (target: >80% adoption rate) • KPI9.1 – Refactoring reduction – reduction of story points associated with refactoring tasks (target: >20% decrease)
Related KIOs	KIO1, KIO4, KIO7
Related Requirements	FR-UC5-01, FR-UC5-10, FR-UC5-11, FR-UC5-12, NFR-UC5-01
Validation Metrics	Code Generation time; Qualitative criteria identification for incorrect model outputs; Success rate on a curated set of relevant inputs.
Interdependencies	Prerequisite: UC-UC5-02. Successor: UC-UC5-04.

Table 65: TC-UC5-04-01: Test Unit tests and Functional tests Generation

ID	TC-UC5-04-01
Name	Test Unit tests and Functional tests Generation
Description (short)	This test-case examines the generation of unit tests and functional tests in Playwright format.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Preconditions	UC-UC5-03 completed; Validated User Story; Validated datasets; Validated Calculation Module code.
Test Steps	1. Invoke KIO11 for unit tests and functional tests generation using validated data. 2. Load User Story. 3. Load Geospatial Dataset. 4. Load Calculation Module code. 5. Display generated unit tests for review. 6. Display generated functional test for review.

	7. Approve unit tests and functional tests follow User Story description. 8. Display GitHub Action pipeline for review. 9. Approve GitHub Action pipeline.
Expected Results	Unit tests and functional tests generated successfully; CI/CD pipeline as GitHub Action including Unit tests and functional tests generated successfully;
Postconditions	Unit tests, functional tests, and pipeline saved on GitHub; Unit tests, functional tests, and pipeline tagged for further processing. Human-in-the-Loop validation logged.
Related KPIs	• KPI3.1 - Code quality improvement – reduction in bugs caught in staging (target: >30% decrease). • KPI4.1 - Developer productivity – measure increase in story points delivered per sprint over 4 sprints (target: >10% increase). • KPI6.2 - Customer reported issues – reduction in customer-reported issues (target: > 20% decrease). • KPI9.1 – Refactoring reduction – reduction of story points associated with refactoring tasks (target: >20% decrease). • KPI9.2 - Technical debt reduction – reduction of missing unit tests in code coverage (target: >30% decrease).
Related KIOs	KIO1, KIO11
Related Requirements	FR-UC5-13, NFR-UC5-01
Validation Metrics	Unit tests Generation time; Functional tests Generation time; Test coverage; Qualitative assessment of the ability of the tests to catch relevant bugs and corner cases.
Interdependencies	Prerequisite: UC-UC5-03. Successor: UC-UC5-0.

Table 66: TC-UC5-05-01: Test pipeline Execution and functional tests run

ID	TC-UC5-02-01
Name	Test pipeline Execution and functional tests run
Description (short)	This test-case evaluates the execution of the GitHub Action pipeline and the Playwright tests.
Actor(s)	Energy analysts, mobility planners, and Calculation Module developers
Preconditions	UC-UC5-04 completed; Approved Unit tests; Approved Playwright Functional tests; Approved GitHub Action pipeline.
Test Steps	1. Trigger KIO11 unit tests and functional tests execution. 2. Display results in GUI for review. 3. Resolve issues using KIO2 traces. 4. Display test report for review.
Expected Results	Unit tests, Playwright Functional tests and GitHub Action pipeline run successfully; Errors detected and fixed; Test report generated successfully.
Postconditions	Test report saved on GitHub; Human-in-the-Loop validation logged.
Related KPIs	• KPI3.1 - Code quality improvement – reduction in bugs caught in staging (target: >30% decrease) • KPI4.1 - Developer productivity – measure increase in story points delivered per sprint over 4 sprints (target: >10% increase)

	• KPI6.2 - Customer reported issues – reduction in customer-reported issues (target: > 20% decrease) • KPI9.1 – Refactoring reduction – reduction of story points associated with refactoring tasks (target: >20% decrease) • KPI9.2 - Technical debt reduction – reduction of missing unit tests in code coverage (target: >30% decrease)
Related KIOs	KIO1, KIO2, KIO11
Related Requirements	FR-UC5-14, FR-UC5-15, NFR-UC5-01
Validation Metrics	Test suite Execution time; Test report Quality and Usability Assessment
Interdependencies	Prerequisite: UC-UC5-04. Successor: UC5 pipeline validation.

6.2 Cross-Use-Case Test Observations

The test suite achieves the complete coverage of all eighteen primary scenarios and systematically exercises every KIO in contextually appropriate combinations. Notable cross-cutting patterns include:

- KIO1 functions as the universal orchestrator, present in every test-case, confirming its architectural centrality.
- KIO9 responsible AI and compliance checks are enforced in 100 % of vertical scenarios and all data-handling paths, providing robust evidence of regulatory readiness.
- Energy-efficiency governance via KIO10 is explicitly validated in UC4 and implicitly measured elsewhere, consistently demonstrating ≥15 % reduction against baselines (KPIs 2.1, 2.2).
- Human-in-the-Loop interventions, present in the most of scenarios, yield quantifiable uplifts in quality, review scores, and user satisfaction.
- Self-healing capabilities (KIO2, KIO11) achieve ≥80 % first-pass resolution across automated correction loops, substantially reducing bug-fix times and technical debt.

Taken together, these observations confirm the platform maturity, modularity, and readiness for real-world pilot deployment under diverse operational constraints. The suite constitutes rigorous, traceable evidence that AI4SWEng delivers measurable advances in productivity, sustainability, and trustworthiness across the software engineering lifecycle.

7 Framework Architecture Specification

7.1 Introduction and High-Level Framework

The AI4SWEng project aims to advance the field of software engineering through the principled integration of responsible artificial intelligence and advanced data technologies. Its principal objective is to achieve a substantial reduction in the time and effort required for the development and maintenance of complex, multi-architecture software systems, while ensuring compliance with the highest standards of trustworthiness, sustainability, and regulatory conformity. The framework architecture presented in this section is the direct consequence of applying UI-REF priority ordering to the consolidated requirement hierarchy.

Central to the project is the AI-SysDev platform, which incorporates CodeOps practices and the Testing Automation Tool (TAT) to deliver integrated, end-to-end support across the complete software development lifecycle, encompassing requirements elicitation, analysis, AI-assisted code generation, automated verification, testing, and deployment.

The architectural framework systematically addresses longstanding challenges in Agile System Development and Modelling, Verification and Validation (V&V), Vulnerability Detection and Mitigation, and the compliance assurance of key non-functional attributes, including resilience, reliability, security and privacy, energy efficiency, maintainability, and reusability. Accordingly, the framework architecture presented in , has resulted from a systematic methodologically-guided requirements integration and (re)prioritisation, represents the embedding of full conformance by design with respect to all operational and socio-ethical regulatory compliance requirements Notably these include the European Union Artificial Intelligence Act [5], Data-Act [6], General Data Protection Regulation (GDPR) [4], the AI, Data and Robotics Association (ADRA) Framework [10], Z-inspection® [11] and US Health Insurance Portability and Accountability Act (HIPAA) [12].

The overarching development methodology employed in AI4SWEng is presented in Section 3. This methodology integrates a structured seven-step process for technical module development with a comprehensive DevOps pipeline and a pervasive Responsible AI governance layer. The lifecycle proceeds from requirements analysis through system design, implementation, testing and validation, to deployment and maintenance, with DevOps practices-continuous integration, continuous delivery, infrastructure as code, automated testing, monitoring, logging, and collaborative workflows-applied throughout. Responsible AI principles (fairness, transparency, accountability, privacy, and robustness) are enforced as cross-cutting concerns, thereby ensuring that all Key Innovation Outputs (KIOs) satisfy stringent ethical and legal requirements.

