



MANOLO

CLOUD-EDGE EFFICIENT & TRUSTWORTHY AI

D1.3 MANOLO Architecture, and Benchmarking framework v.2

CeADAR

30/06/2025



Funded by
the European Union

Document Information

Issued by:	NUIDUCD-CeADAR (CeADAR)
Issue date:	30/06/2025
Due date:	30/06/2025
Work package leader:	NUIDUCD-CeADAR
Dissemination level:	PU - Public

Document History

Version	Date	Modifications made by
0.1	12/06/2025	First complete version of the document by NUIDUCD-CeADAR
0.2	27/06/2025	Inclusion of WP leaders & partners contribution in each component section
1.0	30/06/2025	Final version by NUIDUCD-CeADAR including reviewer's feedback

Authors and Contributors

Name(s)	Beneficiary
Ricardo Simon Carbajo, Cristian Bosch Serrano, Eric Arazo, Andrés L. Suárez-Cetrulo, Hristo Stoev	NUIDUCD-CeADAR
Stasinos Konstantopoulos, Natalia Koliou	NSCR-D
Sebastian Karl, Maen Mallah, Leonidas Richter	Fraunhofer
Eva Marin Tordera, Xavier Masip, Muhammad Imran	UPC
Carlos Sanchez, Francesco D'Andria, Alejandro Garcia	ATOS
Manolis Kelaidis, Panagiotis Zazos, Giorgos Levis	FDI
Mounir Bensalem, Admela Jukan	TUBS
Guillaume Charpiat	INRIA
Marco Rosa, Francesco Ferro	PAL Robotics
Luis Montesano, Maria Sierra	Bit&Brain
Pantelis Kaplanoglou	ARX.NET
Magnus Westerlund, Roberto V. Zicari, Elisabeth Hildt	ARCADA
George Spyridopoulos, Apostolos Tsolakis	Q-PLAN

In case you want any additional information, or you want to consult with the authors of this document, please send your inquiries to: ricardo.simoncarbajo@ucd.ie

Quality Reviewers

First Name	Beneficiary
Sebastian Karl	Fraunhofer
Marco Rosa	PAL Robotics

Disclaimer

Funded by the European Union under GA no. 101135782. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or CNECT. Neither the European Union nor the granting authority can be held responsible for them.

©MANOLO Consortium, 2025

Reproduction is authorised provided the author gives consent and the source is acknowledged.

Executive Summary

This deliverable outlines the second version of the architecture of the MANOLO framework, library and toolset which includes the Benchmarking framework. MANOLO is a Horizon Europe project which focusses on the development of a novel framework, algorithms and methodologies for Efficient and Trustworthy AI in the Cloud-Edge Continuum. The project will deliver a framework which can be utilised by AI practitioners to optimise their AI pipelines and models to execute more efficiently across the cloud-edge continuum, while adopting trustworthy AI principles by design and provide tools to help enforce such principles. This architecture has been created to fulfil the project's objectives, and be applicable to any kind of domain, while being validated via three use cases.

The architecture is designed to meet the overall project non-functional requirements identified in Section 3.3 which are then mapped into high-level functional requirements in Section 3.4 and subsequently fine-tuned into functional requirements for each main component of the architecture in Chapters 4-7.

The description of the MANOLO architecture follows the ISO/IEC/IEEE 42010:2022 standard¹ to ensure thorough coverage of all stakeholder concerns. This document presents the MANOLO architecture from several key views, covering: 1) a functional perspective, detailing the core functionalities provided to the users, 2) a logical view, that breaks down the system's structure into distinct, functional interconnected components, 3) an operational view, examining the system's operation and interactions between components and data exchange, and 4) an integration view, that focuses on how the system will be integrated and updated.

This document also aims to align all consortium partners on a unified set of principles for system design and vocabulary for the MANOLO architecture. This approach ensures a shared understanding of the MANOLO system's functionalities and a consistent style for system design and implementation across the consortium. This is particularly crucial as different partners work independently within various technical Work Packages throughout the system's implementation.

This document corresponds to Deliverable 1.3 which supersedes the first version of the MANOLO architecture provided in Deliverable 1.2. This document constitutes a solid approach to the MANOLO architecture; however it is expected to slightly evolve during the ongoing development and validation phases.

¹ ISO/IEC 42010:2022: "Systems and software engineering – Architecture description" (November 2022).

Table of Contents

EXECUTIVE SUMMARY	4
LIST OF FIGURES	7
LIST OF TABLES	7
LIST OF TERMS AND DEFINITIONS.....	8
1. INTRODUCTION	9
1.1 PURPOSE OF THE DOCUMENT.....	9
1.2 RELATION TO OTHER PROJECT WORK	9
1.3 STRUCTURE OF THE DOCUMENT.....	10
2. METHODOLOGY	11
2.1 SYSTEM ARCHITECTURE CO-DEFINITION	11
2.2 TRUSTWORTHY AI CO-DESIGN (ARCADA).....	12
3. MANOLO ARCHITECTURAL OVERVIEW (CEADAR).....	14
3.1 MANOLO INTRODUCTION	14
3.2 FUNCTIONAL ARCHITECTURE OVERVIEW	16
3.3 HIGH-LEVEL NON-FUNCTIONAL REQUIREMENTS.....	18
3.4 HIGH-LEVEL FUNCTIONAL REQUIREMENTS.....	20
3.5 FRAMEWORK OVERVIEW	21
3.6 MANOLO LIBRARY.....	21
3.6.1 MANOLO Class.....	21
3.6.2 MANOLO Command Line Interface	22
3.6.3 MANOLO Base Module	22
3.6.4 MANOLO Core Module.....	24
3.6.5 MANOLO Extended modules.....	25
3.7 MANOLO SUITE AND SYSTEM ARCHITECTURE	25
3.7.1 MANOLO APIs	28
3.7.2 User Management & AI Workload Orchestrator	28
3.7.3 Data Management and Monitoring Tools (Data-Tier).....	29
3.7.4 Benchmarking Framework	30
3.7.5 MANOLO online catalogue (optional).....	31
3.8 MANOLO SYSTEM USAGE SCENARIOS	31
3.8.1 MANOLO Library Usage Scenario.....	31
3.8.2 MANOLO Suite Usage Scenario.....	32
3.8.3 MANOLO Hardware Aware Neural Architecture Search and Neuromorphic Hardware Usage Scenario	33
3.8.4 MANOLO Cloud-Edge Continuum Allocation Usage Scenario.....	34
3.8.5 MANOLO Object Manipulation Usage Scenario (PAL Robotics Use Case)	35
3.8.6 MANOLO Sleep Stage Prediction Usage Scenario (Bit&Brain Use Case).....	35
3.8.7 MANOLO Mobile Image Classification Usage Scenario (ARX.Net Use Case)	37
4. DATA INSPECTION & GENERATION (NCSR)	39
4.1 COMPONENT DESCRIPTION AND INTERACTIONS.....	39
4.1.1 Data Management and Provenance Framework.....	40
4.1.2 Data quality estimation and augmentation.....	41
4.1.3 Data distillation and synthetic feature representation.....	42

4.2	FUNCTIONAL REQUIREMENTS	42
4.3	TECHNOLOGIES	43
4.4	DEVELOPMENT CYCLE	44
5.	HW-AWARE MODEL TRAINING AND OPTIMIZATION (FRAUNHOFER)	45
5.1	COMPONENT DESCRIPTION AND INTERACTIONS.....	45
5.1.1	<i>Compression: Pruning, Quantization and Knowledge Distillation.....</i>	<i>46</i>
5.1.2	<i>Domain Adaptive Few-shot Continuous Learning</i>	<i>47</i>
5.1.3	<i>Neural architecture search.....</i>	<i>48</i>
5.1.4	<i>Frugal-oriented neural architecture growth</i>	<i>48</i>
5.1.5	<i>Neuromorphic computing</i>	<i>48</i>
5.2	FUNCTIONAL REQUIREMENTS	49
5.3	TECHNOLOGIES	51
5.4	DEVELOPMENT CYCLE	53
6.	CLOUD-EDGE CONTINUUM ML MODEL FUNCTION ALLOCATION (UPC)	54
6.1	COMPONENT DESCRIPTION AND INTERACTIONS.....	54
6.1.1	<i>Federated learning.....</i>	<i>56</i>
6.1.2	<i>Cloud-edge Resource and Infrastructure Mapping</i>	<i>58</i>
6.1.3	<i>Trustworthy-driven Policy Validation</i>	<i>58</i>
6.1.4	<i>Trustworthy & Infrastructure Oriented Model and Learning Allocation</i>	<i>60</i>
6.1.5	<i>Workflow of the Cloud-Edge Continuum ML Model Function Allocation.....</i>	<i>61</i>
6.2	FUNCTIONAL REQUIREMENTS	62
6.3	TECHNOLOGIES	62
6.4	DEVELOPMENT CYCLE	63
7.	TRUSTWORTHY EFFICIENCY AND PERFORMANCE BENCHMARKING (ATOS)	64
7.1	COMPONENT DESCRIPTION AND INTERACTIONS.....	64
7.1.1	<i>Benchmarking Framework Platform</i>	<i>65</i>
7.1.2	<i>Energy Efficiency and Model Performance Monitoring and Analysis</i>	<i>66</i>
7.1.3	<i>Explainability and Robustness Validation</i>	<i>68</i>
7.2	FUNCTIONAL REQUIREMENTS	68
7.3	TECHNOLOGIES	69
7.4	DEVELOPMENT CYCLE	70
8.	DEVELOPMENT PROCESS, INTEGRATION, AND VALIDATION ENVIRONMENT (CEADAR & UPC)	71
8.1	DEVELOPMENT AND INTEGRATION TECHNOLOGIES.....	71
8.2	DEVELOPMENT CYCLE AND INTEGRATION PROCESS	71
8.3	TESTING AND VALIDATION ENVIRONMENTS	72
8.4	LICENSING.....	73
9.	CONCLUSIONS	74
APPENDICES	76	
APPENDIX 1: Z-INSPECTION® TRUSTWORTHY AI CO-DESIGN PROCESS	76	

List of Figures

Figure 1. Full MANOLO Suite usage scenario.....	15
Figure 2. Reduced MANOLO Suite usage scenario.....	15
Figure 3. MANOLO Library usage scenario.....	16
Figure 4. The MANOLO project Conceptual Architecture.....	16
Figure 5. MANOLO command line interface	22
Figure 6. The MANOLO base module provides the building blocks for the rest of the modules to ensure compatibility. ..	23
Figure 7. The data building block is split into two big components; the first addresses data quality, and the second further processes the data to increase variety, extract relevant features, etc.	24
Figure 8: MANOLO High-Level Network Architecture - MANOLO Suite Modules (Controller & Node).....	26
Figure 9: MANOLO System Architecture - MANOLO Suite Modules (Controller & Node).....	27
Figure 10. System Use Case (model compression) for the MANOLO library.....	32
Figure 11. System Use Case (NAS) for the MANOLO library.....	32
Figure 12. System Use Case for the MANOLO suite.....	33
Figure 13. Data Inspection and Generation high-level architecture (highlighted in red).....	40
Figure 14. HW-aware Model Training and Optimisation high-level architecture (highlighted in red).	46
Figure 15. The compression component: it leverages the three most widely used compression techniques: knowledge distillation, quantization, and pruning.....	47
Figure 16. The meta-learning module: it provides the tools to maintain model adaptation to the target task.	48
Figure 17. Cloud-Edge Continuum AI Model Function Allocation high-level architecture (highlighted in red).....	55
Figure 18. Cloud-Edge Continuum AI Model Function Allocation: Components interactions	56
Figure 19. Federated learning functionality sequence diagram in MANOLO.	57
Figure 20. Cloud-edge Resource and Infrastructure Mapping component: Workflow.	58
Figure 21. Trustworthy-driven Policy Validation Sequence Diagram.....	59
Figure 22. Trustworthy and infrastructure-oriented model and learning allocation component interactions.....	60
Figure 23. Trustworthy & Infrastructure Oriented Model and Learning Allocation component: Workflow	61
Figure 24. Cloud-Edge Continuum ML Model Function Allocation: Workflow.	61
Figure 25. Trustworthy efficiency and performance benchmarking framework architecture (highlighted in red).....	64
Figure 26. Benchmarking Framework: Main components and interactions (based on Figure 9)	65
Figure 27. Energy Efficiency and Model Performance Monitoring and Analysis.	67
Figure 28. MANOLO integration process.	72
Figure 30. MANOLO Template steps for creating socio-technical scenarios.	76
Figure 30. Trustworthy AI system Co-design.....	78

List of Tables

Table 1. Terms and Definitions.	8
Table 2. TAI HLEG Requirements by the EU Ethics Guidelines for Trustworthy AI.	18
Table 3. High-level MANOLO non-functional requirements.....	19
Table 4. High-level MANOLO functional requirements.....	20
Table 5. Data Inspection & Generation: Components, associated tasks, responsible partner and planned integration....	39
Table 6. Data Inspection & Generation: Functional Requirements.....	43
Table 7. Initial mapping of proposed technologies for Data Inspection & Generation building block.....	43
Table 8. Initial mapping of baseline datasets to support early Data Inspection & Generation development	44
Table 9. Model Training & Optimisation: Components, associated tasks, responsible partner and planned integration. .	45
Table 10. HW-aware Model Training and Optimisation: Functional requirements.	51
Table 11. HW-aware Model Training and Optimisation: Initial mapping of potential baseline datasets and models in WP3 to support the early development of this building block.	52
Table 12. HW-aware Model Training and Optimisation: Initial mapping of proposed technologies.	53

<i>Table 13. Cloud-Edge Continuum ML Model Function Allocation: Components, associated tasks, responsible partner and planned integration.</i>	<i>54</i>
<i>Table 14. Cloud-Edge Continuum ML Model Function Allocation: Functional requirements.....</i>	<i>62</i>
<i>Table 15. Initial mapping of proposed technologies.</i>	<i>62</i>
<i>Table 16. Trustworthy Efficiency & Performance Benchmarking framework: Components, associated tasks, responsible partner and planned integration</i>	<i>64</i>
<i>Table 17. Trustworthy efficiency and performance benchmarking framework: Functional requirements.....</i>	<i>69</i>
<i>Table 18. Trustworthy efficiency and performance benchmark: Initial mapping of technologies per components.</i>	<i>69</i>
<i>Table 19. Mapping of technologies for overall orchestration and integration of MANOLO framework.</i>	<i>71</i>

List of Terms and Definitions

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CEC	Cloud-Edge Continuum
DA	Domain Adaptation
FLOPS	Floating-Point Operations per Second
KD	Knowledge Distillation
NAS	Neural Architecture Search
NAG	Neural Architecture Growth
NSL	N-Shot Learning
SNN	Spiking Neural Network
YAML	Yet Another Markup Language

Table 1. Terms and Definitions.

1. Introduction

1.1 Purpose of the document

This document presents deliverable 1.3, and update of the MANOLO architecture presented in D1.2, containing the outcome of the collaborative effort performed as part of Work Package 1 in the development of the MANOLO Architecture (T1.4) and its associated Benchmarking framework (T1.3).

This document outlines the core structural elements of the MANOLO architecture. It details their functions, characteristics, and how they interact to fulfil all system requirements. The architecture is presented from four perspectives:

- **Logical view, or MANOLO overview:** illustrates the MANOLO toolset composition, breaking it down into components with specific functions and connections.
- **Functionality view:** focuses on the services provided to the user.
- **Operational view:** examines the system's dynamic operation, including component interactions, synchronisation methods, and data exchange.
- **Implementation view:** provides insights into the software development and integration of the components.

This architectural overview adheres to the structure and vocabulary outlined in the ISO/IEC/IEEE 42010:2022 standard for architectural descriptions. Furthermore, the use of standardised views and modelling languages, such as UML sequence diagrams, ensures clarity and facilitates efficient communication of concepts both within and outside the consortium.

This document is strategic as it is the result of ongoing discussions and collaborative effort to foster a shared understanding of the MANOLO system's capabilities, goals, and development approaches. A unified terminology and methodology have been established, facilitating efficient communication and cooperation among partners for the development and integration of each component.

1.2 Relation to other project work

This document outlines the second version of the architectural design for MANOLO. It presents a comprehensive set of architectural perspectives essential for coordinating the design, development, and validation of MANOLO. The primary target audience includes consortium members involved in component development, trustworthy assessment, integration and testing, and use-case implementation.

Additionally, this document could be valuable to external stakeholders in adjacent areas where model efficiency and trustworthiness are a must, such as researchers engaged in Horizon Europe projects that may leverage or expand upon MANOLO outcomes.

1.3 Structure of the document

This document is structured into nine main sections and supporting Appendix:

1. **Introduction:** This section is outlining the purpose of the document and relation to other project work.
 2. **Methodology:** This section explains the core principles and methodology followed for the definition of the system's architecture and the incorporation of trustworthiness by design in MANOLO.
 3. **MANOLO Architectural Overview:** The functionality and MANOLO building blocks and components are introduced and briefly described in this section along with the high-level non-functional and functional requirements of the system, as well as the system and network architecture along with proposed technologies. Note that each building block and component functionality is described in more depth in Sections 4-7. This section also details various user-centred usage scenarios of the MANOLO framework, highlighting core functionalities and actions users can perform.
 4. **Data Inspection and Generation:** This section describes the Data Inspection and Generation building block and its components within MANOLO and outlines its corresponding functional requirements, technologies, metrics, and development cycle.
 5. **HW-aware Model Training and Optimisation:** This section details the HW-aware Model Training and Optimization building block and its component within MANOLO. An overview is given of the components, and its interactions with other components are described, along with functional requirements, technologies, metrics, and development cycle.
 6. **Cloud-Edge Continuum ML Model Function Allocation:** This section describes the building block and its components which aggregates different functionality to assist in effective and efficient allocation of AI activities and resources in the Cloud-Edge Continuum. It describes the component, requirements, technologies and metrics, and development cycle.
 7. **Trustworthy Efficiency and Performance Benchmarking:** This section describes the Trustworthy Efficiency and Performance Benchmarking building block / framework of MANOLO. Its interactions with other building blocks and components are detailed, along with requirements, technologies and metrics, and the development cycle.
 8. **Development Process, Integration, and Validation Environment:** The development plans and the integration of the MANOLO framework are outlined in this section.
 9. **Conclusions:** This section concludes the MANOLO architecture document, presenting a final summary of the collaborative development methodology, the core functionality, and the foundation for future work and refinement.
- A. Appendix 1:** Z-Inspection® Trustworthy AI Co-design process is presented in more detail

2. Methodology

The primary goal of the architectural design phase in Work Package 1 (WP1) is to create a comprehensive, high-level view of the MANOLO toolset while addressing key user requirements on efficiency and trustworthiness.

2.1 System Architecture Co-definition

This architecture describes the MANOLO framework and toolset as a whole, focusing on its overall capabilities and interactions with users, as well as their building blocks and associated components and potential external libraries and tools. Details about the system, flow of information and technologies are provided and a technical overview of the key components of the MANOLO framework described in Sections 4-7 (which maps to technical WP2-5) is provided and further expanded in their corresponding deliverables submitted until M18.

Based on a thorough analysis of the key components, the consortium collaboratively developed an initial architecture for the MANOLO system for D1.2. This process involved a series of virtual and in-person meetings, brainstorming sessions, and model refinement iterations. This was performed in multiple co-design workshops and mainly presented and refined in the following meetings: i) in Barcelona at the 2nd MANOLO consortium meeting on 3-4 July 2024, ii) virtually on 4 September 2024, and iii) in Nuremberg at the 3rd MANOLO consortium meeting on 3-4 December 2024.

Initially, different architectural approaches were considered and evaluated throughout this process. The consortium assessed these options against a set of criteria derived from the project requirements, captured in this deliverable, and quality standards (ISO/IEC 25010²). The requirements for the MANOLO project were identified, and outcomes were documented here through a comprehensive analysis involving all key stakeholders and participants across the project's value chain.

To identify and prioritise requirements, the MoSCoW method was employed. This approach categorises requirements into four levels of importance:

- **Must have:** Essential for MANOLO core functionality and compliance.
- **Should have:** High-value features that enhance the product but are not strictly necessary.
- **Could have:** Desirable but non-essential features.
- **Will not have:** Features deferred to future iterations.

By applying this method to each requirement category, we focused on the "must-have" and "should-have" requirements, creating a manageable and prioritised list. Lower-priority items were excluded to maintain project focus.

The naming convention used for labelling the requirements in each section is as follows:

² ISO/IEC 25010:2011: "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models" (November 2011).

1. Functional and Non-Functional Requirement: Denotes whether the requirement is functional or non-functional (**FR**_XX_XX refers to functional requirements, **NFR**XX refers to a non-functional requirement).
2. Building Block: Denotes the building block associated with the functional requirement linking it to the correspondent section of the document (e.g., **FR_HL_XX** refers to a High-Level functional requirement whereas **FR_C4_XX** refers to specific building block, in this case the Data Inspection and Generation covered in Section 4) which may also be denoted as **C4_DATA**.
3. Requirement Number: The unique identifier for the requirement (e.g., **FR_CX_01**, **FR_CX_02**, **FR_HL_01**, **NFR01** etc.).

The initial evaluation criterion for the proposed solutions was their alignment with the specified functional requirements. Subsequently, the alternatives were assessed based on their ability to meet non-functional quality attributes. This was crucial to ensure that the final MANOLO toolset exhibited the desired performance characteristics. Key quality factors ensure the following attributes: model efficiency, robustness and trustworthiness, scalability of the toolset, resource availability in the cloud-edge continuum, trust and security of the data transfers and storage, and privacy (among others).

Both High-Level Non-functional and Functional requirements are further elaborated in the MANOLO Architectural Overview defined in Section 3. The elicitation of these requirements has been performed in three steps: i) through the activities of the project tasks T1.1 and T1.2, which focused on the user requirements with a focus on high-level efficiency and trustworthiness (as reported in D1.1); ii) the core technical system requirements, through the activities of T1.3 and T1.4; and iii) through the design of the use cases (use case requirements) under the activities of T6.1. The 3rd step (i.e., use cases requirements) completed and documented in deliverable D6.1 by M12.

Subsequent monthly meetings have occurred to align on the next version of the architecture and benchmarking framework. A dedicated workshop to agree on strategic details of the second version of the architecture took place in Athens as part of our 4th MANOLO consortium meeting on 13-14 May 2025. The resulting architectural diagrams, designs and set of technologies are presented in this document as version 2.0 of the MANOLO architecture.

2.2 Trustworthy AI Co-design (ARCADA)

To ensure that the design of the MANOLO AI system and its outcomes will be trustworthy, MANOLO uses the Z-Inspection® process (see Appendix 1) to co-design (T1.2) and iteratively monitor and assess (T5.5) its AI components, the overall system (or in between pipelines), and its application to the envisioned use cases, where the AI components are considered in the context of each application domain.

To achieve this, dedicated (open) teams of experts were formed within the consortium to follow the formalised process introduced by Z-Inspection® to ensure that each element of the MANOLO suite has been designed and will operate in a transparent and ethical manner.

In particular, for each MANOLO element, socio-technical usage scenarios have been created as a participatory design tool for achieving a trustworthy AI design and implementation. Socio-technical usage scenarios are also a useful tool to describe the aim of the system, the actors, their expectations, the goals of

actors' actions, the technology, and the context while consequentially fostering moral imagination and providing a common ground where experts from different fields can come together. In MANOLO, we use socio-technical scenarios within discussion workshops, where teams of experts (ethics, social, legal, and technology) work together to systematically examine and elaborate the various tasks with respect to different contexts of AI under consideration. Socio-technical scenarios can also be used to broaden stakeholders' understanding of one's own role in the technology, as well as awareness of stakeholders' interdependence.

Building on the TAI co-design activities performed under T1.2 (which continue under T1.3 and T5.5), the MANOLO consortium established a shared understanding of trustworthiness while identifying key areas of focus in the design of its components, guided by the seven AI HLEG general requirements, with particular attention to the cloud-edge continuum (CEC) context. Insights (see Table 18 in D1.1) from these activities and the socio-technical scenarios have guided the elicitation of non-functional requirements, which are presented in Section 3 (and also transformed and mapped into functional requirements per component in subsequent sections). These requirements could be redefined for the next version and will later be followed as part of the validation process for the project's success in implementing a trustworthy AI design.

At this point, it is important to highlight that the CEC environment presents unique challenges in ensuring trustworthiness, which go beyond applications at the cloud or the edge. MANOLO aims to address these by presenting an architecture that can effectively function across the cloud-edge continuum.

3. MANOLO Architectural Overview (CeADAR)

The second version of the overall MANOLO framework architecture to fulfil the MANOLO concept will be explained in this section. It contains the concept, system and network architectural diagrams and explain the development of the MANOLO library as a core set of algorithms and primitives for efficient and trustworthy ML pipeline development, and the MANOLO suite, as a set of tools to support the library for operation in cloud-edge environments which includes our benchmarking framework.

We will identify and define the MANOLO building blocks, its components and modules, outlining their interdependencies and usage scenarios; further details about functionality of these components will be provided in subsequent Sections 4-7, while Section 8 will discuss how the system will be integrated, deployed and validated.

Breaking down the MANOLO architecture according to the logical decomposition scheme is particularly useful for describing it in a detailed manner. It simplifies MANOLO by dividing it into manageable, distinct building blocks and components. This approach optimizes design, implementation, and testing efforts. Moreover, it facilitates the identification and consolidation of common functionalities and patterns across the system.

The next subsections describe MANOLO at different levels of abstraction. First, Section 3.1 “MANOLO Introduction”, presents the project from the broadest and highest perspective level and outlines its basic functionalities. Second, Section 3.2 “Functional Architecture Overview”, describes how MANOLO is divided into different building blocks and components that interact among them. Third, Section 3.3 “High-level Non-Functional Requirements”, introduces the basic non-functional requirements of MANOLO, raising concepts that permeate through all components. Then, Section 3.4 lists and describes the high-level functional requirements for the MANOLO software. After that, Section 3.5 gives an overview of the whole framework, and subsequently, Sections 3.6 and 3.7 provide descriptions of the “MANOLO library” and “MANOLO suite” including System and Network architectural diagrams to understand the deployment of MANOLO and the key modules of the Suite, their interconnections, flow of information, key functionality and technologies. Finally, a set of usage scenarios are presented in Section 3.8, which will enlighten the reader about different MANOLO pipelines and components involved in them both from a system usage perspective and from the perspective of the three use cases in the MANOLO project.

3.1 MANOLO Introduction

MANOLO is a framework that comprises multiple aggregated tools to assist AI practitioners in the development and deployment of efficiency and trustworthy AI solutions, providing a high degree of adaptability to accommodate the needs and resources of different users. At its core, MANOLO provides a library for efficient and trustworthy operations and also a suite which aggregates multiple connected tools, along with the benchmarking framework, for operation in the cloud-edge continuum according to the need and requirements of the user and infrastructure.

To better understand MANOLO, this subsection starts by presenting three levels of usability of the MANOLO framework that illustrate and exemplify the interaction between users and MANOLO. Note that Section 3.7 “MANOLO Suite”, provides further details and illustrations of the application of MANOLO.

The most complex configuration for using the MANOLO framework is depicted in Figure 1, which offers the full capabilities integrated in MANOLO as a suite for installation in a server or edge devices; this suite will be accessed through the MANOLO API, allowing different users to interact with all the functionalities from the

framework and using them to exploit the efficient pipeline tools tailored to the specific needs of each user's case, in a trustworthy environment.

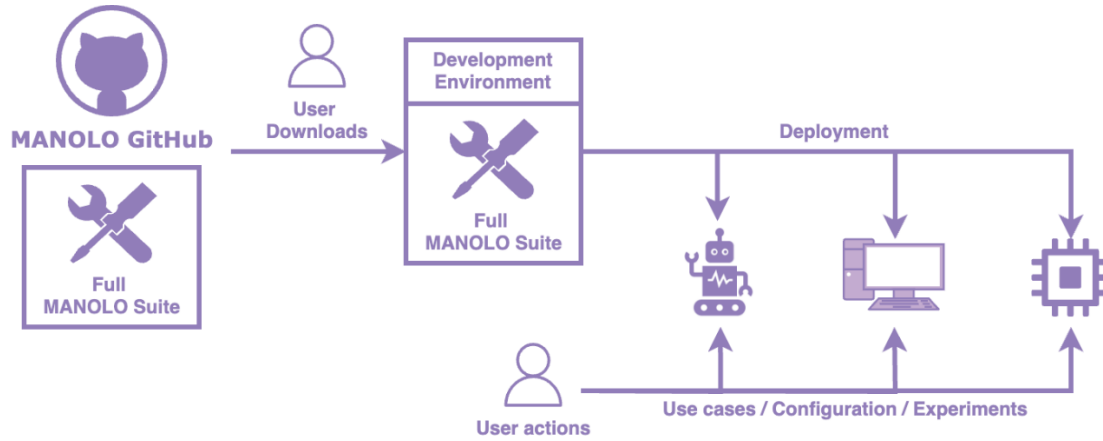


Figure 1. Full MANOLO Suite usage scenario.

Figure 2 presents the second usage option of the MANOLO framework, where the user directly interacts with the MANOLO repository. In this scenario, the user only downloads the elements from MANOLO strictly necessary based on the requirements and specifications for a specific use case. This could be directly deployed into an edge device or downloaded to a server and used as a Reduced MANOLO suite.

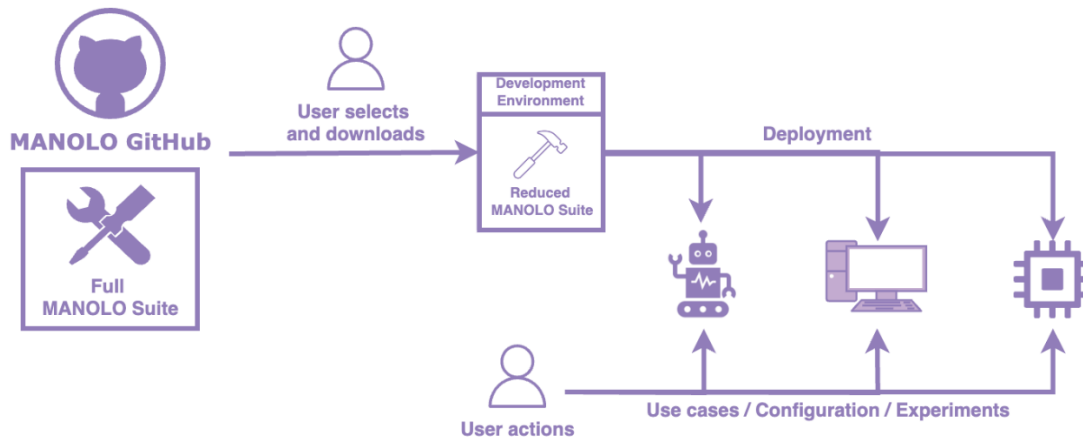


Figure 2. Reduced MANOLO Suite usage scenario.

Figure 3 presents the third usage option of the MANOLO framework, where the user directly interacts with just one of the main elements of the MANOLO framework, the MANOLO library. This contains the main ML functionalities of the framework without the additional overhead of system/suite functionalities. In this option, the user downloads the library into their development environment and uses it as a toolset to address a particular application, in this case deploying a trained model to an edge device or server.

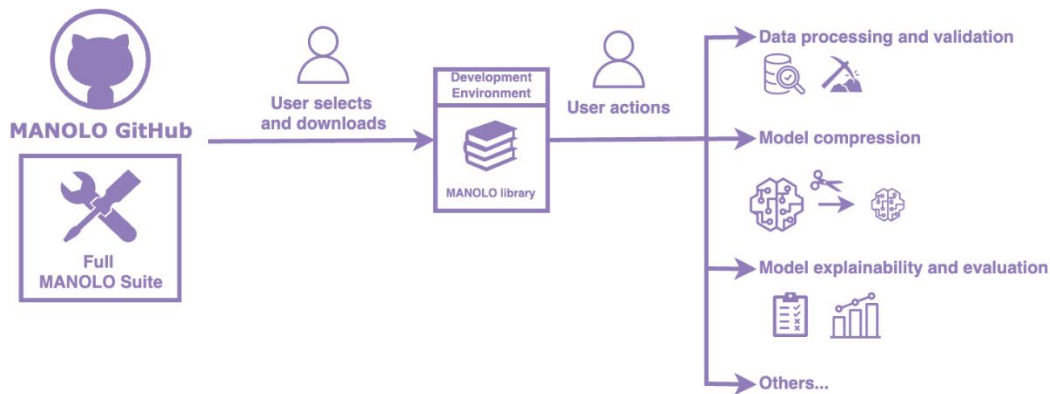


Figure 3. MANOLO Library usage scenario.

3.2 Functional Architecture Overview

MANOLO is divided into several technical building blocks (see AI Engine components in Figure 4).