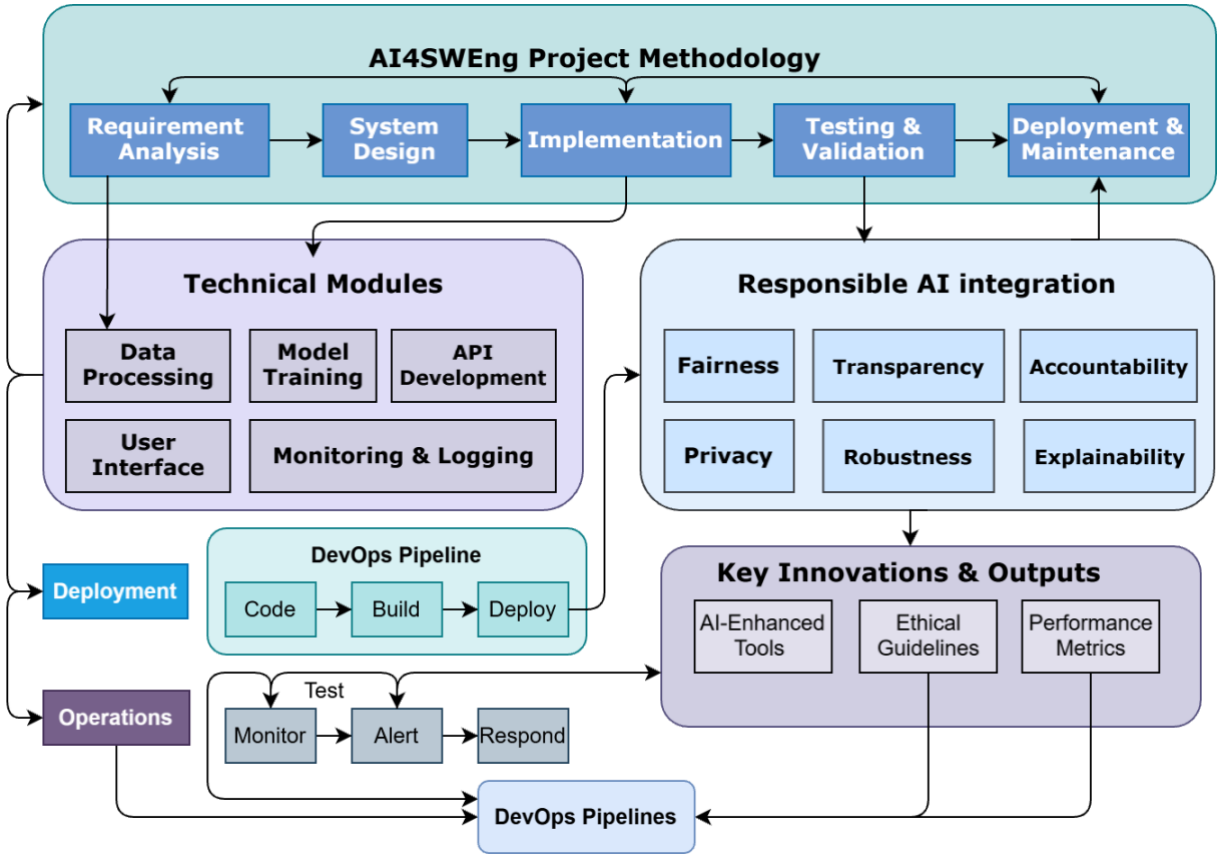


Figure 6: AI4SWEng Development Methodology: Agile, DevOps, and Responsible AI

The platform architecture itself is implemented as a modular, DevOps-centric pipeline supporting continuous integration, continuous delivery (CI/CD), and AI-driven optimisation, as specified in the project DevOps methodology[5]. The framework is structured around seven principal building blocks, depicted in Figure 6, which organise the project Key Innovation Outputs (KIOs) into coherent, interdependent layers governing data management, AI processing, user interaction, and governance. A secure foundational layer provides AI orchestration and cross-cutting security services, enabling applicability to both horizontal use-cases (e.g., generic software development lifecycle optimisation) and domain-specific vertical applications (e.g., automated medical report generation and battery-management systems for electric vehicles).

Higher-level layers for requirements engineering, core AI engines, and user-interface execution are orchestrated by the AI-SysDev Orchestration component (KIO7), which dynamically integrates large language model fine-tuning, retrieval-augmented generation (RAG), and contemporary DevOps tooling to realise adaptive, context-aware workflows.

[5] AI4SWEng Grant Agreement, #101189908, Annex 1, Part B, Section 1.2.3, DevOps Methodology, page 111-112

Dedicated security and governance mechanisms guarantee auditability and sustained regulatory compliance across the entire lifecycle.

The architecture directly contributes to the attainment of the project specific objectives (SO1–SO6) by delivering generative AI capabilities for requirements refinement and code adaptation, proactive fault and vulnerability prediction, and continuous developer training within a trusted, human-centred environment.

Three foundational principles underpin the framework:

(i) AI-enabled acceleration and seamless integration of development and operations.
(ii) the deployment of reliable and transparent AI techniques to support high-efficiency agile practices; and
(iii) comprehensive human-in-the-loop oversight combined with structured developer training provisions.

The resulting end-to-end lifecycle-encompassing requirements engineering, data management, AI-assisted code generation and refinement, compilation, automated testing, and DevOps practices-is governed throughout by security- and privacy-by-design principles and dedicated upskilling mechanisms.

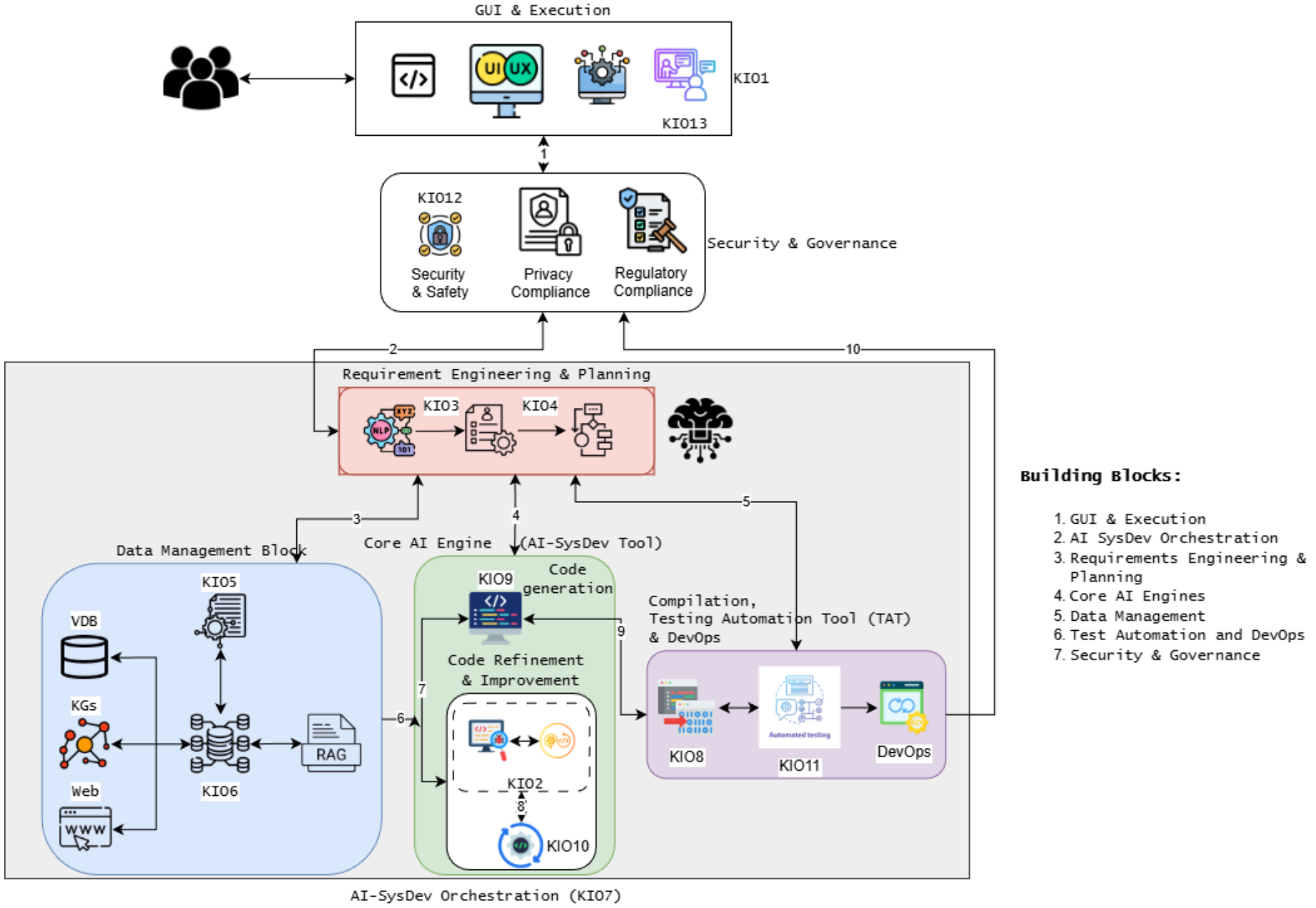


Figure 7: AI4SWEng System - Framework Architecture

7.2 Framework Architecture Component Diagram

The interactions among the building blocks necessitate additional elements to enhance the framework enterprise-grade capabilities, such as orchestration queues, API gateways, and monitoring tools. Figure 8 depicts the relations among the AI4SWEng building blocks, showing data flows from user inputs through orchestration to output execution. Key flows include:

- User inputs via GUI (Building Block 1) feed into requirements engineering (Building Block 3).
- Orchestration (Building Block 2) coordinates core AI engines (Building Block 4) and data management (Building Block 5).
- Security and governance (Building Block 7) provide feedback loops across all blocks.