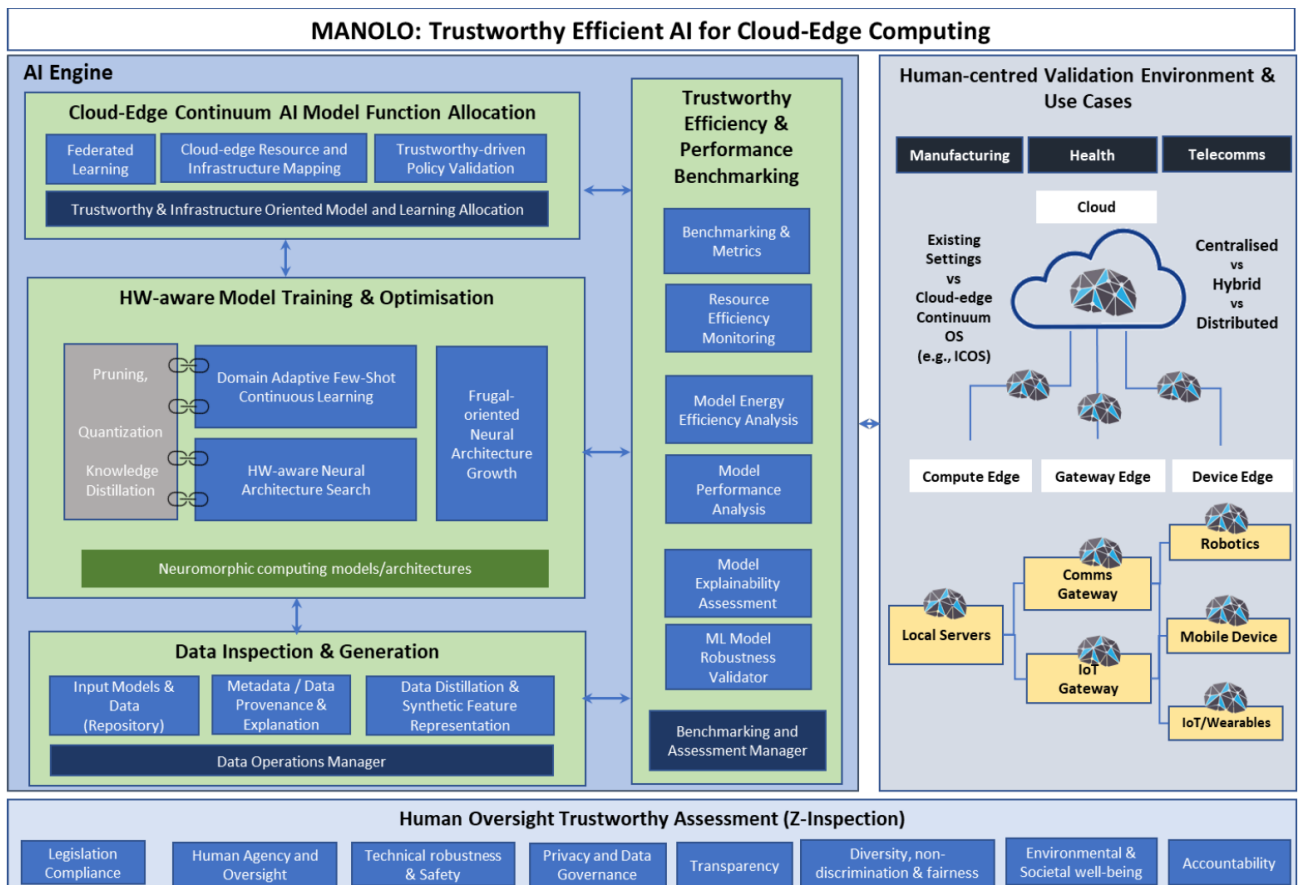


Figure 4. The MANOLO project Conceptual Architecture.

The ***Data Inspection & Generation*** technical building block (covered in detail in Section 4) deals with the collection of data, models, and context information to create a suitable environment for AI model development. This block also provides data provenance to the pipeline and provides tools to extract, compress, and synthesise datasets for efficient algorithm development within the MANOLO suite.

The ***HW-aware Model Training and Optimisation*** technical building block (covered in detail in Section 5) comprises algorithms that optimise models for computationally efficient deployment in the cloud-edge continuum. These algorithms are utilised to provide HW-specific optimised ML models and are grouped into the following four components: 1) compression with pruning, quantisation, and knowledge distillation, 2) domain adaptive few-shot continuous learning and model compression meta-learning, 3) HW-aware neural architecture search, 4) frugal-oriented neural architecture growth, and 5) Neuromorphic computing.

The ***Cloud-Edge Continuum ML Model Function Allocation*** technical building block (covered in detail in Section 6) consists of four components that aim to provide strategies to allocate resources according to specific AI learning and training needs. First, the federated learning component provides functionality for distributed training. Second, the cloud-edge resource and infrastructure mapping component will collect, organise, and map information about network structure and resource capacities. Third, the trustworthy and infrastructure-oriented model and learning allocation component will design a strategy to allocate resources in the continuum according to model specifications. Finally, the trustworthy-driven policy validation component will validate the outcomes of the previous component to guarantee learning and compliance with trustworthiness policies.

The ***Trustworthy Efficiency and Performance Benchmarking*** technical building block (covered in detail in Section 7) focuses on the development of a HW-aware cloud-edge-oriented benchmarking system specific to evaluating AI systems (pipelines, data, models, algorithms) in terms of efficiency, performance, and trustworthiness. It integrates components for modelling and evaluating performance and energy efficiency for different HW/resource environments being monitored. Specific modules for assessment and validation of explainability and robustness of ML models exposed to multiple datasets generated as part of the benchmarking experiments controlled in the overall manager module.

The ***Integration and Validation Environment*** building block (covered in Section 8) integrates and evaluates the full potential of the MANOLO framework. The library and suite will be integrated, deployed and tested in cloud-edge infrastructure/settings, and different hardware platforms and environments available within some of the partners premises. These environments will allow for further benchmarking MANOLO across existing and future conditions.

The ***Human Oversight Trustworthy Assessment*** building block (see Section 2.2) offers the processes and tools for co-designing, monitoring, assessing, and ensuring that the AI system developed will adhere to ethical and legal principles, while meeting societal expectations, following a Trustworthy AI co-design based on the Z-Inspection® process. The Z-Inspection® process provides a holistic and dynamic framework to evaluate the trustworthiness of specific AI systems at different stages of the AI lifecycle, including intended use, design, and development. It focuses, in particular, on the discussion and identification of ethical issues and tensions through the analysis of socio-technical scenarios and a requirement-based framework for ethical and trustworthy AI.

3.3 High-level Non-functional Requirements

This section compiles the main high-level non-functional requirements for MANOLO. It also links them with the seven key requirements introduced by the EU Ethics Guidelines for Trustworthy AI (see Table 2) which help AI practitioners adhere to four ethical principles rooted in fundamental rights: i) Respect for human autonomy; ii) Prevention of harm; iii) Fairness; and iv) Explicability. Furthermore, the mapping in Table 3 continues the initial mapping introduced in Table 18 in the project deliverable D1.1, which introduced a broader list of user requirements. The mapping also converges with the use case mapping to be presented in D6.1, which seeks to introduce validation environments for the MANOLO components to ensure that trustworthy requirements are met in the use case design.

ID	Requirement name
TAI01	Human Agency and Oversight
TAI02	Technical Robustness and Safety
TAI03	Privacy and Data Governance
TAI04	Transparency
TAI05	Diversity, Non-Discrimination, and Fairness
TAI06	Societal and Environmental well-being
TAI07	Accountability

Table 2. TAI HLEG Requirements by the EU Ethics Guidelines for Trustworthy AI.

Table 3 lists the project's non-functional requirements (NFR) linked to ensuring trustworthiness, robustness and efficiency. The table provides an introductory description of each requirement and introduces the link with the seven Trustworthy AI (TAI) requirements and the key components in MANOLO that primarily address each non-functional requirement.

ID	Requirement name	Description	TAI HLEG Requirements	MANOLO Key Blocks/Components
NFR01	Model efficiency optimisation	MANOLO will reduce energy consumption and computing processing of AI model operations and improve their performance.	TAI01, TAI02, TAI06	C4_DATA, C5_TRAINING, C6_CLOUD_EDGE, C7_BENCHMARK
NFR02	Efficient AI-hardware allocation in the CEC	MANOLO will provide functionality to recommend for efficient allocation of models across the CEC.	TAI01, TAI02, TAI06	C6_CLOUD_EDGE, C7_BENCHMARK
NFR03	Data efficiency at the edge	MANOLO will reduce the volume of data required in re-training activities at the edge.	TAI02, TAI03, TAI05, TAI06, TAI07	C4_DATA, C5_TRAINING

NFR04	H/W and S/W interoperability & compatibility	MANOLO will run on multiple platforms and provide an environment and tools compatible with different CEC scenarios.	TAI03, TAI04, TAI05	C8_INTEGRATION
NFR05	Model explainability	MANOLO will include mechanisms and techniques to explain models and ensure transparency in terms of training data, model specification and performance.	TAI01, TAI04	C4_DATA, C5_TRAINING, C7_BENCHMARK
NFR06	Data Quality Validation and enhancement	MANOLO will implement methods and techniques to improve quality of the datasets, ensuring data integrity as well as bias mitigation, detecting and addressing bias in data; identifying data quality issues.	TAI03, TAI05	C4_DATA, C7_BENCHMARK
NFR07	Continuous Monitoring	MANOLO will implement pipelines that will allow continuous monitoring of data and models across the CEC.	TAI01, TAI02, TAI03, TAI04, TAI05, TAI06, TAI07	C7_BENCHMARK
NFR08	Regulatory and standard compliance tools	MANOLO will provide tools and methodologies able to generate supporting data to assist with the compliance of existing data and AI-related regulations, with an emphasis on trustworthiness.	TAI01, TAI02, TAI03, TAI04, TAI05, TAI06, TAI07	C4_DATA, C5_TRAINING, C6_CLOUD_EDGE, C7_BENCHMARK, C8_INTEGRATION
NFR09	AI pipeline performance and benchmarking	MANOLO will provide functionality to evaluate the performance of AI pipelines, working on multiple devices at the cloud and edge, and functionality for benchmark purposes.	TAI01, TAI02, TAI04, TAI07	C4_DATA, C5_TRAINING, C6_CLOUD_EDGE, C7_BENCHMARK, C8_INTEGRATION
NFR10	Secure environment	MANOLO will integrate appropriate mechanisms to secure data access and models, within the overall MANOLO framework.	TAI02, TAI03, TAI04, TAI07	C4_DATA, C6_CLOUD_EDGE, C7_BENCHMARK, C8_INTEGRATION

Table 3. High-level MANOLO non-functional requirements.

3.4 High-level Functional Requirements

This section compiles the main high-level (HL) requirements for MANOLO (see Table 4) that describe functionality of the library or suite as a whole; component-specific functional requirements will be introduced in their respective Sections 4-7 and will map to the NFR listed above. An introductory description is provided and their level of importance of each requirement along with the link to the corresponding non-functional requirements introduced in Section 3.3.

ID	Requirement name	Description	Linked NFRs	Importance
FR_HL_01	MANOLO API	The MANOLO API is the entry point of the MANOLO suite. Every interaction with the suite will be performed through this end-point.	NFR10	Must have
FR_HL_02	MANOLO evaluation TAI	Trustworthiness will be kept and evaluated in every single step of the MANOLO pipeline.	NFR07, NFR08	Must have
FR_HL_03	Suite customisation	The customised installation/download of the MANOLO suite will allow the selection of the required tools for a given application, resulting in a reduced version of the dull MANOLO suite.	NFR04	Should have
FR_HL_04	Online catalogue integration	Through the MANOLO API, the user will be able to access online AI resources such as models and data.	NFR01, NFR08	Could have
FR_HL_05	Model registry	The MANOLO suite will incorporate a registry to store models. This functionality will allow the user to store and access AI models.	NFR02,NFR04, NFR07, NFR09	Must have

Table 4. High-level MANOLO functional requirements.

3.5 Framework Overview

The MANOLO framework comprises three main well-defined parts:

- The **MANOLO library**, which includes technical modules developed in the context of the MANOLO project. This is covered in Section 3.6.
- The **MANOLO suite** includes packages for benchmarking, data access, model serving capabilities, and orchestration in the cloud-edge continuum in addition to the MANOLO library. This is covered in Section 3.7. The MANOLO suite will be accessed through the MANOLO REST API, which will be introduced later in the section.
- The **MANOLO external components**: this represents optional external tools that can be used by the MANOLO library and MANOLO suite, e.g. potentially the MANOLO online catalogue.

The MANOLO toolset introduces a sophisticated suite for optimising performance and efficiency of AI models and pipelines in the cloud-edge continuum. Central to MANOLO's design is the integration of flexible deployment strategies, hardware-adaptation, and a lightweight codebase, enabling developers to deploy the system as needed and to respond effectively to changes in their deployment targets, technical environment, and resource constraints.

3.6 MANOLO Library

The most fundamental part of the MANOLO framework is the MANOLO development library aiming to provide AI practitioners with a set of techniques that are easy to incorporate into their pipelines and capable of improving their efficiency and trustworthiness. From data generation to model compression and efficient learning, the main functionality of MANOLO is contained in this library. It provides a high-level class exposed to the user, i.e. the **MANOLO Class** (Subsection 3.6.1), and potentially a **MANOLO Command Line Interface** (Subsection 3.6.2).

Envisioned as a growing tree of elements that provide specific tools, the library root will be the **MANOLO Base module** (Subsection 3.6.3), which will incorporate the common imported technologies to support the whole functionality except for those tools belonging exclusively to the suite. Outside this base, there will be a set of tools that define the **MANOLO Core module** functionality of the library, i.e. data and training and optimisation core modules (Subsection 3.6.4), and another set of eligible extras meant to provide users with components that better accommodate their particular needs (Subsection 3.6.5).

3.6.1 MANOLO Class

The MANOLO Class integrates all the elements from the MANOLO framework from a functionality point of view to provide a continuous pipeline that performs the high-level functionalities of MANOLO. This class includes all the low-level functions required to address the efficiency and trustworthiness of a particular model for a specific application. By interconnecting these low-level functions with the interface described in the following subsection, the MANOLO class ensures that the requirements of the user are met and are aggregated under a high-level interface.

3.6.2 MANOLO Command Line Interface

The MANOLO command line interface (CLI), proposed as an optional element, is illustrated in Figure 5, interconnects the main elements in the MANOLO framework and gives the user access to a high-level instantiation of MANOLO. Through this interface, runnable from the terminal, users will provide their data source and actions to be taken, such as feature extraction, data generation, etc. File path sources of trained models can be provided as well, with options for compression or fine-tuning, albeit the user will have the option to execute NAS or NAG from the class too. Trustworthiness and benchmarking options will be available, with default techniques predefined.

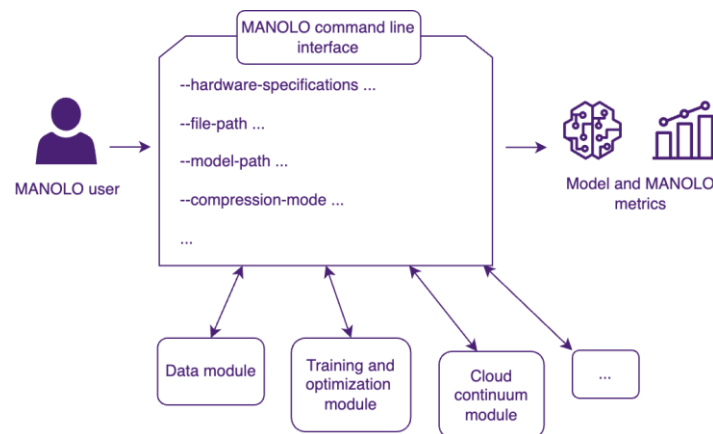


Figure 5. MANOLO command line interface

In short, through this interface, the user should be able to control the whole process being able to call every related element in the library implicitly. Advanced users also have the option of importing separate modules and building their own pipelines at desired levels of complexity.

3.6.3 MANOLO Base Module

3.6.3.1 Base (manolo.base)

The MANOLO base module contains the most elemental blocks used in the MANOLO library. Figure 6 shows an example of the interactions between these blocks. These are class structures to unify and guide the development of new functionalities and ensure compatibility across modules. This module is also used as an initial layer to import libraries and packages that will be used in the main functionalities of MANOLO.

The purpose of establishing a Base module is to make the partners involved in the development of the project responsible for ensuring consistent compatibility of the code with external libraries, which will result in more comfortable and traceable maintenance. Different software design patterns have been discussed, where the adapter pattern is a main candidate to fulfill the purpose of this module, with an agreement on the initially provided compatibility among partners, but open to further contributions by its different partners, as needed. Considering long term goals of the project, we will aim to establish a comfortable structure for future contributors to increase the functionality of MANOLO and expand the lifetime of the software.