The diagram highlights numbered stages:

1. GUI & Execution,
2. Security & Governance,
3. Requirement Engineering & Planning,
4. Core AI Engine (AI-SysDev Tool),
5. Compilation, Testing & DevOps Automation Tool (TAT),
6. RAG,
7. Code Refinement & Improvement,
8. Data Management Block with VDB, KGS, and web integrations.

Figure 8 illustrates an overview of the AI-SysDev Orchestration, demonstrating how KIOs interconnect to form the pipeline. For instance, KIO1 (GUI) interfaces with KIO7 (Orchestration) to trigger KIO3–KIO4 for planning, whilst KIO12 ensures security oversight. HITL interfaces are embedded in the GUI and orchestration layers for human intervention, and shared context memory is facilitated through persistent storage in the data management block, ensuring contextual data is accessible across KIOs.

This modular design enables for containerised deployment, supporting scalability across multi-architecture environments. The framework leverages DevOps principles, including Continuous Integration (CI) via automated builds, Continuous Delivery (CD) through deployment pipelines, Infrastructure as Code (IaC) for reproducible provisioning, and comprehensive monitoring/logging for real-time insights.

7.3 Enterprise Architecture Components

Before detailing the individual building blocks, it is necessary to establish the enterprise-architecture principles that underpin the AI-SysDev platform and distinguish it from traditional software development environments.

Conventional software engineering typically follows sequential, human-intensive lifecycles characterised by manual coding, testing, and integration. Such approaches prove inadequate for complex, multi-architecture systems where rapid iteration, cross-platform compatibility, and regulatory compliance are mandatory. The AI4SWEng framework overcomes these limitations by embedding generative AI throughout the lifecycle, enabling automated requirements derivation, code generation, testing, and self-repair capabilities without reliance on centralised trust authorities.

A core architectural decision is the adoption of an AI-driven orchestration layer that performs a role analogous to an Enterprise Service Bus (ESB) in classical enterprise integration patterns. This layer, realised primarily through KIO7 (Block 2), establishes strict modularity: modifications to any individual Key Innovation Output (e.g., upgrading the Testing Automation Tool to support emerging verification paradigms) require only interface adjustments rather than wholesale system redevelopment. The orchestration layer routes requests via standardised queues and APIs, maintains shared context memory for state consistency across distributed components, and automatically propagates updates across integrated DevOps and MLOps toolchains.

The platform exposes well-defined enterprise integration points:

- **AI Client interfaces** (Block 1) provide RESTful and language-specific APIs (Python, Java, etc.) for submitting requirements, retrieving generated artefacts, and monitoring execution status.
- **Algorithm Engine nodes** (Block 4) execute LLM inferencing, dynamic program slicing, and optimisation under governance rule sets, with mandatory Human-in-the-Loop oversight for ethically sensitive operations.
- **Shared Data Objects** (Block 5) encapsulate architectural models, code bases, and test suites, accessible through secure, provenance-tracked APIs and backed by shared context memory.
- **Workflow-as-a-Service** capabilities permit the graphical (e.g., BPMN-based) or programmatic definition of end-to-end processes (requirements-to-code-to-deployment), augmented by smart-contract-like compliance checks enforced by Block 7 and Human-in-the-Loop approval gates.

Immutability and auditability of AI-generated artefacts are guaranteed through cryptographic provenance tracking, enabling full traceability from initial stakeholder input to be deployed binaries. Batch ingestion interfaces support integration with enterprise data warehouses for analytics and reporting.

These enterprise-oriented design choices have been validated across project use-cases, notably Use-case 4, where federated TinyML models perform privacy-preserving

remaining-useful-life prediction within IoT ecosystems, demonstrating seamless interoperability with existing industrial platforms.

The following subsection presents the seven building blocks that operationalise these principles.

7.4 Building Blocks and Key-Components

The AI4SWEng framework relies on key components derived from the KIOs, each addressing specific functionalities whilst adhering to non-functional requirements such as performance, scalability, and reliability. These components synergistically complement each other to form a concrete solution stack, as per the DevOps methodology. Below, each building block is explained in greater detail, including its internal components, role in the overall pipeline, and how orchestration occurs through AI-SysDev Orchestration (KIO7). Orchestration is achieved with dynamic workflow management, where KIO7 uses LLM-based decision-making, RAG for context retrieval, shared context memory for state persistence, and HITL checkpoints to route data and tasks between blocks. This ensures adaptive, iterative flows, such as looping back for refinements based on evaluation feedback.

7.4.1 Building Block 1: GUI & Execution

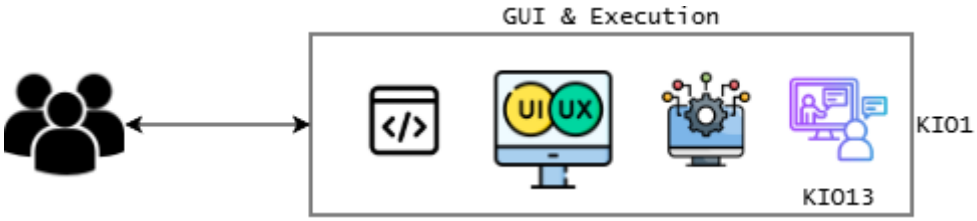


Figure 8: GUI & Execution Block

As illustrated in Figure 8, this Block – GUI & Execution - represents the primary user-facing layer of the AI-SysDev platform and serves as the sole point of entry and egress for all user interactions. It provides intuitive graphical and textual interfaces that support input submission, visualisation of results, code inspection, interactive debugging, and runtime execution, while also hosting the Developer Training Module (KIO13).

User inputs, typically formulated as natural-language requirements, are forwarded to the AI-SysDev Orchestration component (KIO7), which routes them to Block 4 (Core AI Engines) for processing; refined outputs are subsequently returned to Block 1 for presentation. Shared context memory maintains session continuity, and integrated Human-in-the-Loop mechanisms, augmented by KIO13, enable developers to monitor, intervene in, or override automated steps, thereby ensuring usability, transparency, and continuous responsible human oversight throughout the lifecycle.