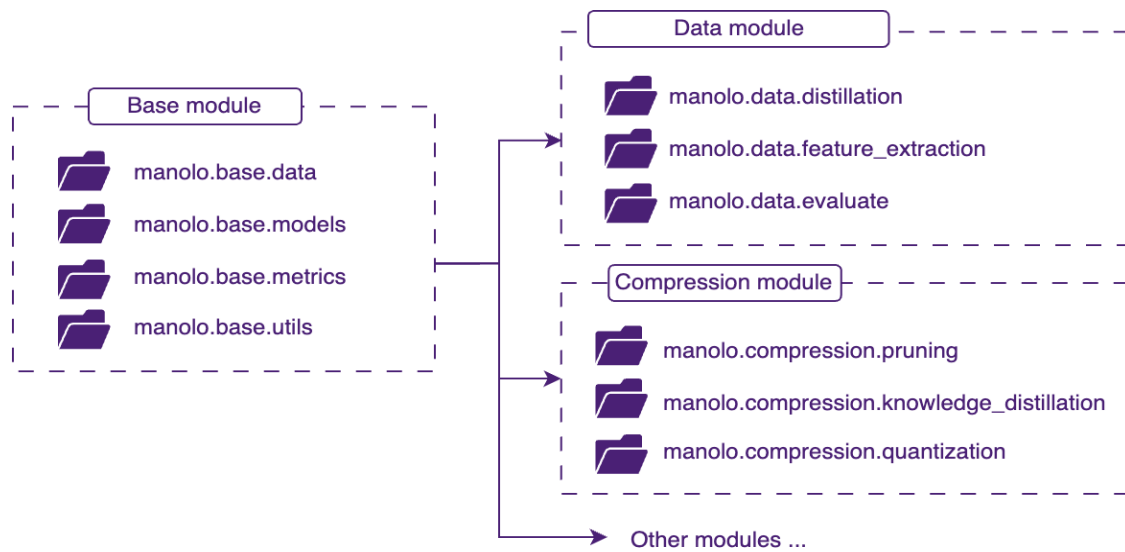


Figure 6. The MANOLO base module provides the building blocks for the rest of the modules to ensure compatibility.

As will be discussed later in this document, the proposed development strategy will be to encapsulate all the different modules under the MANOLO GitHub repository. This will allow certain freedom to the early adopters of MANOLO and developers regarding the usage of libraries particularly appropriate to their development without the need to implement (or request the implementation of) changes in the Base module.

3.6.3.2 Base data (manolo.base.data)

This base submodule will include the wrapper necessary to access the MANOLO data tier which will be further discussed in Section 4, related to the architecture data component. Besides, it will include data loaders and other data-related basic functionality required by the MANOLO library development.

3.6.3.3 Base models (manolo.base.models)

This module will create interfaces to basic models from external frameworks, to leverage their use either for domain adaptation and compression, for users with datasets but no pre-trained model.

3.6.3.4 Base metrics (manolo.base.metrics)

The metrics module implements various metrics for evaluating different elements in the pipeline. This includes performance metrics such as accuracy or F1-score to evaluate the quality of the predictions of a model given a particular task, efficiency and latency metrics to quantify the computational and networking impact of a model, and trustworthiness metrics to assess different elements such as explainability or interpretability.

A set of metrics to be considered initially in MANOLO is listed below.

- **Model Performance:** Accuracy, F1, Cohen Kappa, RMSE, MAE, MASE, MAPE.
- **Efficiency and Latency System/Network Metrics:** FLOPS and RTT.
- **Model Explainability Metrics:** Infidelity, Sensitivity, Expected Calibration Error (ECE).
- **Energy Metrics:** FLOPS/W.
- **Memory Metrics:** Model size (in bits and # of parameters), raw memory used (RAM).

3.6.3.5 Base utils (manolo.base.utils)

Under this submodule, the most convenient and frequently used development tools for the MANOLO partners will be collected.

3.6.4 MANOLO Core Module

The most basic functionality of MANOLO which extends from the Base is implemented in the core module, providing only the most fundamental services required to run the library. This design pattern is extensible, allowing developers to plug and play other MANOLO modules into their environment on an as-needed basis while keeping the local MANOLO installation as light and flexible as possible. Hence, the main components of this module are the HW-aware model training and optimisation module, the data module, and a module with extended functionalities such as Federated Learning.

3.6.4.1 Data Inspection and Generation building block

This building block provides the user with tools to curate and adapt the data to their specific needs. Its functionalities are divided into two components:

1. Data quality.
2. Data distillation and synthetic feature representation.

These components aim to identify quality issues that could affect AI model performance and generate new data or feature vectors to provide a reliable and computationally efficient set of inputs for the models.

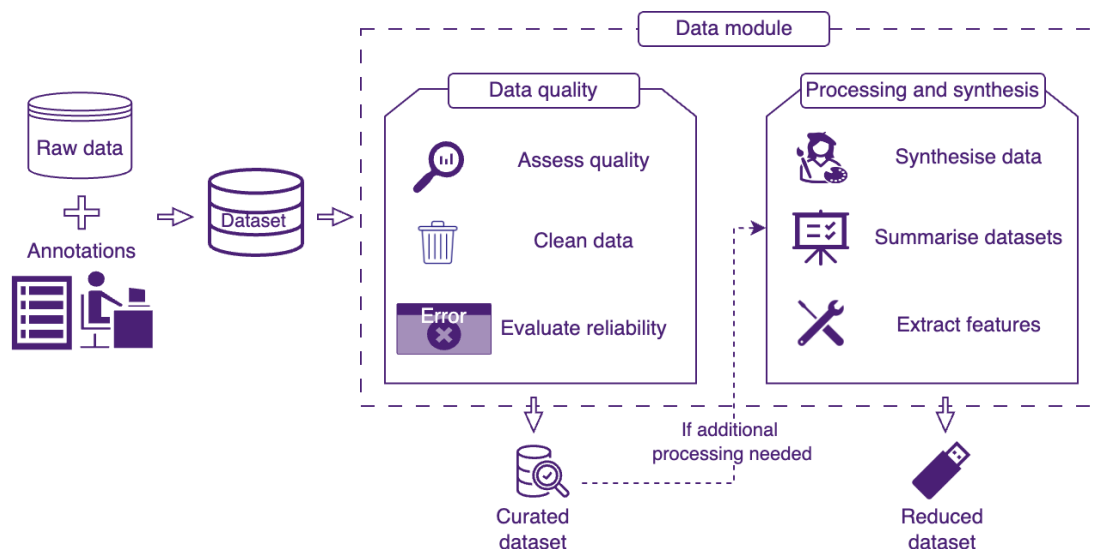


Figure 7. The data building block is split into two big components; the first addresses data quality, and the second further processes the data to increase variety, extract relevant features, etc.

This building block receives the dataset from MANOLO users and provides the tools to exploit the data in the most efficient manner. Figure 7 shows the high-level functionalities included in this module.

3.6.4.2 HW-Aware Model training and optimization building block

The following building block provides the MANOLO library with functionalities to adapt a model to fit the needs of a specific application. In particular, these components include tools and techniques to limit model computational requirements and reduce performance degradation. This subsection introduces the following components:

1. Compression
2. Domain Adaptive Few-shot Continuous Learning
3. Neural architecture search
4. Frugal-oriented neural architecture growth
5. Neuromorphic computing

These components are presented in more detail in Section 5. The Neural architecture search module and the Neuromorphic module will rely on a web-based API service to ensure adequate protection measures for the proprietary solutions. More details about this will be provided in the next project deliverable D3.1 in M18.

3.6.5 MANOLO Extended modules

In addition to the core functionality described in the above sections which will need to be downloaded for the library to operate, the MANOLO library also includes a number of extra modules and tools to suit users' needs which can be extended as per their requirements and settings. This includes extended functionality in the library for federated learning and cloud-edge model allocation recommendation (component covered in Section 6). For instance, the MANOLO CEC model/resource allocation functionality manages the selection of resources for the allocation of the different AI/ML models and functions (for both phases of training/inference) across the entire edge-cloud continuum. The user will request the suggested set of resources for the training or the inference phase. The CEC resource allocation module will apply optimised mechanisms to select the optimal resources according to i) user requirements, ii) the defined policies (including trustworthy policy validation), iii) the model requirements, and iv) the landscape (mapping of the available resources).

These tools and functions will be integrated in the MANOLO library but can also be accessed separately by developers to be integrated into their AI pipelines if there is a need.

3.7 MANOLO Suite and System Architecture

The MANOLO suite is a toolset comprising the MANOLO library described in the previous section (both the core and extended modules) and the following elements/functionality:

- The MANOLO APIs
- User Management & AI Workload Orchestrator
- Data Management and Monitoring Tools (Data-Tier)
- Benchmarking Framework
- MANOLO Online Catalogue (optional)

The complete MANOLO suite is offered via a set of containerised services and thus is designed to operate on the cloud and local servers ensuring compatibility with various system architectures. The MANOLO suite is comprised of two well-defined containerised service modules: 1) the MANOLO Controller and 2) the MANOLO Node. Figure 8 and Figure 9: MANOLO System Architecture present such these two modules and how they are deployed in a network as well as the system architecture considering the different high-level functionality and design elements presented in the previous sections.

Figure 8 presents the deployment of the complete MANOLO suite in any given network: only one instance of the MANOLO Controller is required as a central unit which in charge of interaction with the user, orchestration of AI workloads, global storage & management, as well as benchmarking; in this type of deployment, one or multiple MANOLO Nodes are deployed in the network in a variety of hardware and virtual environments for executing AI workloads as instructed by the MANOLO Controller. This way the MANOLO Controller acts as the central controller of all the AI workloads in the network and benchmarks different AI tasks, data and models in a variety of nodes/hardware and network environments, while aggregating the global results and artifacts for benchmarking purposes and user access. Note that the MANOLO suite can also be used without the MANOLO Controller such that the MANOLO Node via its dedicated API is accessed directly by the user to perform specific AI workloads, without employing the orchestrator and benchmarking functionality; in this modality the user needs to retrieve the required metadata and data from the Node itself, requiring the user to connect to each of the MANOLO Nodes for this task. Finally, in Figure 8, a Constrained Device as such that it does not have capabilities to run the MANOLO Node container and thus it has to rely on use the functionality provided by the MANOLO suite by connecting to a MANOLO Node via specific user scripts.

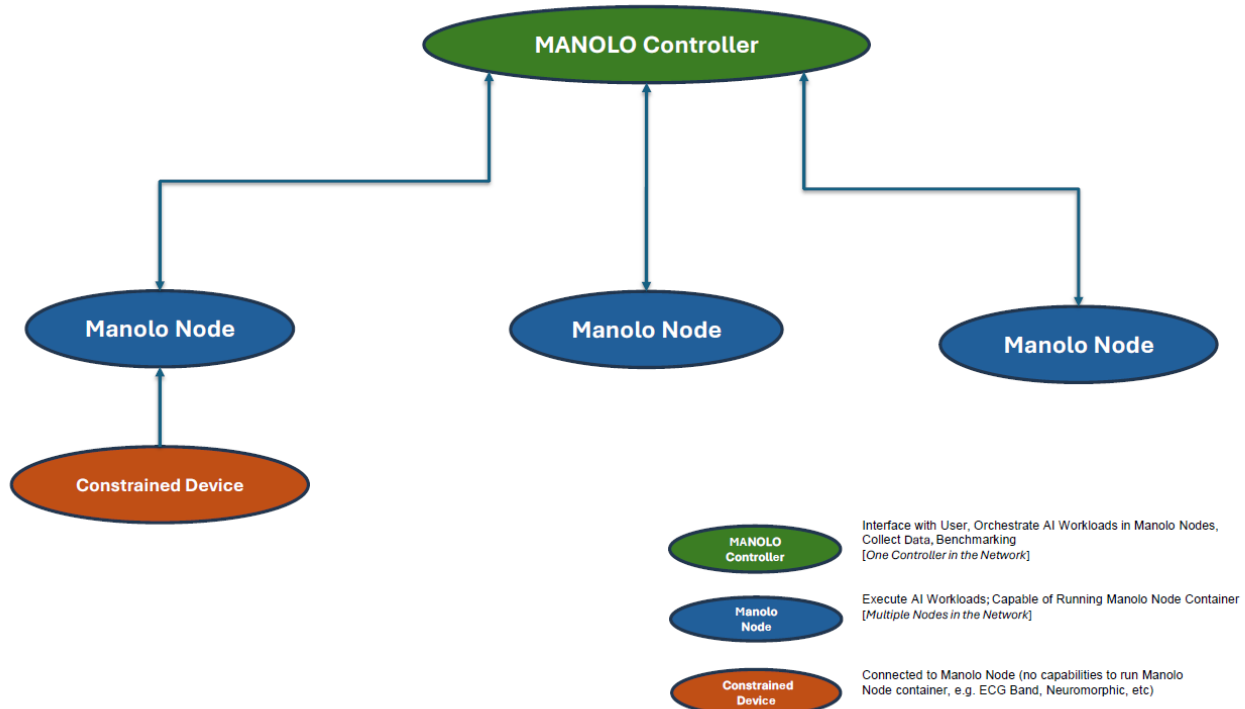


Figure 8: MANOLO High-Level Network Architecture - MANOLO Suite Modules (Controller & Node)

Furthermore, the MANOLO System Architecture is depicted in Figure 9 which illustrates the key components/services under each of the previously described MANOLO Suite modules, i.e. the MANOLO Controller and the MANOLO Node, including their interactions and flow of data.

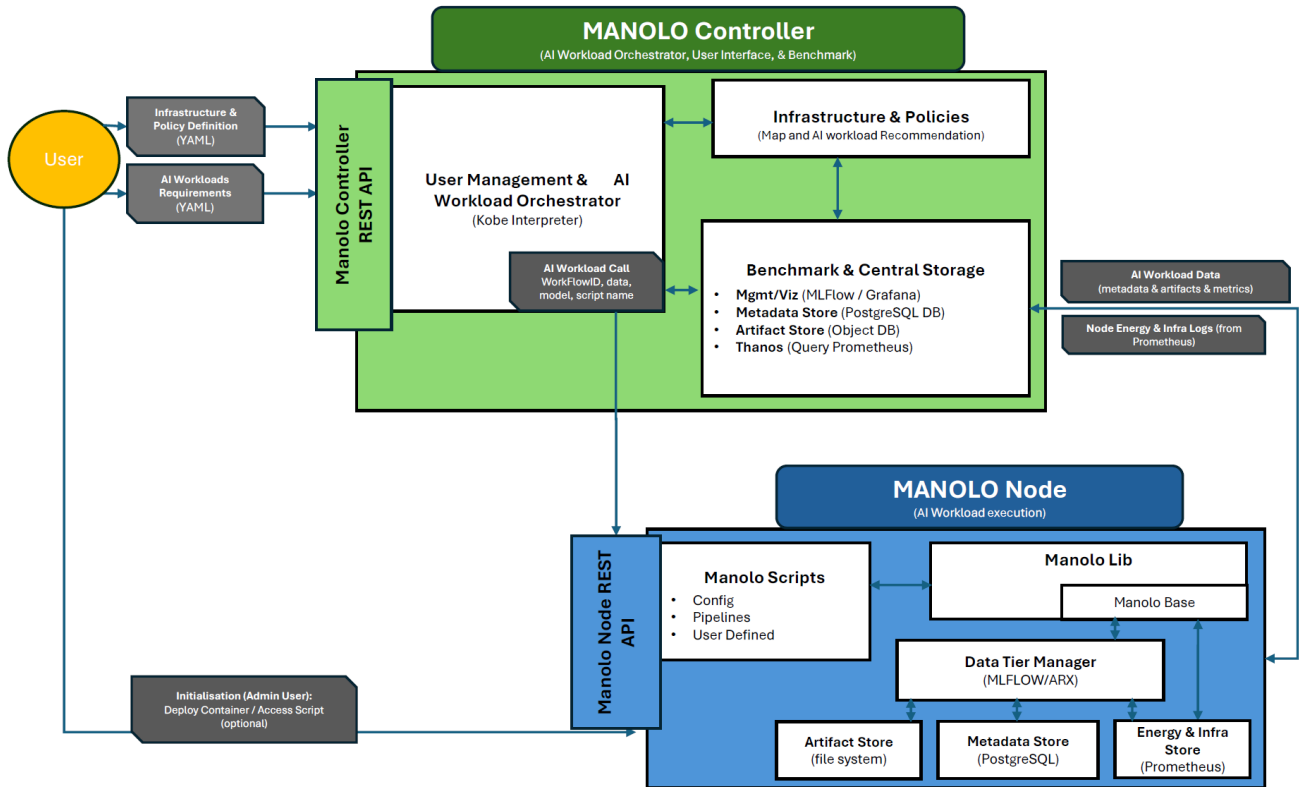


Figure 9: MANOLO System Architecture - MANOLO Suite Modules (Controller & Node)

Initially, the Admin User of the MANOLO Suite needs to download the MANOLO Suite modules in the form of software containers and deploy them in the network as desired, e.g. using Kubernetes. The Admin User is responsible to deploy and initialise them so that the MANOLO APIs are launched and services are available; at this stage the Admin User can configure which users are authorised the system.

Once this deployment and configuration is completed, Users can interact with the MANOLO Controller, depicted in the topmost box in Figure 9, which contains functionality to manage and authenticate users, orchestrate AI workloads, benchmark them, as well as provide a global store and registry of metadata and artifacts which can be accessed by the user to get the results of the workloads as well as artifacts (resulting models, datasets, etc). The MANOLO Controller REST API includes methods to configure the environment and for the user to inject and configure AI workloads by using YAML files which are subsequently processed by the AI Workload Orchestrator and deployed across the network of MANOLO Nodes by calling the MANOLO REST API of each of the nodes. The MANOLO Controller contains tools to store metadata and artifacts (when required) at global level thus providing the benchmark of the AI workloads across the different devices in the network; the registration and management of such metadata and data is performed by MLFlow which is also used for benchmarking. These databases receive metadata from MANOLO Nodes and also are populated via Thanos which polls energy and infrastructure data continuously from each node. Finally, the MANOLO Controller encapsulates functionality for the Admin user to initially map the Infrastructure in the network, along with Policies (network, data, etc) which need to be enforced at the network level; this will be done by

the Admin User via an specific YAML file, and this component will store it in the metadata store. This component will also be used by the AI Workload Orchestrator to check conformity with Policies and to get recommendations of improvements in the AI Workloads against the user requirements based on benchmarking data.

On the other side of the network, the MANOLO Node containers running on multiple virtualised devices will be executing AI Workloads as per the request of the MANOLO Controller's AI Workload Orchestrator. This module exposes the sort of methods to initially configure the node, run pred-defined pipelines or ingest and execute user-defined scripts. These scripts build on top of the MANOLO Library which is also partially accessible via the REST API for advance usage. As discussed in previous sections, the MANOLO Library implements at its foundation the MANOLO Base, which is a lower-level class used for harmonisation but also to centralise and handle the library's calls to store metadata and artifacts. The MANOLO Base calls the Data Tier Manager which employs a combination of MLFlow and ARX distributed data management services to manage the relationships and registry of metadata and artifacts and store these in the local storage systems at the node level as well as at the central storage level in the MANOLO Controller. In parallel, the node's energy consumption and infrastructure properties are managed by the Manolo Base using external libraries and stored in the Prometheus timeseries database (which is regularly queried by Thanos to populate the central storage database of the MANOLO Controller).

3.7.1 MANOLO APIs

The MANOLO suite will include two APIs to efficiently deliver its services to users by leveraging FastAPI, a modern web framework for building RESTful APIs with Python. FastAPI's capabilities will enable seamless interaction with the MANOLO suite from any device in the cloud-edge continuum through the HTTP protocol. Two APIs are defined, one for each Suite module: 1) the MANOLO Controller REST API, which is the central API to be used by Users to configure and execute AI workloads, and 2) the MANOLO Node REST API which, in its full Suite mode, is only called by the MANOLO Controller to request for AI Workloads to be executed at such node/s. The MANOLO Node API will provide a set of high-level MANOLO functionality to trigger processes and ML pipelines and will be linked to the MANOLO library to allow specific tasks to be performed, always under a unique AI Workload identifier which is used for storage and benchmarking coordination. User authentication mechanisms will be provided for the Admin to instantiate them which will provide access to Users of the organisation to the MANOLO Controller API, while securing connections between the MANOLO Controller and the MANOLO Node API.

3.7.2 User Management & AI Workload Orchestrator

The User Management & Workload Orchestrator is key a component to manage the interaction with the User and its requirements and requests to execute AI workloads in the network. Via the MANOLO Controller API, this component receives the following YAML files from the User: 1) the Infrastructure & Policy Definition YAML file, which is used by the Admin user to configure the initial list of nodes in the network, their end points and specifications, as well as general policies pertaining to such infrastructure, data and models, and 2) the AI Workloads Requirements YAML file, which is used to declare the requirements of the AI workload, the artifacts and the list of nodes where the workloads will be run in the network. This component interacts with the Infrastructure and Policies component which is in charge of processing the Infrastructure & Policy Definition YAML file, store this information in the global metadata store via the Data-Tier (see Data Management Tools section below), control inconsistencies in the infrastructure based on updates from the network, evaluate the requirements of the AI Workload against the global policies and state of the infrastructure, and provide

recommendations to the Orchestrator on how to optimise the AI workloads in the overall network based on these information and previous benchmarked data.

Subsequently the AI Workload Orchestrator is in charge of parsing/interpreting the AI Workload YAML file and orchestrate the set of API request calls to the MANOLO Nodes to configure, upload scripts and artifacts when needed and start workloads. The Orchestrator will generate a unique AIWorkflow ID per workload, i.e. in the Central Storage at the Controller, there will be a unique AIWorkflowID + NodeID to store metadata and data for that particular node's workload which will be used for benchmarking purposes. This component will also store data in the Central Storage which can be used to more optimally distribute data to nodes via the Data-Tier. Finally, this component will provide the required end points for the User to access benchmarking information/interface and results from the Workflows in the form of artifacts and metadata, including alerts which will be managed by the User; this component will provide specific controlled actions to act on an AI Workflow running.

3.7.3 Data Management and Monitoring Tools (Data-Tier)

The data tier of MANOLO is a set of data management libraries, tools and databases which operate at both the MANOLO Controller (Central Storage) and MANOLO Node (Local Storage) level and coordinates the overall storage and registry of metadata and artifacts as well as its distribution when needed in the MANOLO environment. The data tier has functionality embedded in the MANOLO Library, specifically in its Base core component, and makes use of tools such as MLFlow to log and register metadata and artifacts, and the ARX data management service developed in MANOLO for distributed storage. These tools operate at the Controller and Node level and are coordinated to store data and metadata in object and SQL databases at both global and local levels. These storage systems aggregate all the artifacts and metadata pertaining to the AI Workloads. Energy data and infrastructure updates of the node are monitored using auxiliary libraries managed by the Base component in the MANOLO Library, using Prometheus Time Series database, and continuously aggregated at the Central Storage in the MANOLO Controller by using Thanos for pulling data from each of the Node's Prometheus instances in the network and storing that in the central metadata storage.

To this end, MANOLO's data tier encapsulates data stores that can be relational, object based and can use the filesystem directly as its data store. Artifacts like datasets and samples are stored in the data tier as objects, while metadata like class annotations and values of metrics can be attached or semantically linked to their respective data objects. Additionally, the data tier provides user authentication, user access control, logging of operations, and encryption for the object's data and metadata.

There is a versatile web API that provides primitive methods to its consumer, which hides the low-level queries executed in the underlying data store and the file system operations. These methods are used by higher-level wrapper objects that are part of the Python framework. The API considers artifacts as item objects that belong to data structure objects and provides methods to handle arrays, lists (collections), trees, graphs, and dictionaries. For each object, there is a unique identifier, its data, key-value pairs for metadata properties, and a list of relations between this object and others. Each relation is a semantic triplet {subject (this), predicate, object (other)} while the data tier supports the definition of custom predicates. Each AI Workload is defined by an AIWorkflow ID, which is generated by the AI Workload Orchestrator and is common to all the nodes which are running such task for benchmarking purposes.

The data can be organised into containers, such as a dataset containing samples, each one containing data points/vector values. Groups of samples like mini batches can be defined without the need to store their data multiple times by using their IDs as part of semantic relations. Moreover, the data tier will choose the

appropriate data store depending on the kind of data provided to parameters of the API methods that store them (e.g., generic, vector, image tensor, signal as time series).

The MANOLO API also allows for the saving and loading of models along with their metadata, organising their artifacts under collection data structures considered model registries. Items that belong to a model registry include:

- A hyperparameter configuration,
- A set of parameter values organised in the model's structure,
- A training state that holds all variables for the training algorithm,
- Benchmark metadata for the training process, and
- Evaluation metrics.

Thus, it allows for the full reproduction of the training context of the model, pausing and resuming training if needed and loading and evaluating trained models. For increased performance, the API provides methods that upload/download files via HTTPS directly to/from the file system.

Through API methods for storing semantics, the data storage tier maintains relationships between models, e.g., model B is derived from model A, as well as relationships between models and datasets, e.g., models A and C are trained on dataset D. The framework is capable of organising multiple experiments under the same context, that can be performed on the same data with different training hyperparameters for the same or different model architecture, or alternatively the same model and training setup on different data.

3.7.4 Benchmarking Framework

The benchmarking framework is a comprehensive building block in MANOLO designed to monitor and evaluate the performance of data analysis algorithms, machine learning algorithms and models, deployed across a range of critical dimensions and infrastructure, via the development of multiple benchmarking experiments. As such MANOLO operates as benchmarking framework when using the MANOLO Controller which gives an overall view and aggregates a comprehensive set of metadata and data from all the AI Workloads conducted in the MANOLO Nodes in the network. The MANOLO Benchmarking framework uses the components in the MANOLO Controller, and functionality at the MANOLO Node for monitoring. Specifically, the Benchmarking and Central Storage component act as the place to create the global view of the AI Workloads in the network, using both the Metadata (SQL) and Artifact (Object) storage databases to aggregate all data related to the AI Workloads, and energy and infrastructure updates related to the node. MLFlow is used as a tool to assist in the organisation and benchmarking of such information which is also analysed to provide a high-level benchmarking view which can both be utilised by the Infrastructure & Policies component to provide recommendations to improve AI Workload YAML requirements for the benefit of the User, and for the User to directly inspect the benchmarked data, potentially providing a graphical view (MLFlow & Grafana).

For benchmarking purposes, the assessment of infrastructure performance is conducted through the analysis of metrics such as CPU/GPU utilisation, memory usage, disk I/O, and inference latency, etc. This approach provides detailed insights into the efficiency with which computational resources are utilised by the systems under evaluation. The efficiency metrics, including energy consumption, time to train models, and time to perform inferences, will assist in determining the effectiveness of the models in operation, thereby optimising both processing speed and resource consumption.

In terms of accuracy and robustness, the engine will assess models both at development time and at deployment time. At development time, the MANOLO Library (Base component) will collect the standard metrics (precision, recall, ROC for tuning, etc.) about the model during training and other data transformation functions. At deployment time, the framework will track system reliability over time, using metrics such as system uptime, error rate, and model degradation. This long-term monitoring will provide insights into the stability and durability of AI workflows. Adaptability will also be a focus, identifying when a model needs to be retrained, ensuring updates of models are uniquely tracked corresponding to new AI Workflows to address challenges like data drift while maintaining optimal performance and giving the User full control of deploying these updated models. By considering these diverse metrics, the benchmarking functionality aims to provide a comprehensive overview of the performance, reliability, and adaptability of AI models in different points of the cloud-edge continuum and giving such information to the User in real-time to make decisions as well as using such information to improve on the AI Workflows requirements.

3.7.5 MANOLO online catalogue (optional)

The MANOLO online catalogue, as an optional functionality currently being explored. It can be executed as a digital catalogue that offers MANOLO users a wide range of resources to simplify the procurement, deployment, and management of efficient and trustworthy AI models. Pre-trained models will be uploadable into this online catalogue, and shared AI practitioners using MANOLO, without the need of being part of the consortium.

This catalogue will allow quick integration into existing environments of AI models that will have already been optimised for solving specific tasks without the need for extensive training time or hardware.

The MANOLO online catalogue will also support private online catalogue options, allowing organisations to create a customised online catalogue with pre-approved AI tools for their teams.

3.8 MANOLO System Usage Scenarios

This section explores the MANOLO toolset through the user's lens. It pinpoints the core features users expect and breaks them down into fundamental actions, such as those basic interactions between the user and MANOLO, with a specific goal and benefit in mind.

3.8.1 MANOLO Library Usage Scenario

The modularity of the MANOLO library provides the user with a wide range of functions to build an efficient and trustworthy pipeline. Figure 10 and Figure 11 provide two possible scenarios that exemplify how users can interact with the MANOLO library. In particular, the diagram in Figure 10 exemplifies a user that applies the compression functionalities to a particular domain following these steps:

- **Load data:** User loads the raw data.
- **Assess data quality:** Data is (re)assessed and evaluated in terms of quality and to identify elements that could hinder model performance.
- **Extract features:** The feature extraction module selects the most relevant features in the data.

- **Adapt the model to a new domain:** model architecture is adapted to the new domain and fine-tuned for improved performance.
- **Compress the model:** The model is compressed following the requirements introduced by the user and the characteristics of the data.
- **Evaluate model:** The model is evaluated on a validation set of the data. If no more compression is needed, the model and the related metrics are provided as outputs of the pipeline.

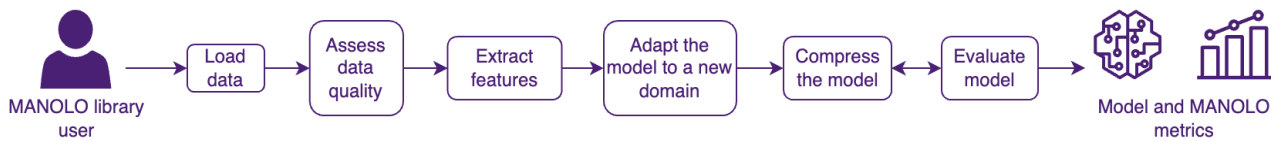


Figure 10. System Use Case (model compression) for the MANOLO library.

Similarly, the diagram in Figure 11 exemplifies a user exploiting the neural architecture search module of MANOLO to obtain an efficient model that fits the provided data:

- **Load data:** User loads the raw data.
- **Assess data quality:** Data is assessed and evaluated in terms of quality and to identify elements that could hinder model performance.
- **Augment data:** The data is processed and augmented to address a lack of variability or imbalance in the available classes.
- **Extract features:** The feature extraction module selects the most relevant features in the data.
- **NAS:** The neural architecture search iterates over several neural network architectures to provide the one that best fits the user requirements.
- **Evaluate model:** The model is evaluated on a validation set of the data. If the model provided by the NAS module meets the user requirements, the model and the related metrics are provided as outputs of the pipeline.

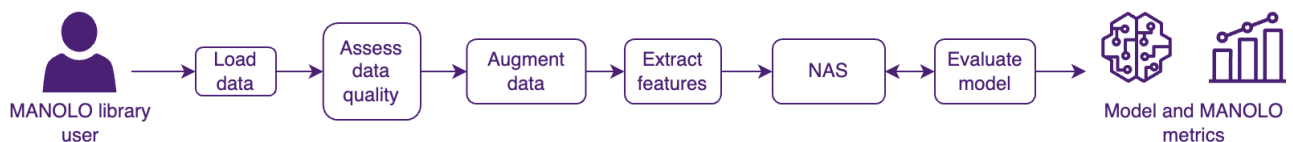


Figure 11. System Use Case (NAS) for the MANOLO library.

3.8.2 MANOLO Suite Usage Scenario

This section explores the System Use Cases for the MANOLO suite. Figure 12 illustrates the functionality of the use cases. In particular, the different elements from the MANOLO suite usage scenario are the following:

- **Deploy suite into container:** Allows the user to deploy the MANOLO suite container to a server or edge devices.
- **Calls API:** External systems or applications can connect to the MANOLO API to access suite services.
- **Access to data tier:** Enables users to securely store, organise, and manage various types of data and saved AI models within the suite. For instance, it allows users to track, manage and revert to different versions of datasets.
- **Define experiments:** A YAML file is created defining the experiment requirements, which will be ingested and orchestrated by the Trustworthy Efficiency and Performance Benchmarking component.
- **Run and benchmark experiments:** Allows users to evaluate and compare performance, efficiency and trustworthiness of AI models using standardised metrics and testing.