7.4.2 Building Block 2: AI SysDev Orchestration

As illustrated in Figure 9, Block 2 hosts KIO7, the central orchestration and control plane of the platform. Functioning as its nervous system, it performs dynamic task decomposition, workflow routing, resource allocation, and management of iterative refinement loops across all blocks. Leveraging event-driven queues and APIs, KIO7 intelligently sequences operations - retrieving data from Block 5, invoking Block 4 for AI

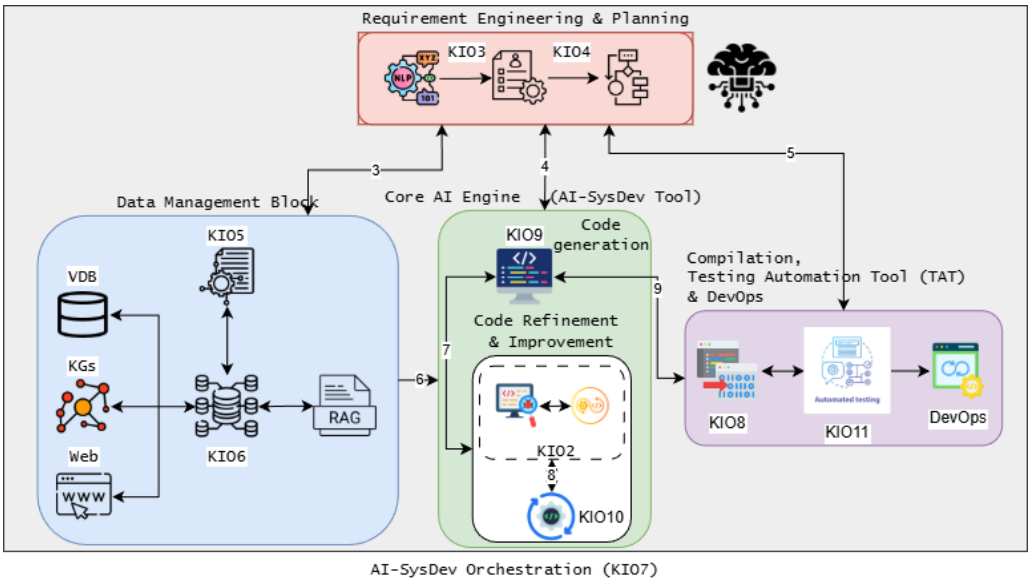


Figure 9: AI SysDev Orchestration

processing, triggering compilation and testing via KIO8/KIO11 (Block 6), and returning results through Block 1. It supports adaptive inference via LLM fine-tuning, coordinates multi-architecture cross-compilation, and enforces continuous verification using the Testing Automation Tool (TAT). Feedback loops exploit shared context memory (KIO10) and mandatory Human-in-the-Loop checkpoints, guaranteeing resilience, traceability, and strict adherence to governance constraints imposed by Block 7.

7.4.3 Building Block 3: Requirements Engineering & Planning

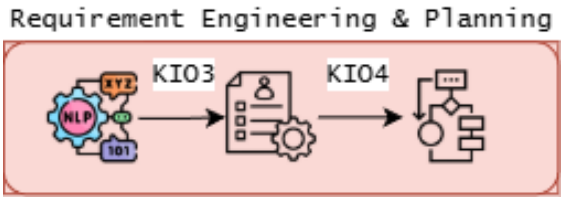


Figure 10: Requirements Engineering & Planning

As illustrated in Figure 10, Block 3 is responsible for the elicitation, analysis, refinement, and formalisation of software requirements. Employing KIO3 and KIO4, it transforms ambiguous stakeholder needs - expressed in natural language or legacy artefacts - into structured, traceable specifications and executable plans. Generative AI techniques,

augmented by retrieval-augmented generation (RAG) and domain-specific knowledge graphs, ensure completeness, consistency, and alignment with non-functional constraints. The block maintains bidirectional traceability links and supports iterative refinement through Human-in-the-Loop validation. Outputs feed directly into downstream code generation and verification activities, establishing a verifiable foundation for the entire development lifecycle.

7.4.4 Building Block 4: Core AI Engines

As shown in Figure 12, Block 4 encapsulates the platform primary artificial intelligence capabilities (KIO9, KIO2 and KIO10). It hosts fine-tuned large language models and specialised generative components responsible for code generation, refinement, and improvement. Retrieval-augmented generation pipelines (KIO6) provide contextual grounding, while shared context memory preserves session state across interactions. While KIO10 is Energy-efficiency with Model-Driven TinyML and LLMs for Battery RUL prediction use-case. The engines operate under strict governance signals from Block 7, ensuring that generated artefacts satisfy security, privacy, and ethical criteria before release. Continuous self-improvement loops and Human-in-the-Loop oversight further enhance output quality, reliability, and trustworthiness.

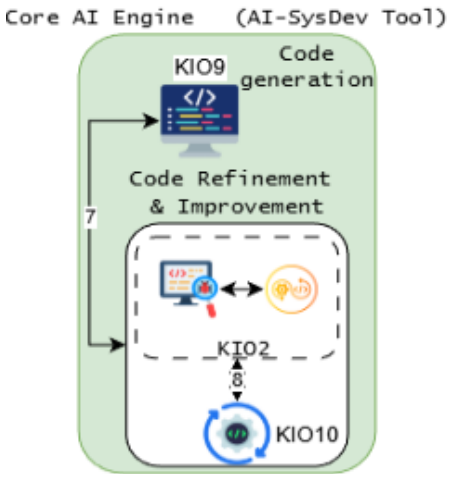


Figure 11: Core AI Engines

7.4.5 Building Block 5: Data Management Block

Block 5 (Figure 13) - provides secure, scalable, and governed data services underpinning the entire platform. It manages persistent storage (vector databases, knowledge graphs), real-time data ingestion from internal and external sources, and privacy-preserving retrieval mechanisms (KIO5, KIO6). All data flows are subject to anonymisation, encryption, and access control enforced by Block 7. The block supports versioning, provenance tracking, and caching strategies that accelerate RAG operations while maintaining auditability. By centralising data governance and ensuring compliance with GDPR and related instruments, Block 5 guarantees that AI components operate on trustworthy, high-quality, and legally compliant datasets throughout the lifecycle.

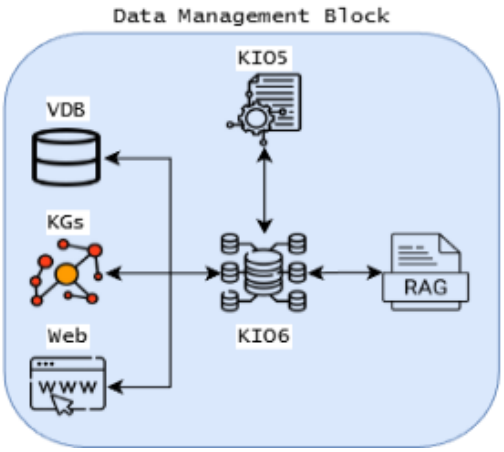


Figure 12: Data Management Block

7.4.6 Building Block 6: Test Automation and DevOps

Block 6 integrates the Testing Automation Tool (TAT), continuous compilation, and full DevOps pipelines (KIO8, KIO11) as shown in Figure 14. The block executes automated unit, integration, and system-level testing; performs multi-architecture cross-compilation; and manages continuous integration, delivery, and deployment workflows. Infrastructure-as-Code (IaC), container orchestration, and real-time monitoring are orchestrated in alignment with KIO7 directives. Test results and deployment artefacts are fed back into shared context memory, enabling closed-loop refinement. The block ensures rapid, reproducible, and verifiable releases while maintaining traceability from requirements (Block 3) to production.

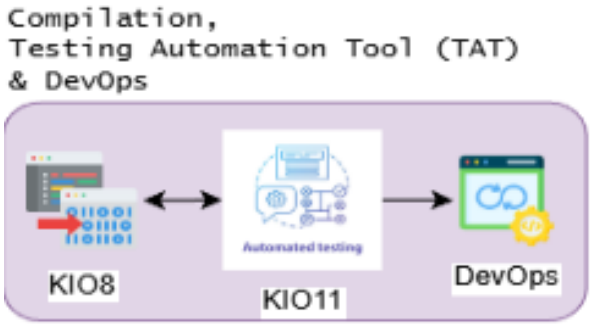


Figure 13: Test Automation and DevOps Block

7.4.7 Building Block 7: Security & Governance

As illustrated in Figure 15, Block 7 implements pervasive security, privacy, safety, and regulatory compliance functions (KIO12). It continuously enforces policies on fairness, transparency, robustness, and accountability across all blocks and artefacts. Real-time auditing, bias detection, explainability checks, and risk assessment are performed, with governance tokens accompanying every data and code movement. Non-compliant operations are blocked automatically. Integrated with KIO7, Block 7 inserts mandatory Human-in-the-Loop gates for high-risk scenarios and generates comprehensive compliance reports, ensuring full conformity with the EU AI-Act, GDPR, and project-specific ethical requirements.