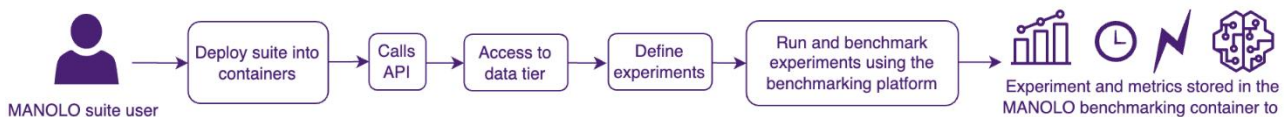


Figure 12. System Use Case for the MANOLO suite

3.8.3 MANOLO Hardware Aware Neural Architecture Search and Neuromorphic Hardware Usage Scenario

A set of sub-scenarios are described where the Neural Architecture Search and Neuromorphic hardware algorithms/functionality is utilised:

- Hardware decision for the user (recommendation for hardware platform), module *Neural Architecture Search Module*
 - The user can call the neural architecture search (NAS) functionality by providing a dataset and pretrained neural network (NN) and then get a report of different possible NNs and their expected KPIs (accuracy, memory, latency, energy) on different supported hardware (HW).
 - The user can then use this report to select a suitable target HW for their use case (UC).
- Development for a specific hardware platform (best model for hardware platform), module *Neural Architecture Search Module*
 - The user can call the NAS functionality by providing a dataset, (pretrained) baseline model, and a specific target HW from the list of supported HW.
 - The user then gets a report of different possible NNs and their expected KPIs (accuracy, memory, latency, energy) for that specific HW.
 - The user can then choose the model with the best trade-off according to their preferences.
- Development of a collection of NNs for a UCs to be deployed on multiple platforms/target HWs with different specifications, module *Neural Architecture Search Module*

- The user can call the NAS functionality by providing a dataset and (pretrained) baseline model and then get a report of different possible NNs and their expected KPIs (accuracy, memory, latency, energy) on different supported HWs.
- During exploration, the NAS functionality determines the models that achieve the highest AI performance (i.e., accuracy) that can be deployed on each supported target hardware according to its specification and available resources.
- The user can then use this report to select a suitable NN and target HW for the UC.
- DNN to SNN Knowledge Distillation (DSKD), module *Neuromorphic Computing Algorithms*
 - The proposed usage for the DSKD module is for a user with a trained DNN. The user can call the central KD function, where the necessary arguments are the trained DNN and the dataset used to train it. The function will return an SNN distilled from the DNN, with reporting on the achieved accuracy and memory usage for the distilled SNN in comparison to the submitted DNN.
 - Advanced options for the KD-function can include: (a) modifying the KD structure, its hyperparameters (see functional requirements), and the SNN-Architecture, (b) making use of compression algorithms for the provided DNN that are integrated in MANOLO (T3.1) before conversion, (c) extended reporting on the speed of convergence of the SNN during KD-supported training and the accuracy in comparison to accuracy in the converted SNN without further training.

3.8.4 MANOLO Cloud-Edge Continuum Allocation Usage Scenario

The following is a user scenario example for the cloud-edge continuum ML model function allocation module. This particular scenario consists of the following steps:

- The user sends a request:
 - Content: Request specifying the resource requirements and trustworthy policies
- Cloud-edge resource and infrastructure mapping process:
 - Identifies the available infrastructure resources that match the user requirements.
 - Maps the identified resources to the available infrastructure.
 - Elaborates and sends the resource mapping plan.
- Trustworthy policy validation:
 - Validates the model and data handling against relevant policy regulations.
 - Approves it and sends the policy-compliant model.
- Trustworthy & Infrastructure Oriented Model and Learning Allocation:
 - Applies optimisation mechanisms to find the resources for model allocation efficiently.
 - Provides the strategy for model/data/resource allocation back to the user

3.8.5 MANOLO Object Manipulation Usage Scenario ([PAL Robotics Use Case](#))

This Use Case in MANOLO brought by PAL Robotics implements the PAL AI system to allow non-expert users to control and interact with a robotic system through natural language. Specifically, the system will help with object manipulation tasks, such as picking up an object or opening a door. To that end, the system will be able to understand verbal commands accompanied by non-verbal signals, like gaze direction or body gestures. This is implemented with an end-to-end pipeline that will be trained using MANOLO components.

The system will be tested in two different scenarios in PAL laboratories: caregiver support in a secondary-care hospital environment and collaborative assembly in a manufacturing environment. As they are privacy-sensitive environments, all the processing will be executed on the robot hardware, and no data will be sent to the cloud. In both cases, MANOLO modules and suite will be used to optimise the PAL AI system to achieve greater efficiency and trustworthiness in resource-limited edge devices. Among other things, this would imply achieving greater frugality and less energy consumption.

1. **Dataset inspection:** a dataset based on Google's Open X-Embodiment dataset together with a Tiago Pro (robot) interaction dataset, will be loaded into MANOLO's Data Inspection and Generation component (data operations manager) for distillation and synthetic feature representation. This ensures that only relevant and high-quality data are extracted for the training process.
2. **Model training:** Using *MANOLO's frugal-oriented neural architecture growth*, *HW-aware neural architecture search*, and *Domain adaptive few-shot continuous learning*, the AI model is iteratively trained and adapted to the hardware constraints in the edge devices while ensuring that the model only grows to meet the requirements of the task while being as lightweight as possible.
3. **Model evaluation:** The model's performance is evaluated using *MANOLO's Benchmarking and assessment manager* to ensure it meets the required accuracy and efficiency metrics. On the one hand, metrics such as response latency, memory usage, or battery consumption will be assessed to ensure the model operates effectively on the robot's onboard resources using *MANOLO's Resource efficiency monitoring*, *Model energy efficiency analysis*, and *Model performance analysis* components. On the other hand, as the robot directly interacts with people, the model robustness will also be taken into account with the *Model explainability assessment* and *ML Model robustness validator* components.
4. **Deployment:** after the model has been evaluated to be frugal (for energy and resource consumption) and robust (for compliance in critical environments) enough, it will be directly deployed into the robot and continuously monitored.

3.8.6 MANOLO Sleep Stage Prediction Usage Scenario ([Bit&Brain Use Case](#))

This use case in MANOLO brought by Bit&Brain implements a real-time closed-loop auditory stimulation intervention during sleep, using minimally invasive EEG-based brain-computer interfaces. A wearable textile headband measures brain activity, which must be analysed in real-time to stimulate the user at precise points in time based on the sleep stage. Patients use the system at home on their own while results are inspected by doctors, which requires an understanding of the decisions made by the system.

We have considered two potential scenarios to map the use case requirements to MANOLO's architecture. In both, the use case employs MANOLO modules to optimise the AI model that decodes the sleep stage and processes slow-wave sleep patterns to trigger auditory stimulation. Each scenario covers different needs for use case implementation.

Scenario 1 - MANOLO library:

Develop an energy-efficient system that operates on a neuromorphic chip embedded in the wearable device, using MANOLO architecture modules for local real-time processing, ensuring that the model is compact, lightweight, and optimised for neuromorphic computing.

1. **Dataset loading:** A pre-recorded dataset from EEG activity during sleep is loaded into the MANOLO pipeline for pre-processing.
2. **Data curation:** *MANOLO's data inspection* component is used to clean and filter the dataset, ensuring only high-quality and relevant data are used for model training. Data anomalies and noise are removed, and relevant features are extracted to optimise training accuracy and efficiency.
3. **Model training:** it can be implemented in two different ways:
 - a. **Frugal-oriented neural architecture growth:** Using *MANOLO's frugal-oriented neural architecture growth* component, an AI model is trained on the curated data. The frugal approach ensures the model grows only as necessary to meet the needs of the task, ensuring it remains lightweight and suitable for running on low-resource edge devices. The model is iteratively optimised during training to balance performance and efficiency, making certain it fits the hardware constraints while maintaining accuracy.
 - b. **Model compression:** A non-frugal model is provided or learned, and the *MANOLO's compression* component is applied to reduce the model's memory and computational footprint.
4. **Neuromorphic computing adaptation:** The compressed and/or frugally optimised model is adapted using *MANOLO's neuromorphic computing* component, which transforms the model into a format compatible with neuromorphic hardware.
5. **Model evaluation:** The model's performance is evaluated using *MANOLO's Model Performance Analysis module* to ensure it meets the required accuracy and efficiency metrics. Metrics such as response latency, memory usage, and battery consumption are assessed to ensure the model operates effectively on the edge device.
6. **Deployment on wearable devices:** After evaluation, the optimised and neuromorphic-ready version of the model is deployed directly onto the wearable device.

Scenario 2 - MANOLO suite:

Deployment of a more flexible system that runs on a tablet rather than on the wearable device, allowing for more computational resources and storage. The system leverages cloud resources to improve performance and personalization through federated learning while ensuring privacy and scalability.

1. **Model loading:** A model previously trained by Bit&Brain is deployed on the tablet within the MANOLO suite.

2. **Data collection and execution:** The wearable EEG headband collects EEG data and sends it to the tablet. The pre-trained model running on the tablet processes the EEG data in real time and determines when to deliver auditory stimulation.
3. **Data curation and anonymization:** Once the sleep session is complete, the collected data is processed through *MANOLO's data inspection and generation* component. This step ensures only high-quality and relevant data is retained for further use. The data is anonymized to protect patient privacy, guaranteeing that no sensitive patient information is exposed during the next steps and quality is analysed.
4. **Federated learning for model updating:** The locally-trained model's parameters are sent to the cloud, where *MANOLO's federated learning module* is used to improve the model. Model parameters from several patients are aggregated into a global model on a central server, allowing the global model to improve based on local model training from different sources without sending the actual patients' data which would compromise their privacy. Continuously evaluation of robustness of global model by using Benchmarking component monitoring and evaluation functionality.

The refined global model is sent back to the individual tablet devices for each patient and downloaded. This ensures that the new model version is personalized based on the individual data and benefits from the collective learning of all participants.

3.8.7 MANOLO Mobile Image Classification Usage Scenario ([ARX.Net Use Case](#))

This use case in MANOLO brought by ARX.NET implements image classification and face detection on mobile devices, preserving user privacy and providing model interpretability to the user. The user selects a collection of photographs stored on a mobile device so that the system can automatically tag them with built-in and user-defined categories. Moreover, the face of the user is detected on images that are tagged as “my face”.

We have considered the following usage scenarios for the capabilities provided by the MANOLO framework. Its primary use will be to facilitate training of custom image classifiers with less complexity and/or reduced data while retaining the same accuracy, thus providing increased efficiency. At the same time, the models should provide interpretations and explanations that will be presented to users to increase the trustworthiness of the AI features that use their photographs.

1. **Dataset creation and loading:** An image dataset of predefined general-purpose classes is created to train a baseline neural network classifier, that is stored in the MANOLO data tier. Additionally, multiple user datasets are maintained, by dynamically adding photographs that are captured and annotated by each user. These images will be used to fine-tune a user-specific image classifier, using MANOLO's dataset loading extensions for the training process that is started by the mobile application.
2. **Data curation:** MANOLO's data curation functionality will be used to ensure compliance of annotations to co-designed ethical and law requirements, for both class descriptions (tags) and image content. Tags should be checked whether they belong amongst black-listed keywords, while the image content can be tested with MANOLO image curation models to ensure ethical use of the mobile application. Image similarity algorithms are used to ensure reliability of content, prominently for “my face” detection, by comparing the annotated face with a photograph grabbed by the mobile camera.

3. **Data reduction:** Data distillation will be employed to reduce the size of the user's image dataset.
4. **Model partitioning:** The functionality offered by MANOLO library, allows to split a neural network model into two parts, a lightweight stem which runs on the mobile edge and the rest of the layers that perform higher level feature extraction, which runs on the computational cloud.
5. **Efficient model training:** The mobile application will use all available MANOLO methods that result in the reduction of training time, both by reducing training set size and model complexity. These are knowledge distillation, model compression, few-shot learning, frugal-oriented neural architecture growth, neural architecture search. Also, will employ Federated Learning to create global classification models that can increase trustworthiness of user-defined classes that have identical semantics.
6. **Model evaluation:** MANOLO's Model Performance analysis module will provide metrics for model performance (accuracy), efficiency and interpretability/explainability. Power consumption, inference latency and memory will be the main indicators used to select a suitable image classifier, with regard to its mobile edge operations.
7. **Model deployment:** The application will use MANOLO's model deployment features to deploy the corresponding user model, either a single or an ensemble of classifiers, that will predict both built-in image tags and user-defined image tags.
8. **Model interpretability:** For a classification/face detection model used by the mobile application (the predictor), MANOLO will provide the corresponding interpretability model (the explainer). Interpretations will be presented to users towards increasing trustworthiness.

Up to this point, the document has covered the high-level aspects of MANOLO, describing and defining its different elements. The document started with an overview introducing MANOLO, presenting several ways that it will address user's needs, and describing its different components (library, suite, and external components). In this and the following section, the focus moves to the specific elements of the different technical building blocks from the project. Firstly, Section 4 addresses the Data Inspection and Generation technical building block. Secondly, Section 5 describes the HW-aware Model Training and Optimisation block. Then, Section 6 introduces the Cloud-Edge Continuum ML Model Function Allocation block. After that, Section 7, introduces the Trustworthy Efficiency and Performance Benchmarking block. Finally, Sections 8 addresses the development and integration of MANOLO, followed by the conclusions from the report.

4. Data Inspection & Generation (NCSR)

4.1 Component Description and Interactions

This Data Inspection & Generation building block, implemented via WP2 in the MANOLO project, focuses on the aspects of data management, inspection and generation within the MANOLO architecture, highlighting their significance in maintaining data integrity and enhancing the overall efficiency of AI applications. It has a total of three components, listed in Table 5. The objectives of this function are the following:

- Create a secure cloud-based repository for managing and storing digital artefacts from the MANOLO project.
- Develop automated methods for annotating and identifying low-quality or unreliable data through anomaly detection and noise detection techniques.
- Develop a framework that generates high-quality distilled data and metadata for training AI models when original data is unavailable or inefficient.

Task	Component	Main Effort / Task Lead	Planned integration in MANOLO
T2.1	Data Management and Provenance Framework	ARX.NET	open-source library
T2.2	Data Quality Estimation	NCSR “D”	open-source library
T2.2	Data Distillation & Synthetic Feature Representation	NUIDUCD-CeADAR	open-source library

Table 5. Data Inspection & Generation: Components, associated tasks, responsible partner and planned integration.

Figure 13 illustrates the high-level interactions between the main data inspection and generation aspects and other modules in the MANOLO suite. Section 4.1.1 covers the *Data Management and Provenance Framework component*, the core data tier which includes methods for organising and managing data structure and item objects. Next, section 4.1.2 explains the *Data Quality Assessment component* which provides algorithms to evaluate data quality, ensure consistency, and identify anomalies. It also employs data augmentation techniques to enhance model robustness. Finally, 4.1.3 details the *Data Distillation and Synthetic Feature Representation component* including tools for feature extraction and data synthesis, aiming to compact datasets without losing performance.

Subsequently, section 4.2 identifies the *Non-Functional Requirements*, this section lists performance criteria linked to functional needs, whereas 4.3 *Functional Requirements* specifies the essential operations the components must support. 4.4 *Technologies* provides an overview of technologies used in development, while 4.5 *Development Cycle* outlines the stages of the component development process, emphasising the need for collaboration among partners throughout the process.

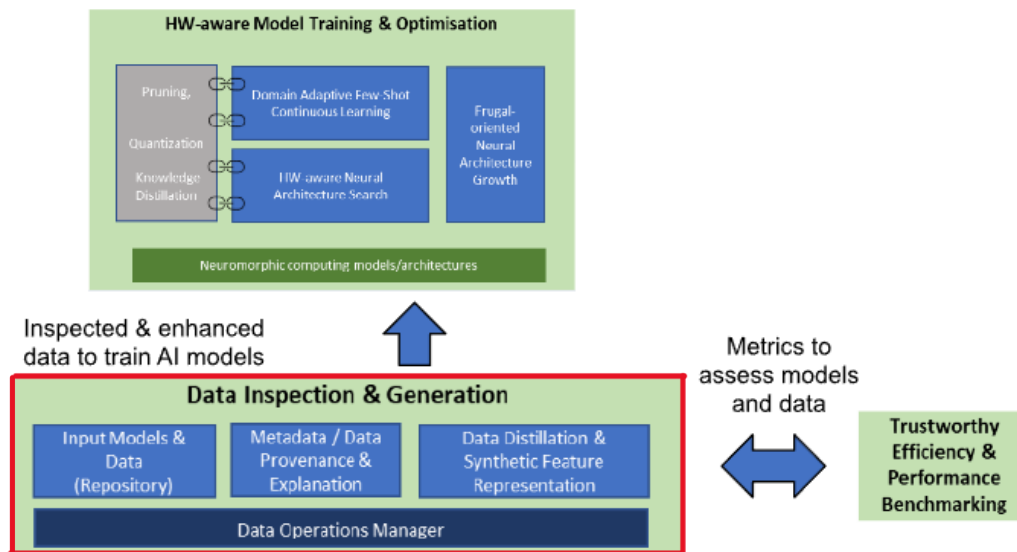


Figure 13. Data Inspection and Generation high-level architecture (highlighted in red).

4.1.1 Data Management and Provenance Framework

The core MANOLO data tier is a self-sustained component that runs processes which are isolated from the rest of the framework. It can be deployed on a host operating system, a virtual machine or a container depending on the DevOps requirements. It contains processes for database management and secure file system management, providing indirect access to the central database that is hosted in the MANOLO controller. Additionally, it implements an encrypted storage in the local file system of the controller and/or the node, offering increased privacy. It is currently using PostgreSQL as its main relational and object data store, while it can store very large files (e.g. model saved states) directly in the file system. Both the contents of binary large object fields (BLOB) in the database and the files in the operating system are encrypted using AES (Advanced Encryption Standard).

The source code for a method/model implementation uses specific functions of the MANOLO data tier to perform an AI workflow via MLFlow, access model registries, and manage metadata that MLFlow stores in the MANOLO central PostgreSQL database. The rest of the MANOLO framework modules use the endpoint */DataTier* that is offered by the ARX data management service, to access data, metadata and artefacts without the need of direct access to the underlying SQL database. The ARX service is providing a unified access to the user, by creating a unique object identifier (UUID) for both its native ARX data entities, and for links that reference MLFlow data entities (e.g. runs). The service can operate through a single opened port with HTTPS, allowing to restrict direct network access to ports of the underlying data services. The API provides primitive methods for data, metadata and semantics management.

Data structure objects and item objects

All MANOLO artefacts are organised as items belonging to data structures, and the MANOLO API provides methods to register and enumerate them. For example, datasets and models are list data structures. The first data structure in the data tier is reserved for maintaining a list of all data structure objects. There is also a build-in data structure to keep special items that link to records in MLFlow tables. When the API consumer asks for the data of an MLFlow record, the service ensures that a link exist in the data tier before retrieving and returning them.

The API methods for item objects require a specific data structure number, and item data are passed as string values to POST methods, while for large files the API consumer can use upload/download streams in via POST methods. For example, image samples are items in a dataset data structure where each image is stored as a file, while time-series EEG samples store their data as a sequence of floating-point numbers. In the example of a neural network model stored as a list data structure, its layers can be handled as different items, while their parameter values are stored in the object's data as real-number tensors. A deletion operation simply flags objects for garbage collection, without initially removing their stored data.

Data semantics

One of the main benefits offered by the MANOLO data tier are the API methods that manage semantic relationships between item objects or data structure objects. There is a method for creating a semantic triplet in the form of subject-predicate-object, and various methods to perform semantic queries on the stored relations. Triplets can combine native ARX entities and MLFlow entities, to facilitate custom MANOLO needs over the standardized data schema that MLFlow provides. The data tier provides extensibility allowing for the definition of custom predicates, apart from the built-in ones. For example, a user can define a custom predicate "run-on" for expressing a relationship between an AI workflow "A" and various infrastructure configurations "B", "C", "D". The data tier stores 3 semantic sentences, each one with the workflow A as the subject, "run-on" as the predicate, and one of the alternative infrastructures "B", "C", "D".

The API offers specific methods to handle parent-child relationships, allowing hierarchical relationships to be stored into tree data structures. For example, we can add many mini-batch data structure objects as the children of a dataset data structure object. The sample item objects will be stored only in the parent dataset. Moreover, it supports any kind of graph besides trees, allowing for undirected (bi-directional) edges that can hold values. Nodes are item objects in the graph data structure and their values are stored in the object's data. The data tier API offers methods to query stored semantic relationships.

Metadata dictionaries and global aliases

Every native entity in the data tier supports a dictionary and the API offers methods for managing its key-value pairs. The dictionary can hold metadata for the object and its relations. For example, on a model training state object that is maintained by MLFlow, a user can attach the keys "total_energy_for_training", "mW_per_epoch", "mW_per_sample" to pair them with the metrics that are handled by MLFlow like accuracy, precision, F1 score. Global aliases are unique names that are used instead of object IDs in the data tier, providing a user-friendly access for frequently used artefacts.

4.1.2 Data quality estimation and augmentation

The algorithms in this component (manolo.data.quality) assess data quality in terms of annotations and samples. These analyse the training data in terms of homogeneity of the sample distributions, consistency of the labelling, and absence of anomalies. The aim of this submodule is to be used to filter out data that is expected to reduce model accuracy if included in the training set. Data augmentation and distillation will then be used to prepare high-quality training datasets at the volumes needed for deep learning.

Although downstream ML algorithms might be robust to noise or bias, early detection of these issues a) gives human oversight on data that are ignored; and b) improves computational efficiency.

The data quality assessment module exposes an interface similar to that of standard machine learning modules:

- Training involves building a data quality estimation model using labelled data, if available. These labels correspond to the task-specific classes (e.g., classification), rather than to data quality itself. If labelled data is not available, the model uses an unsupervised learning approach (e.g. reconstruction).
- Testing evaluates the model's performance using data quality labels if they are available, measuring both effectiveness and energy efficiency. If data quality labels are not available, the evaluation is done by comparing the performance of the downstream model when trained on the original dataset versus the filtered one.
- The deployed model accepts data as inputs and responds with data quality labels, inferred from its behaviour on the task.

This component also provides functionality for Data Augmentation as a module which employs advanced techniques to generate new synthetical data. It enhances AI system performance across domains and data types, supporting multiple augmentation methods for various data formats. By exposing models to diverse data variations during training, it improves robustness, reduces overfitting, and increases adaptability to real-world scenarios, strengthening the reliability and performance of ML systems throughout their lifecycle.

4.1.3 Data distillation and synthetic feature representation

This component includes tools for compacting data in terms of feature extraction and data synthesis. The former provides alternatives to process data and extract relevant lower-dimension features to be used as densely informative inputs for the model. The latter leverages data distillation techniques to compress the available data into compact datasets that are smaller in size but maintain the performance achieved with the original datasets. Further specifications of the methods used for particular modalities will be described in the corresponding deliverable, i.e., D2.1 v1.

4.2 Functional Requirements

Requirement ID	Requirement Name and Description	Linked NFRs	Importance
FR_C4_01	Data System: MANOLO will create data storage accessible by every other component in the framework through an endpoint, and an API allowing human-in-the-loop operations.	NFR03, NFR07, NFR08, NFR10	Must have
FR_C4_02	Registry access: MANOLO will allow authorised users to load and save models, datasets and algorithms from a registry.	NFR09, NFR10	Must have
FR_C4_03	Data quality assessment: MANOLO will identify and report dataset samples that are a potential source of	NFR01, NFR06, NFR07, NFR08	Must have

	model performance degradation.		
FR_C4_04	Feature extraction: MANOLO will include methods for representative feature extraction from samples, facilitating the transition to compressed datasets.	NFR01, NFR03	Must have
FR_C4_05	Data synthesis and generation: MANOLO will implement data synthesis and generation methods to overcome original dataset shortcomings and train better models.	NFR03, NFR06	Must have
FR_C4_06	Data Distillation: MANOLO will provide tools to distil datasets for a more efficient training of the models.	NFR01, NFR03	Must have
FR_C4_07	Metadata: MANOLO will create and associate metadata to datasets in order to trace the origin, quality and transformations of the dataset.	NFR02, NFR06, NFR07, NFR08	Must have
FR_C4_08	Data Augmentation: MANOLO will provide tools to transform and add noise to data samples in order to improve the robustness of models.	NFR06	Should have

Table 6. Data Inspection & Generation: Functional Requirements

4.3 Technologies

Table 7 shows a mapping of the technologies used, as well as potential 3rd party technologies that will be utilised for the component's development, the project task associated with the development of the components, and the responsible partners that contribute to the implementation of the component.

Component	Technology	Task	Responsible Partner
Data distillation	PyTorch, Transformers, Kornia	2.3	CeADAR
Data generation/synthesis	PyTorch, FrEIA	2.3	CeADAR
Data storage	BentoML model registry	2.1	CeADAR, Arx.NET
Data augmentation	PyTorch, TensorFlow, Keras	2.2	CeADAR, ATOS
Data quality estimations	Transformer, Frouros	2.2	NCSR, FDI

Table 7. Initial mapping of proposed technologies for Data Inspection & Generation building block.

Publisher	Dataset Name	Link	Description	Baseline Model
BIT&BRAIN	Internal dataset	N/A	EEG data, annotated with noise level and sleep stage	Quality Assessment
InsightFace	InsightFace	Github	Dataset for FIQA (Face Id Quality Assessment)	InsightFace

Table 8. Initial mapping of baseline datasets to support early Data Inspection & Generation development

4.4 Development Cycle

The general approach of WP2 is to simultaneously develop the core data service (T2.1, Section 4.1.1) and the methodologies and early prototypes for data quality estimation (T2.2, Section 4.1.2) and data distillation (T2.3, Section 4.1.3) and then use the first version of the data service as the integration base for the data quality estimation and data distillation prototypes.

Step 1:

- As a starting point of the research and technical implementation of WP2, a reference point (RP) has been agreed on as a set of datasets which are well-studied and where a) numerical quality estimations are known; b) the quality of content and features generated through data and feature synthesis can be estimated by prior, domain-specific methods.
- The RP includes data from different modalities (such as images and time series) to verify the research outcomes in diverse settings.
- A first version of the RP has been established in M6, initially provided by Bit&Brain, and updated with further open datasets in M18.

Step 2:

- The three tasks have proceeded in parallel during the first 18 months, working on static datasets.
- The first version of working prototypes was delivered in M12.

Step 3:

- Integrate the data quality estimation & data distillation prototypes into the data service.
- This integration will result in a data service offering data quality estimation/data augmentation/data distillation services that can be used for testing on live non-static pilots.
- The RP is extended with further datasets from the use case partners and/or relevant open datasets.

Step 4:

- The integrated WP2 data tier is refined and improved under continuous integration during the second period of WP2, until M24.
- The integrated WP2 data tier is integrated into the overall MANOLO system by M24.

5. HW-aware Model Training and Optimization (Fraunhofer)

5.1 Component Description and Interactions

This technical building block focusses on the HW-aware ML model operations and algorithms, and is developed in WP3 in the project. It has a total of five components, clearly mapped to 5 tasks, listed in Table 9. The objectives of this technical building block are the following:

- Develop a complete set of novel NN model compression and reduction and optimisation algorithms and tools
- Develop domain adaptive Few-shot continuous learning algorithms using meta-learning and multi-task optimisation.
- Abstract optimisation of HW-specific models away from the UCs using energy-efficient Network Architecture Search methods.
- Develop novel Frugal-oriented Neural Architecture Search algorithms.
- Train Spiking Neural Network models from a teacher Deep Neural Network model using knowledge distillation.
- Co-define and implement the interfaces to register algorithms and models between all the relevant modules
- Integrate the already developed neuromorphic HW toolchain to train, simulate, and deploy DNNs/SNNs.

Task	Component	Main Effort/ Task Lead	Planned integration in MANOLO
T3.1	Compression: Model Pruning, Quantization, and Knowledge Distillation	CeADAR	open-source library
T3.2	Domain Adaptive Few-shot Continuous Learning	CeADAR	open-source library
T3.3	HW-aware Neural Architecture Search	Fraunhofer	web-based API
T3.4	Frugal-oriented Neural Architecture Growth	INRIA	open-source library
T3.5	Neuromorphic computing	Fraunhofer	web-based API

Table 9. Model Training & Optimisation: Components, associated tasks, responsible partner and planned integration.

The relationship between different functions in MANOLO with regard to this development process is illustrated in Figure 14.

The following subsections introduce the modules that form the HW-aware Model Training and Optimisation part of the MANOLO framework. First, Subsection 5.1.1 describes the compression module and introduces the basic compression techniques included. Second, Subsection 5.1.2 presents the continual learning and adaptation module and describes the meta-learning module. Third, Subsection 5.1.3 presents the neural-architecture search client. Fourth, Subsection 5.1.4 introduces the frugal-oriented neural architecture growth module. Finally, Subsection 5.1.5 presents the neuromorphic computing client.

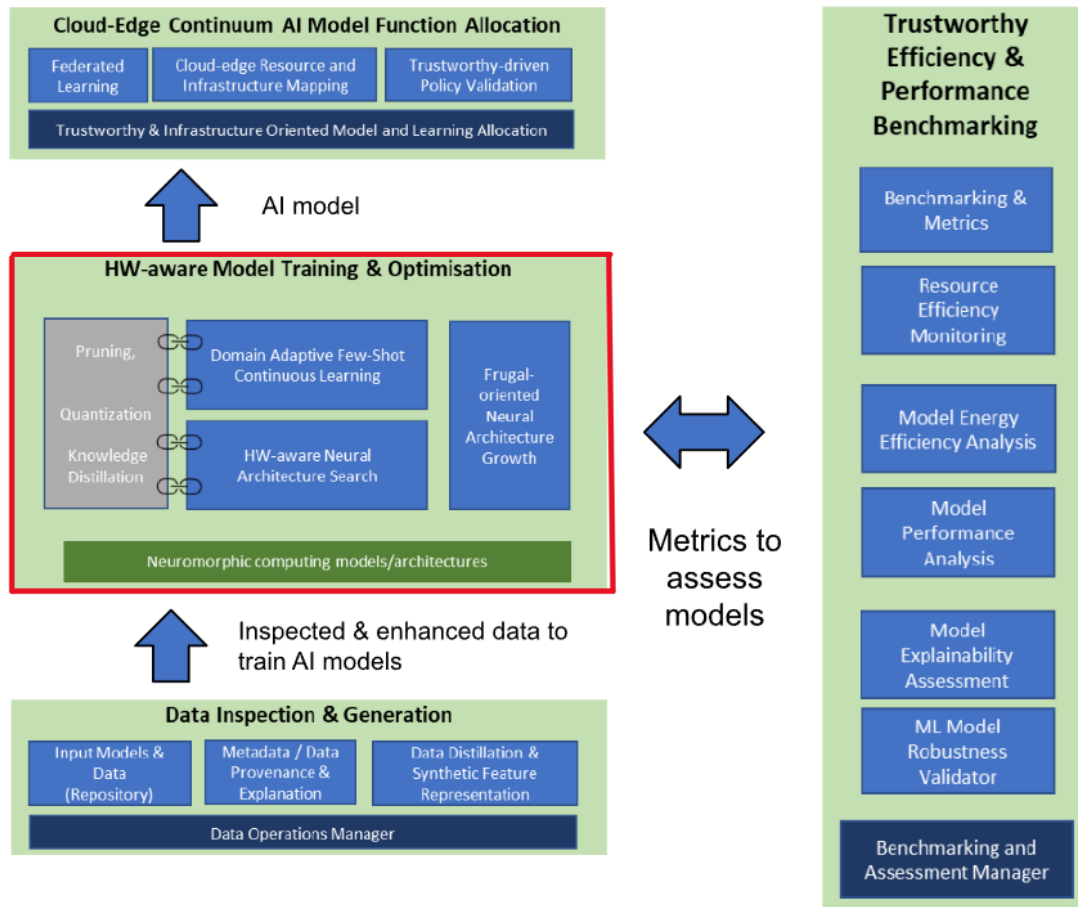


Figure 14. HW-aware Model Training and Optimisation high-level architecture (highlighted in red).

5.1.1 Compression: Pruning, Quantization and Knowledge Distillation

The compression component includes a variety of algorithms to reduce model computational requirements in terms of memory, speed, or energy consumption. These algorithms leverage knowledge distillation, pruning, and quantisation techniques, individually and in conjunction, to accommodate the computation needs of the user. These techniques are divided into the three main modules in the MANOLO compression component: `manolo.compression.distillation`, `manolo.compression.pruning`, and `manolo.compression.quantization`.

Figure 15 outlines the basic principles behind the three compression techniques included in the MANOLO compression component.

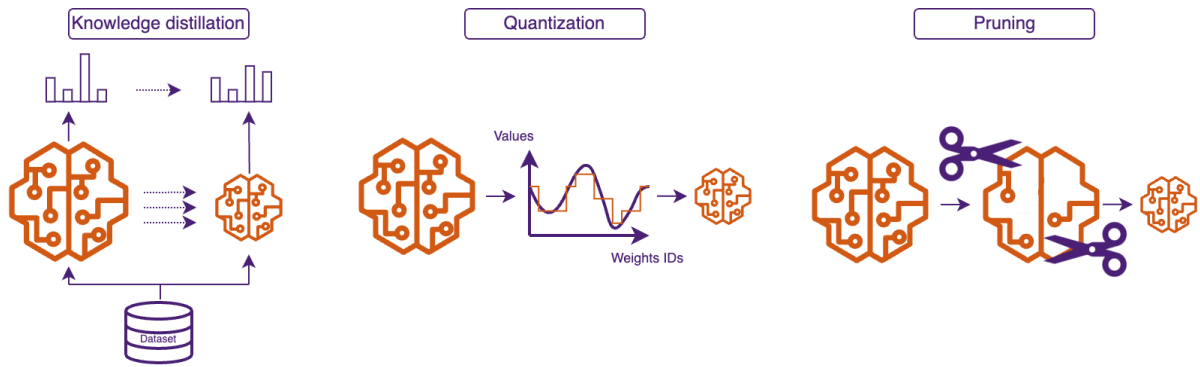


Figure 15. The compression component: it leverages the three most widely used compression techniques: knowledge distillation, quantization, and pruning.