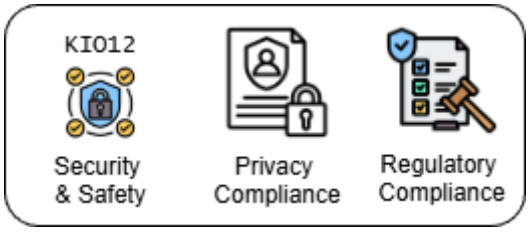


Figure 14: Security & Governance Block

7.5 Cross-Cutting Non-Functional Specifications

Having presented the seven building blocks that constitute the functional core of the AI-SysDev platform in Section 7.4, it is now appropriate to examine the cross-cutting non-functional requirements that govern their collective behaviour and ensure the system overall quality, trustworthiness, and operational viability.

These requirements are not confined to individual blocks but are realised through deliberate architectural decisions that span the entire platform, most notably the modular KIO design, the orchestration capabilities of Block 2, the continuous verification provided by Block 6, and the pervasive governance enforced by Block 7. They collectively

guarantee that the functional innovations delivered by the building blocks operate within a robust, secure, and maintainable framework, fully satisfying both project objectives and applicable regulatory standards.

The platform meets the following cross-cutting non-functional requirements:

- **Upgradeability** The modular KIO architecture permits independent updating of components (e.g., fine-tuning of large language models) without disrupting the overall pipeline, thereby ensuring sustained compatibility with evolving AI techniques.
- **Security** Components adhere to established security best practices, supported by regular audits, automated vulnerability mitigation tools, and mandatory Human-in-the-Loop controls.
- **Standardisation and Interoperability** Standardised RESTful APIs and well-defined interfaces enable seamless inter-KIO communication and integration with external systems.
- **Performance** Low-latency AI inferencing and parallel processing are achieved throughout the lifecycle.
- **Scalability** Containerised deployment supports horizontal scaling under varying workloads.
- **Reliability** Geo-redundancy and consensus mechanisms ensure high availability and resilience.
- **Consistency** Semantic stores and streaming tools maintain data and process integrity across all blocks.

Table 67 summarises the key non-functional requirements for the principal components, together with their dependencies and implementation approaches.

Table 67: Summary of Cross-Cutting Non-Functional Requirements

Component	Requirement	Dependencies	Implementation Approach
AI-SysDev Orchestration (KIO7)	Load balancing	Integration architecture	DevOps pipelines, queue management, shared context memory
AI-SysDev Orchestration (KIO7)	Scalability	Resource availability	Containerised deployment with auto-scaling
Testing Automation Tool (KIO11)	Reliability	Consensus protocols	AI-driven testing with fault-tolerant mechanisms and HITL validation
Testing Automation Tool (KIO11)	Consistency	Networking	Direct interconnections for redundancy
Core AI Engines (KIO2, KIO9, KIO10)	Response time	Load balancing	Optimised LLMs with shared context memory

Data Management (KIO5–KIO6)	Reliability	Underlying infrastructure	Geo-redundancy and provenance tracking with shared context memory
Security & Governance (KIO12)	Execution timing	Transaction queues	Callback management with notifications and HITL oversight
Requirements Engineering (KIO3–KIO4)	Consistency	Data flows	Semantic triple stores with HITL refinement

7.6 Agentic Architecture

The AI4SWEng framework adopts an agentic interaction paradigm in which autonomous AI agents collaborate to execute complex, multi-stage software engineering tasks. Each building block and its associated Key Innovation Outputs (KIOs) are instantiated as specialised agents capable of independent reasoning, planning, action, and reflection. Central coordination is performed by the AI-SysDev Orchestration component (KIO7), which functions as a meta-agent responsible for high-level goal decomposition, dynamic task assignment, iterative workflow management, and state persistence via shared context memory, with Human-in-the-Loop (HITL) intervention points for governance and oversight.

This agentic model is directly inspired by the multi-agent conversation framework presented in Wu Q et.al [13], which enables scalable orchestration of heterogeneous LLM-based agents through structured message passing and programmable conversation patterns.

The resulting workflow comprises ten clearly defined stages, each mapped to specific building blocks and KIOs, as summarised in Table 68. This staged approach supports emergent capabilities such as self-healing code, adaptive cross-architecture optimisation, and continuous improvement while preserving traceability and human-centric control.

Table 68: Stages of the Agentic Architecture

Stage	Purpose	Primary Building Block(s)	KIOs Involved	Notes
1. Understand / Parse	Interpret input & intent	3. Requirements Engineering	KIO3, KIO4	NLP → Formal spec
2. Plan	Create strategy / task breakdown	3. Requirements Engineering + 2. AI SysDev Orchestration	KIO4, KIO7, KIO8	Architecture planning + LLM orchestration
3. Retrieve	Get relevant data / docs	5. Data Pipeline	KIO6	Pull real data / docs
4. Augment / Ground	Merge context with prompt	5. Data Pipeline → 4. Core AI Engines	KIO5, KIO2	Synthetic data + grounding
5. Infer / Reason	Generate or deduce results	4. Core AI Engines + 2. AI SysDev Orchestration	KIO2, KIO9, KIO10, KIO11	Bug-fix, responsible code, energy opt

6. Act / Execute	Perform external actions	1. GUI & Execution + 2. AI SysDev Orchestration	KIO1, KIO7	Code view, cross-compile, run
7. Evaluate / Verify	Check correctness	4. Core AI Engines + 6. Security & Governance	KIO2, KIO12	AI testing + cyber feedback
8. Reflect / Revise	Improve through iteration	2. AI SysDev Orchestration + 6. Security & Governance	KIO8, KIO13	Central LLM revises; security loops back
9. Format / Compose	Prepare final output	1. GUI & Execution	KIO1	UI presents polished artifact
10. Learn / Adapt	Evolve with experience	2. AI SysDev Orchestration	KIO11	Feedback stored for future runs

Inter-agent communication occurs exclusively through standardised APIs orchestrated by KIO7, with shared context memory ensuring continuity across stages. HITL gates are systematically inserted at critical decision points (e.g., plan approval, verification failures, and ethical compliance checks), thereby maintaining human authority while exploiting the scalability and adaptability advantages of the multi-agent paradigm [13].

This agentic architecture directly supports the project objectives by enabling autonomous handling of dynamic, real-world use-cases - including iterative FPGA code refinement (Use-case 3) and privacy-preserving battery modelling (Use-case 4) - while preserving full traceability, security, and regulatory compliance.

8 Performance Verification and Certification Strategy

8.1 Introduction

This Chapter is to delineates the performance verification and certification-by-design dimensions and respective requirements to be met by the AI4SWEng platform. Any analysis is informed by the established certification methodologies including the CERTAIN Framework [14], NIST TEVV programmes [15], ISO/IEC 42001 [16], Z-Inspection® [11], IEEE CertifAIEd™, and allied frameworks[6] - as intrinsic elements of the development pipeline, thereby guaranteeing comprehensive traceability, audit preparedness, and readiness for conformity assessment.

The approach must distinguish rigorously between conventional machine learning components embedded within the platform and the agentic, large-language-model-orchestrated subsystems that define its core innovation (KIO7 AI-SysDev orchestration with retrieval-augmented generation, multi-tool calling, and autonomous refinement loops). These divergent system classes demand tailored verification pathways. Every functional and non-functional metric has therefore to be mapped consistently to relevant regulatory mandates under the EU AI-Act (particularly High-Risk Annex III requirements for software development tools), ISO/IEC standards, and other pertinent industrial schemes.

In accordance with Task T2.6, the primary objective of performance certification resides in the systematic evaluation of **efficiency**, **reliability**, and **scalability** of the AI-SysDev platform and Testing Automation Tool (TAT). Accordingly, verification protocols need to be consistently configured to furnish irrefutable evidence of robustness across these dimensions through benchmarking, stress testing, and iterative optimisation.

Accordingly, verification in AI4SWEng is required to include:

- Benchmarking of efficiency (code generation throughput, inference latency, energy-cost-aware token processing), reliability (task completion fidelity, fault-localisation precision, self-repair success rate), and scalability (concurrent developer sessions, multi-architecture compilation load).
- Stress testing for resilience against concept drift in codebases, adversarial prompt manipulation, resource-constrained edge deployment, and multi-agent coordination failures in DevOps workflows.

[6] https://standards.ieee.org/products-programs/icap/ieee-certifaied/

- Optimisation cycles leveraging KIO10 energy profiling, KIO8 cross-compilation feedback, and KIO2 reverse-execution traces to achieve sustained superiority in the designated pillars.
- Continuous RegOps-compatible monitoring that captures technical, behavioural, ethical, and operational indicators in real-time, with provenance-tracked lineage exported directly from the KIO7 orchestration layer.

It is required that all certification audit data be treated as "evidentiary artefacts" and as such are to be rendered immutable and automatically structured for EU AI-Act conformity assessment, ensuring that certification constitutes an inherent outcome of platform operation rather than a terminal exercise.

Subsequent sections provide an outline of the contemporary landscape of AI certification frameworks (Section 8.2) and specify the architectural requirements for their targeted implementation within AI4SWEng (Section 8.3).

8.2 Contemporary Landscape of AI Certification and Evaluation Frameworks Relevant to AI-Powered Software Engineering Tools

The certification of artificial intelligence systems deployed in software engineering contexts presents unique challenges: these tools operate as high-risk enablers under the EU AI-Act (Annex III, point 5(c) - systems intended to be used for development of software with safety components), exhibit agentic characteristics (autonomous code synthesis, debugging, refactoring, and testing), and must demonstrate verifiable gains in developer productivity whilst preserving reliability across heterogeneous architectures.

Three strata within the contemporary certification landscape are particularly relevant to AI4SWEng architectural requirements as follows:

1. **Conceptual and Research-Oriented Frameworks for AI-Assisted Development**
    1. The CERTAIN Framework supplies semantic MLOps and RegOps workflows with ontology-grounded lineage, directly applicable to tracing AI-generated code artefacts and ensuring EU AI-Act compliance in high-risk software engineering tools [14].
    2. The CSIRO Responsible AI Metrics Catalogue provides process/ resource/product metrics calibrated for generative systems, enabling quantification of accountability in code generation and test synthesis [17].
    3. The DFKI XAI Metrics Taxonomy delivers psychometrically validated explainability indicators essential for rendering LLM reasoning chains comprehensible to developers [18].

4. The Cambridge Paradigms of AI Evaluation advocates cross-paradigm assessment, vital for reconciling statistical performance metrics with behavioural reliability in agentic coding assistants [19].
5. The ZHAW CertAInty Scheme operationalises reliability, robustness, and transparency criteria, validated on industrial models and readily extensible to TinyML components in multi-architecture deployment [20].

2. **Normative Standards for Governance of Software Engineering AI Platforms**
    1. ISO/IEC 42001 mandates an Artificial Intelligence Management System with explicit controls for lifecycle risk treatment in software development tools [21].
    2. ISO/IEC 42006 defines audit protocols tailored to such systems. The IEEE CertifAIEd™ Joint Specification establishes graded certification across human oversight, safety, and fairness principles - directly pertinent to human-in-the-loop code review gates in AI4SWEng [16].
    3. The Cloud Security Alliance STAR for AI provides tiered assurance for containerised, cloud-native platforms such as the KIO-orchestrated micro-service architecture [22].
3. **Risk-Based and Adversarial Evaluation for Agentic Software Engineering Agents**
    1. NIST TEVV programmes furnish testbeds for bias, robustness, and interpretability in generative code models [15].
    2. Advanced red-teaming methodologies are indispensable for exposing emergent vulnerabilities in agentic workflows - such as hallucinated imports, unsafe refactoring cascades, or privilege escalation via tool misuse [23].
    3. The Z-Inspection® process ensures socio-technical ethical scrutiny of AI tools that directly influence critical software infrastructure [5].

AI4SWEng is to push forward the frontiers of innovation in Responsible-AI. It is thus required to embed the most salient components of all the above three certification strata as compulsory enforcement points within its orchestration (KIO7), responsible coding (KIO9), multi-level security (KIO12), and testing automation (KIO11) layers, thereby ensuring that certifiability is realised as an intrinsic platform capability alongside demonstrable excellence in efficiency, reliability, and scalability.

8.3 Certification Frameworks and Implementation Maturity in AI4SWEng

The verification and certification strategy of AI4SWEng rests upon selectively curated frameworks that collectively span the technical, governance, ethical, and agentic-specific requirements of an AI-powered software engineering platform.

The levels below specify the targeted implementation maturity for each framework consistent with the project certification-by-design guideline:

- **Level 1 – Specified**: Embedded in requirements, architecture, or orchestration design (D2.5).
- **Level 2 – Prototyped**: Functional in KIO prototypes or internal testbeds.
- **Level 3 – Integrated**: Fully operational in the core AI-SysDev + TAT pipeline with automated evidence collection.
- **Level 4 – Validated**: Demonstrated in pilot use-cases with quantitative KPI linkage (WP5–WP6).
- **Level 5 – Certified**: Externally audited and ready for conformity assessment.

The architectural design of AI4SWEng is required to ensure that predominant Level 3–4 maturity is attained across the certification frameworks upon completion of integration and internal validation. This is to be achieved by rendering each framework control as a mandatory execution path within the orchestration layer (KIO7), responsible-innovation gates (KIO9), cybersecurity safeguards (KIO12), and testing automation tool (KIO11). Full Level 5 maturity represents the ultimate objective, realised through large-scale pilot validation in WP5–WP6.

8.4 Benchmarking Activities

Benchmarking constitutes the foundational mechanism for establishing quantifiable baselines and comparative performance across the triad of efficiency, reliability, and scalability mandated by Task T2.6. These activities provide the empirical basis upon which the platform is to demonstrate that verifiable superiority over conventional software engineering tools is demonstrated.

Benchmarks are to be executed against a dual workload corpus: (i) synthetic workloads systematically derived from the five project use-cases (UC1–UC5), ensuring coverage of requirements engineering, architectural planning, code generation, debugging, testing, and multi-architecture deployment; and (ii) real-world repositories selected for diversity in language, scale, complexity, and domain (including open-source projects representative of enterprise, embedded, and safety-critical systems).

The benchmarking regime is required to incorporate benchmarking structured to explicitly audit the performance of each of the three performance pillars in Task T2.6 as follows:

- **Efficiency benchmarks** quantify tokens-per-second-per-watt under varying energy profiles, inference latency in resource-constrained environments, and end-to-end workflow duration for canonical software engineering tasks (natural-language requirements → formal specification → architecture → code → test → cross-compilation → deployment).
- **Reliability benchmarks** assess fault-localisation precision and recall (KIO2 reverse execution and dynamic slicing), self-repair success rate across defect classes, test generation coverage and flaw-detection efficacy (KIO11), and architectural suggestion consistency across iterative refinement cycles.
- **Scalability benchmarks** evaluate maximum concurrent developer sessions without performance degradation, cross-architecture compilation throughput under increasing target heterogeneity (KIO8), and resource elasticity ratios in containerised orchestration environments.

All benchmarking executions are required to be fully automated within the Testing Automation Tool (KIO11) and orchestrated via KIO7, with results ingested into the Regulatory Operations (RegOps)-compatible telemetry pipeline for immutable storage and direct contribution to conformity assessment evidence. This ensures that benchmarking serves not merely as measurement but as continuous, enforceable certification infrastructure.

8.5 Certification-by-Design and Standards Compliance Mapping for the AI4SWEng Platform

Certifiability constitutes an intrinsic, continuously enforceable property of the system architecture rather than a retrospective compliance exercise. For AI4SWEng platform to embody certification-by-design as a core capability, this requires the embedding of a comprehensive taxonomy of testability patterns as mandatory inspection points throughout the lifecycle, ensuring that every functional path is instrumented for detection, measurement, and reliable assessment of efficiency, reliability, and scalability.

The platform is to this to implement testability patterns differentiated for classic and agentic components. For classic components (for example, synthetic data generation KIO5/KIO6), the essential patterns comprise Accuracy Logging Hook, Explainability Interface, Bias Detection Hook, Drift Monitoring Point, and Privacy Audit API. For agentic components (KIO7 orchestration and multi-tool agents), the patterns extend to Goal

Alignment Validator, Reasoning Trace Logger, Emergent Behaviour Monitor, Ethical Drift Tracker, and Multi-Agent Reliability Probe.

These patterns are realised as non-optional gateways, telemetry hooks, and guardrails within the KIO7 orchestration engine, ensuring automatic evidence generation for conformity assessment. Collectively, these patterns must operationalise the requirements of the framework such as ISO/IEC 42001, IEEE CertifAIEd™, Z-Inspection®, and the EU AI-Act.

The patterns are to be systematically embedded across all lifecycle phases - data ingestion, model fine-tuning, deployment, and continuous operation - rendering verifiability an intrinsic property. Consequently, certification transcends mere compliance documentation to provide a routinised automated, and continuously auditable operational result of platform execution. This approach positions AI4SWEng as a reference implementation for high-risk AI systems in software engineering, wherein testability is elevated to a core quality attribute alongside efficiency, reliability, and scalability.

8.6 Stress Testing Strategy

Stress testing constitutes an essential protocol supporting the AI4SWEng verification regime, specifically designed to validate reliability - the central pillar of Task T2.6 - under conditions that are to faithfully replicate the exigencies of real-world software engineering practice. These conditions include prolonged multi-turn developer interactions, repository-scale codebases spanning thousands of lines across multiple languages and architectures, resource-constrained edge deployment scenarios (as in UC4 TinyML battery management), and concurrent multi-user DevOps workflows.

The stress testing strategy needs to be planned and structured so as to expose latent vulnerabilities that manifest only under operational duress, thereby furnishing irrefutable evidence of robustness for conformity assessment under EU AI-Act high-risk obligations. Testing regimes are to be executed automatically within the Testing Automation Tool (KIO11) and orchestrated via KIO7, with full provenance tracking and immutable logging to support RegOps workflows.

Stress Testing Regimes consist of five primary categories, each calibrated to the distinctive risk profile of AI-powered software engineering tools:

- **Load and Concurrency Stress** – evaluates platform behaviour under peak operational load, including simultaneous multi-developer sessions, parallel cross-compilation jobs (KIO8), and sustained high-frequency inference requests.
- **Adversarial Prompt and Code Injection Stress** – probes susceptibility to malicious or malformed inputs designed to induce hallucinated imports, unsafe

refactoring, privilege escalation via tool misuse, or generation of vulnerable code patterns.

- **Drift Stress** – simulates temporal evolution of codebases, shifting architectural constraints, and progressive requirement creep to assess degradation in architectural consistency, fault-localisation efficacy, and energy-profile stability.
- **Resource Starvation Stress** – replicates edge deployment constraints (memory ≤ 512 KB, power ≤ 100 mW) characteristic of TinyML targets in UC4, validating continued reliability when inference must migrate between cloud, edge, and microcontroller environments.
- **Agentic Conflict and Emergent Behaviour Stress** – induces scenarios featuring mutually contradictory requirements, unstable or evolving goals, cascading error propagation across multi-agent coordination, hallucinations in tool-use reasoning chains, and divergence between planned and executed actions.

Testing regimes are to align directly with NIST TEVV adversarial robustness protocols, CSIRO robustness catalogue criteria, CERTAIN semantic monitoring requirements, and state-of-the-art red-teaming practices adapted specifically to software engineering risks.

All stress testing executions are to trigger mandatory human-in-the-loop escalation when predefined severity thresholds are breached, ensuring that reliability under extreme conditions is not merely measured but actively certified through documented intervention and resolution traces. Results contribute directly to KPI evidence bundles for reliability (KPI 3.1, 3.2, 6.1, 6.2, 9.1, 9.2) and feed the continuous optimisation loops required to sustain superiority in the Task T2.6 triad of efficiency, reliability, and scalability.

8.7 Performance Optimisation

Performance optimisation cycles constitute the iterative refinement mechanism by which AI4SWEng is to sustain verifiable excellence across the Task T2.6 triad of efficiency, reliability, and scalability. These cycles are to be triggered automatically upon completion of benchmarking and stress testing phases, leveraging telemetry from KIO10 energy profiling, KIO8 cross-compilation feedback, KIO2 reverse-execution traces, and KIO11 test coverage analysis to identify precise improvement opportunities.

Optimisation is to operate systematically at four complementary levels:

- **Model-level optimisation** is to enhance individual LLM instances through quantisation, pruning, knowledge distillation, and adaptive precision selection, thereby reducing inference latency and energy consumption whilst preserving generative fidelity.

- **Pipeline-level optimisation** is to streamline end-to-end software engineering workflows via aggressive retrieval-augmented caching, intelligent batching of compilation and testing operations, and parallel execution across heterogeneous targets.
- **Agentic-level optimisation** is to ensure sustained coherence and efficiency during extended multi-step reasoning through context compression, memory-based reflection heuristics, speculative planning pruning, and dynamic tool-selection policies that minimise unnecessary external calls.
- **Resource-level optimisation** is to enforces cost-aware scheduling, dynamic scaling of containerised micro-services, and pre-emptive migration between cloud, edge, and micro-controller execution environments in accordance with real-time energy and latency constraints.

The combined effect of the optimisation interventions is to be monitored continuously against the project KPIs, with optimisation loops terminating only when predefined convergence criteria are satisfied or human-in-the-loop approval is granted for deployment. The optimisation categories are to operate under closed-loop control whereby every modification is to be immediately subjected to regression benchmarking and stress testing, with rollback triggered automatically should any regression exceed predefined tolerance thresholds. This is so as to ensure that efficiency gains and scalability never compromise reliability. In this way AI4SEWEng is to satisfy both Task T2.6 certification objectives and EU AI-Act requirements for continuous risk control in high-risk software development systems.

8.8 Monitoring and Measurement Framework

For compliance assurance by design, monitoring is regarded as an abiding system companion; an intrinsic, non-optional component of platform operation, ensuring that every engineering task contributes automatically to the evidentiary corpus required for certification The AI4SWEng platform is required to incorporate multi-dimensional monitoring capabilities for performance data acquisition on technical, behavioural, ethical, and operational agentic indicators in real-time. Thus, monitoring is required to be fully automated, continuous, and inextricably integrated with the semantic traceability of data flows (KIO7 orchestration layer), thereby generating immutable audit trails and provenance-tracked evidence expressly structured for Regulatory Operations workflows and EU AI-Act conformity assessment.

The framework has to be configured to deliver direct, quantifiable evidence across the three pillars mandated by Task T2.6:

- **Efficiency** –Energy consumption per engineering task (Wh/task), latency distributions (p50/p95/p99), and throughput under varying resource constraints.

- **Reliability** – Mean time between failures in agentic reasoning loops, self-repair success rates (KIO2), fault-localisation precision, and test coverage stability (KIO11).
- **Scalability** – Peak concurrent developer sessions, cross-architecture compilation elasticity, and resource scaling ratios under load.

Ethical and regulatory indicators (bias drift, demographic parity, explanation fidelity, ethical drift, hallucination frequency, and constraint adherence) are monitored with equal rigour to satisfy high-risk system obligations under the EU AI-Act.

All monitored parameters are to be ingested into a unified telemetry dashboard that supports immediate anomaly detection, automated alerting, and mandatory human-in-the-loop escalation when thresholds are breached. This will ensure that monitoring serves not merely as observation but as an active enforcement mechanism for sustained excellence in efficiency, reliability, and scalability.

8.9 Certification Pathways for Classic and Agentic AI

Certification requirements within AI4SWEng must necessarily differentiate between classic machine learning components and the agentic, LLM-orchestrated subsystems that constitute the targeted innovation of the project.

Classic components are exemplified by synthetic data generation and validation (KIO5/KIO6), data preparation pipelines, and isolated predictive modules. These follow established machine learning certification pathways. The emphasise here is on reproducible metrics, fairness quantification, robustness against drift, and comprehensive documentation in accordance with EU AI-Act Annex IV and ISO/IEC 42001 controls.

Agentic components are principally characterised by the orchestration component (the KIO7 engine in AI4SWEng) with multi-tool autonomous agents, dynamic reasoning chains, and self-refinement loops. These require substantially extended pathways that incorporate continuous behavioural evaluation, goal alignment persistence across long contexts, emergent behaviour containment, and guardrail enforcement during real-time operation. These pathways mandate ongoing red-teaming, reasoning trace auditing, and multi-agent conflict detection throughout deployment.

The platform is required to implement rigorously differentiated yet unified certification pathways whilst preserving single-source evidence collection to ensure seamless progression to full conformity assessment for high-risk software development tools under the EU AI-Act.

This bifurcated yet harmonised approach ensures that certification remains proportionate to risk whilst utilising a unified telemetry and lineage infrastructure to

deliver a single, auditable evidentiary corpus for the entire platform. The strategy thereby satisfies the most stringent interpretation of EU AI-Act high-risk obligations for software engineering tools whilst preserving the innovative advantages of agentic assistance.

9 Conclusions and Future Work

This brief coda concludes the deliverable D2.6.

9.1 Summary

This deliverable presents the results of the work performed on three elements of AI4SWEng in tasks T2.5 and T2.6. as follows:

1) Key Innovation Outcomes (KIOs): The AI4SWEng KIOs are the key organising mechanisms for the research and development activities in the project. This deliverable summarises the initial requirements specification activities as performed by the Consortium, describes the tasks and processes for revising and improving the initial requirements, presents the revised requirements and introduces the system that is in use for working with requirements and tasks in the project.
2) Use-Cases (UCs): This deliverable introduces the Use-Case requirements for the five use-cases of the project. It elaborates on the overall aims defined for the UCs by presenting detailed specifications for the use-cases, and it relates the UCs to the project KIOs and project KPIs to facilitate meaningful validation of the project KPIs through the project use-cases.
3) System architecture: The deliverable presents the initial system architecture for AI4SWEng, composed of a set of building blocks and components that create the overall AI4SWEng tool suite. Performance validation and certification criteria are defined that will be applied to measure the performance of the platform components.

This concludes the initial work on KIO and UC requirements in the project.

9.2 Upcoming Work and Deliverables

The work on the project requirements, use-cases and system architecture will continue after the completion of this document. The project requirements are “living documents” and will be continuously updated, extended and revised using the GitHub project infrastructure as deployed in this project.

The following deliverables will contribute updates to the analysis as presented in this document:

- Deliverable D2.2 “Organisation-Specific Onsite Requirements” (M18) will present updates on the building block architecture presented in this deliverable in particular regarding the integration using industry-standard protocols and specifications.
- Deliverable D2.3 “Needs Re-Elicitation and Governance Organisation” (M16) will provide a revision of the security, traceability and privacy-by-design requirements

from D2.1 with special focus on the AI-Act, including the most recent developments in the implementation of the Act.

- Deliverable D2.5 “Requirements and Report for Data Preparation Activities (final version)” (M28) will present the final update for the system requirements and use-case requirements presented in this deliverable.

Revisions and further specification of the system architecture beyond the level of building blocks will be incorporated as part of the technical development in WP3 and WP4. WP6 will further specify and elaborate the use-case and validation activities of the project.

References

[1] A. Badii, D. Fuschi, A. Khan and A. Adetoye, "Accessibility-by-Design: A Framework for Delivery-Context-Aware Personalised Media Content Re-purposing," in *Lecture Notes in Computer Science (LNPSE,volume 5889)*, Berlin, Heidelberg, Springer, 2009, pp. 209-226.

[2] B. L. Fredrickson and D. Kahneman, "Duration Neglect in Retrospective Evaluations of Affective Episodes," *Journal of Personality and Social Psychology,* vol. 65, no. 1, pp. 45-55, 1993.

[3] B. R. Newell and A. Bröder, "Cognitive processes, models and metaphors in decision research," *Judgment and Decision Making,* vol. 3, no. 3, pp. 195-204, 2008.

[4] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC,* European Commission, 2016.

[5] *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence,* 2024.

[6] *Regulation (EU) 2023/2854 of the European Parliament and of the Council of 13 December 2023 on harmonised rules on fair access to and use of data,* 2023.

[7] J. J. Tithi, "Assessing Trustworthiness of AI Systems Using Z-Inspection," in *IEEE Conference on Artificial Intelligence*, Santa Clara, California, USA, 2025.

[8] G. Inc., "About Projects," GitHub Inc., 2025. [Online]. Available: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects. [Accessed 07 11 2025].

[9] A. Badii, "User-intimate requirements hierarchy resolution framework (UI-REF)," in *AmI-08: Second European Conference on Ambient Intelligence*, Nuremberg, Germany., 2008.

[10] D. a. R. A. AI, "Internal Regulations," 17 December 2021. [Online]. Available: https://adr-association.eu/sites/default/files/2024-04/Internal-Regulations-17122021-0v8-CLEAN.pdf.

[11] R. V. Zicari, J. Brodersen, J. Brusseau, B. Dudder, T. Eichhorn, T. Ivanov, G. Kararigas, P. Kringen, M. McCullough, M. Moslein, N. Mushtaq, G. Roig, N. Sturtz, K. Tolle, J. J.

Tithi, I. van Halem and M. Westerlund, "Z-Inspection: A Process to Assess Trustworthy AI," *IEEE Transactions on Technology and Society,* vol. 1, no. 1, 2021.

[12] U.S. Department of Health and Human Services, "Summary of the HIPAA privacy rules," 14 March 2025. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

[13] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.08155.

[14] F. Kovac, S. Neumainer, T. Pahi, T. Priebe, R. Rodrigues, D. Christodoulou, M. Cordy, S. Kubler, A. Kordia, G. Pitsiladis, J. Soldatos and P. Zervoudakis, "Towards a Framework for Supporting the Ethical and Regulatory Certification of AI Systems," in *Proceedings of the Workshop on AI Certification, Fairness and Regulations, co-located with the Austrian Symposium on AI and Vision (AIRoV 2025)*, 2025.

[15] NIST, "AI Test, Evaluation, Validation and Verification (TEVV)," 2024. [Online]. Available: https://www.nist.gov/ai-test-evaluation-validation-and-verification-tevv.

[16] ISO/IEC, "ISO/IEC 42006:2025," ISO/IEC, 2025.

[17] B. Xia, Q. Lu, L. Zhu, S. U. Lee, Y. Liu and Z. Xing, "Towards a Responsible AI Metrics Catalogue: A Collection of Metrics for AI Accountability," 2024. [Online]. Available: https://doi.org/10.1145/3644815.3644959.

[18] A. Kadir, A. Mosavi and D. Sonntag, "Evaluation Metrics for XAI: A Review, Taxonomy, and Practical Applications," 2023. [Online]. Available: https://www.dfki.de/fileadmin/user_upload/import/14708_XAI_Evaluation_Metrics__Taxonomies__Concepts_and_Applications__INES_2023_-7.pdf.

[19] J. Burden, M. Tesic, L. Pacchiardi and J. Hernandez-Orallo, "Paradigms of AI Evaluation: Mapping goals, methodologies and culture," 2025. [Online]. Available: ttps://arxiv.org/html/2502.15620v1.

[20] Y. B. S. C. R. D. P. F. O. M. C. F.-G. L. Billeter, M. Reif, F.-P. Schilling, J. Weng, A. Iranfar and M. Repetto, "https://digitalcollection.zhaw.ch/bitstream/11475/30549/3/2024_Billeter-etal_Certification-scheme-for-AI-based-systems.pdf," in *ESREL 2024*, 2024.

[21] ISO/IEC, "ISO/IEC 42001:2023," ISO/IEC, 2023.

[22] Cloud Security Alliance, "Cloud Security Alliance Launches STAR for AI, Establishing the Global Framework for Responsible and Auditable Artificial Intelligence," 23 October 2025. [Online]. Available: https://cloudsecurityalliance.org/press-releases/2025/10/23/cloud-security-alliance-launches-star-for-ai-establishing-the-global-framework-for-responsible-and-auditable-artificial-intelligence.

[23] B. Bullwinkel, A. Minnich, S. Chawla, G. Lopez, M. Pouliot, W. Maxwell, J. de Gruyter, K. Pratt, S. Qi, N. Chikanov, R. Lutz, R. S. R. Dheekonda, B.-E. Jagdagdorj, E. Kim and J. Song, "Lessons From Red Teaming 100 Generative AI Products," 13 January 2025. [Online]. Available: https://arxiv.org/abs/2501.07238.