Knowledge distillation algorithms work under the teacher-student paradigm, providing the tools to guide the training of small student models with large teacher models, improving the final student performance. Pruning techniques strategically remove parts from the model to reduce its computational requirements. Quantisation techniques, on the other hand, decrease computational overhead by representing the weights and activations of the model with low-precision data types instead of the standard full 32-bit floating point values.

This component aims to achieve extreme compression by combining different techniques and exploiting synergies with feature extraction methods from the data module. The Compression module includes tools to maintain trustworthiness after model compression, and it is designed to work in conjunction with the techniques in the Domain Adaptive Few-Shot Continuous Learning component. For its successful operation, the Compression module requires curated and high-quality data, which will be provided by the Data Inspection and Generation building block.

5.1.2 Domain Adaptive Few-shot Continuous Learning

This component provides the necessary functionalities to maintain and adapt deployed models. These include techniques to monitor changes in the data distribution and changes in the model as well as tools to accommodate those changes. This will also include functions to allow continuous fine-tuning and partial fitting of the MANOLO models.

The domain adaptation module compiles tools to integrate the model into the data encountered at inference time. These include algorithms designed to address changes in the data distribution between training and testing, as well as changes in data over time. By including few-shot learning techniques, this module facilitates the adaptation of models when data in the target domain is scarce, or samples have missing annotations. To this end, the meta-learners extract meta-parameters from the model and data and utilise these for model retraining and enhancing adaptability. Figure 16 presents a generic diagram of the meta-learning setup used in MANOLO.

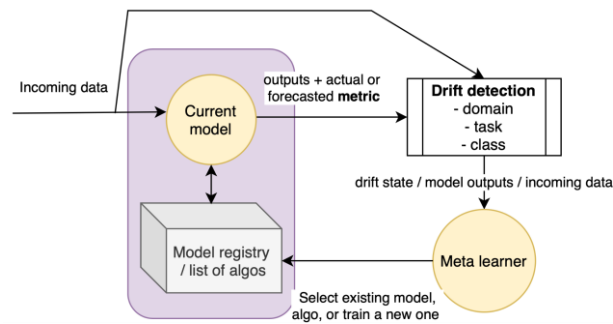


Figure 16. The meta-learning module: it provides the tools to maintain model adaptation to the target task.

The meta-learning and multi-task functionalities of this module aim to enhance the adaption of inference and compression algorithms. These features provide the user with various models tailored to the requirements in terms of the target domain and computational restrictions. Similar to the compression component, this component interacts with the models and algorithms in the compression and data modules.

5.1.3 Neural architecture search

The neural architecture search (NAS) component allows users to automatically match their specific task with the most effective neural network architecture for its execution based on the nature of the task and the target hardware.

Doing so will reduce the time and effort in the development of AI models built using the MANOLO NAS submodule by leveraging automated design.

Currently, it is planned to provide access to this component through a web-based API, ensuring adequate protection measures for this proprietary solution. More details will be provided in deliverables D3.1 and D7.5.

5.1.4 Frugal-oriented neural architecture growth

MANOLO implements the neural architecture growth AI training paradigm, a novel method that increases model size as training progresses by adding neurons to the neural network. The purpose of this is to reduce the training requirements of AI models to just one training phase, which will adaptively grow an optimal model architecture depending on the task at hand, thus eliminating the need for users to train multiple separate models while determining which is most performant.

Making use of this technique also avoids having to train large models and then reduce them using compression techniques, sparing time and computational resources.

5.1.5 Neuromorphic computing

The MANOLO project's Neuromorphic computing component will facilitate energy-efficient AI through the development of innovative training algorithms for Spiking Neural Networks (SNNs). These novel algorithms will optimise SNN parameters by combining knowledge distillation techniques and existing SNN-supervised learning methods, such as surrogate gradient descent.

Compatibility with hardware-specific requirements will be ensured to streamline deployment on power-efficient edge devices.

Currently, it is planned to provide access to this component through a web-based API, ensuring adequate protection measures for this proprietary solution. More details will be provided in deliverables D3.2 and D7.5.

5.2 Functional Requirements

Requirement ID	Description	Linked NFRs	Importance
FR_C5_01	Model Compression factor: MANOLO will provide tools to reduce model size by a factor of 3 or higher by leveraging knowledge distillation, quantisation and pruning techniques.	NFR01, NFR03	Must have
FR_C5_02	Energy and resource consumption reduction: MANOLO will provide tools that reduce the energy and compute time requirements for training DNN models by at least 30% compared to a baseline while preserving accuracy or estimated loss metrics in 4 different hardware architectures	NFR01, NFR09	Must have
FR_C5_03	Development of novel efficient AI model algorithms: MANOLO will deliver next-generation, hardware-aware AI algorithms through the development of at least 4 novel algorithms for energy/data efficient ML.	NFR01, NFR02, NFR03	Must have
FR_C5_04	Model and hyperparameter selection: When given a dataset, model, and task, MANOLO will return the base model and the set of hyperparameters that best suit the current scenario.	NFR01	Should have
FR_C5_05	Performance degradation estimation: Given a trained model, MANOLO will provide an estimation of the concept drift of the model.	NFR07, NFR09	Should have

FR_C5_06	DNN model sampling: Given a baseline architecture or a model, MANOLO will provide a sampling strategy to propose a smaller model or a set thereof in an informed manner so that it minimises latency, memory usage, and energy expenditure while optimising AI and explainability metrics.	NFR01	Should have
FR_C5_07	NAS-based model training: Given a NAS-informed DNN architecture and a dataset, MANOLO efficiently trains the DNN using compression, KD and pruning techniques.	NFR01	Must have
FR_C5_08	DNN model conversion: Given a trained and possibly compressed model in a high-level graph-oriented format, i.e., ONNX, MANOLO can lower the representation for deployment on supported microcontrollers and neuromorphic hardware.	NFR04	Should have
FR_C5_09	Metric estimation: Given a NN model topology and HW target (i.e., supported microcontrollers and neuromorphic hardware), MANOLO will estimate the relevant metrics to be used by FR_C5_08.	NFR09	Should have
FR_C5_10	DNN model growth: Given a DNN baseline architecture and a dataset, MANOLO provides the tools to train and grow a DNN on the given data to reach target accuracy while limiting the number of parameters.	NFR01	Must have
FR_C5_11	Develop novel strategies for growing networks: At least two strategies should be proposed for architecture growth.	NFR01	Should have
FR_C5_12	Basic SNN-based Knowledge Distillation:	NFR04	Must have

	Given a trained DNN and a dataset, MANOLO will perform knowledge distillation from DNN to SNN with a defined default mode.		
FR_C5_13	Convergence and Accuracy Estimation: MANOLO will estimate SNN accuracy over training with knowledge distillation.	NFR09	Should have
FR_C5_14	Spike Encodings: MANOLO will offer multiple options to encode different types of data into spike trains (rate- and temporal-coding; further options tbd).	NFR04, NFR06	Must have
FR_C5_15	Advanced SNN-based Knowledge Distillation: MANOLO will allow for the customisation of SNN architecture (neuron model, encoding, initial connectivity) and the structure of layer-wise feedback from teacher DNN to student SNN.	NFR04	Should have
FR_C5_16	DNN to SNN conversion: Given a trained DNN, MANOLO will offer an established conversion function to generate a baseline SNN.	NFR04	Must have
FR_C5_17	SNN Training: Given a baseline SNN model and a dataset, MANOLO will be able to train the SNN model on the data.	NFR04	Should have

Table 10. HW-aware Model Training and Optimisation: Functional requirements.

5.3 Technologies

The initial development of this technology building block developed in WP3 has been proposed to leverage PyTorch, but this may evolve to include different AI libraries or backends. Models will conform to the base class of MANOLO and versions will be agreed. The format of datasets is not strictly defined, but a standard data loader of PyTorch should be used during the WP3 implementation process. This allows the use of a variety of datasets while ensuring that a standardised API will be in place in the MANOLO suite.

Generally, the datasets used during the implementation of WP3, should be inspected through the tools of WP2. Apart from the data sets and models provided through the use cases, publicly available datasets should be used for the implementation process of WP3. The final list of suggested models and datasets will be specified in upcoming WP1, WP2, and WP3 deliverables.

Publisher	Dataset Name	Link	Description	Baseline Model	Link to Baseline Model
MLPerf Tiny	Speech Commands	Github	Small vocabulary keyword spotting	DS-CNN	Github (not in Pytorch yet)
	Visual Wake Words Dataset	Github	Binary image classification	MobileNet	Github (not in Pytorch yet)
	Cifar10	Github	Small image classification	ResNet	Github (not in Pytorch yet)
	ToyADMOS	Github	Detecting anomalies in machine operating sounds	Deep AutoEncoder	Github (not in Pytorch yet)
FAU Erlangen-Nürnberg	Daliac	FAU	Hierarchical, multi-sensor based classification of daily life activities	[TBD]	[TBD]
Case Western Reserve University	CWRU	CWE	Detection of normal and faulty motor bearing	[TBD]	[TBD]
Google	mini-ImageNet	Github	Github	ResNet	1) Github ; 2) Github
Stanford	IMDB	Stanford	Binary sentiment classification	BERT	Huggingface
Microsoft	MRPC	Microsoft	Paraphrase identification and generation	BERT	Huggingface (huggingface documentation with model, specs, and instructions)
N/A (synthetic data)	N/A	Arxiv	Transmit waveform for bit streams in simulated 5G-compliant PUSCH-link	DeepRX	Arxiv (paper describing the model and data generation)

Table 11. HW-aware Model Training and Optimisation: Initial mapping of potential baseline datasets and models in WP3 to support the early development of this building block.

Table 12 shows an initial mapping of some of the technologies that could be utilised for the component's development, the project task associated with the development and the responsible partners that contribute to the implementation of each component.

Component	Technology	Task	Responsible Partner
Continual Learning	Avalanche, PyTorch, River, NannyML	3.2	CeADAR
Compression	PyTorch, Torch-pruning, NNI, Torchmetrics, CaptumTorchtext, PytorchCV	3.1	CeADAR
Quantization Aware Training	Brevitas, PyTorch	3.3	Fraunhofer
Knowledge Distillation	PyTorch, snnTorch, JAX	3.5	Fraunhofer

Table 12. HW-aware Model Training and Optimisation: Initial mapping of proposed technologies.

5.4 Development Cycle

The general approach of WP3 ensures that the efforts of the different tasks in WP3 are kept separate from each other while allowing for the comparison of the different components and learning from each other. Throughout the course of WP3, the trustworthiness of the components will be iteratively assessed with Z-inspection®.

Step 1:

- This first step took from M5 (start of WP3) until M18 (due date of D3.1 v1). In this time, the partners of WP3 discussed on a monthly basis about the progress of the research for each task within WP3. The scientific approach for the respective task was developed by the involved organizations. The progress of this was reported in D3.1 v1.

Step 2:

- In a second step, the developed components will be integrated into the MANOLO framework and an initial demo will be shown in M21 during the technical review.

Step 3:

- After the initial integration into the MANOLO framework and before the end of WP3 (M30), the developed components will be iteratively improved and further tested on additional data sets. The progress of this final step will be reported in D3.1 v2 and D3.2 (M30).

6. Cloud-Edge Continuum ML Model Function Allocation (UPC)

6.1 Component Description and Interactions

The “Cloud-Edge Continuum AI Model Function Allocation Architecture” technical building block, implemented in WP4, has a total of four components listed in Table 13 along with the respective associated tasks.

The objectives of this function are the following:

- Implement federated learning functionality to support training of AI models with minimal data exposure, reduced data exchange, and lower energy consumption.
- Facilitate communication and interaction through APIs between different components that need landscape and resource allocation information, and the internal components of the Cloud-Edge Continuum ML Model Function Allocation.
- Map, abstract, and categorize computational resources and AI assets across the distributed, heterogeneous cloud-edge infrastructure.
- Design and develop a policy compliance manager to ensure AI/ML models meet predefined trustworthiness, performance, and availability criteria.
- Develop a mechanism to recommend optimal to allocation of AI/ML models across the cloud continuum while ensuring efficient resource utilization.

Task	Component	Main Effort / Task Lead	Planned integration in MANOLO
T4.1	Federated Learning	FDI	MANOLO Suite
T4.2	Cloud-Edge Resource and Infrastructure Mapping	TUBS	open-source library
T4.3	Trustworthy-driven Policy Validation	ATOS	MANOLO Suite
T4.4	Trustworthy and Infrastructure Oriented Model and Learning Allocation	UPC	open-source library

Table 13. Cloud-Edge Continuum ML Model Function Allocation: Components, associated tasks, responsible partner and planned integration.

The *Cloud-Edge Continuum AI Model Function Allocation Architecture building block in WP4* is depicted in Figure 17 and Figure 18 which illustrate both the relationships between tasks, components and their interactions within the MANOLO suite (in both the MANOLO Controller and MANOLO Node).

Initially, Task4.2 is focussed on developing the Cloud-edge Resource and Infrastructure Mapping component which starts by characterising and mapping network resources via a YAML file (top YAML in Figure 18) used by the User Admin. This information is subsequently updated by live network data gathered by each Node’s infrastructure monitoring tool. This component will maintain a current view of the infrastructure which

will be used as input in Task 4.4 for allocating against the requirements of the User (provided as a different YAML for AI Workload Requirements).

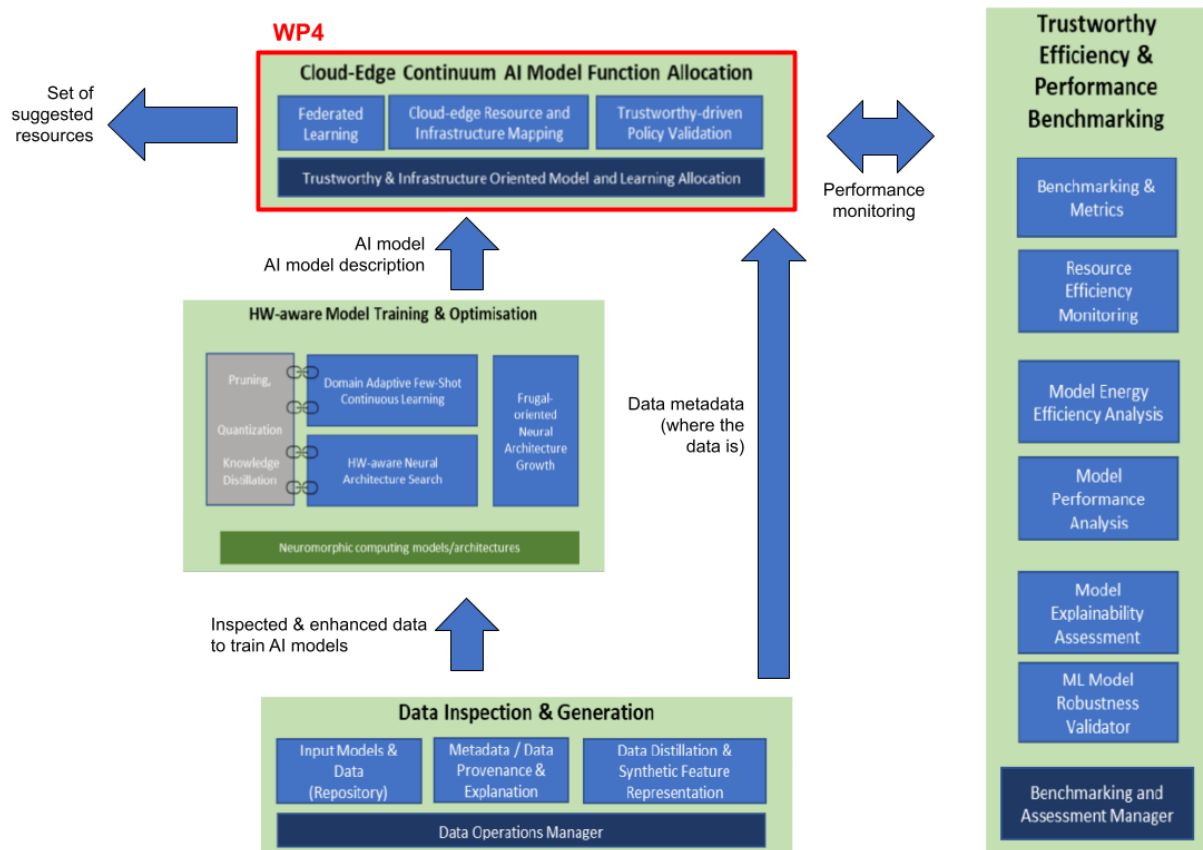
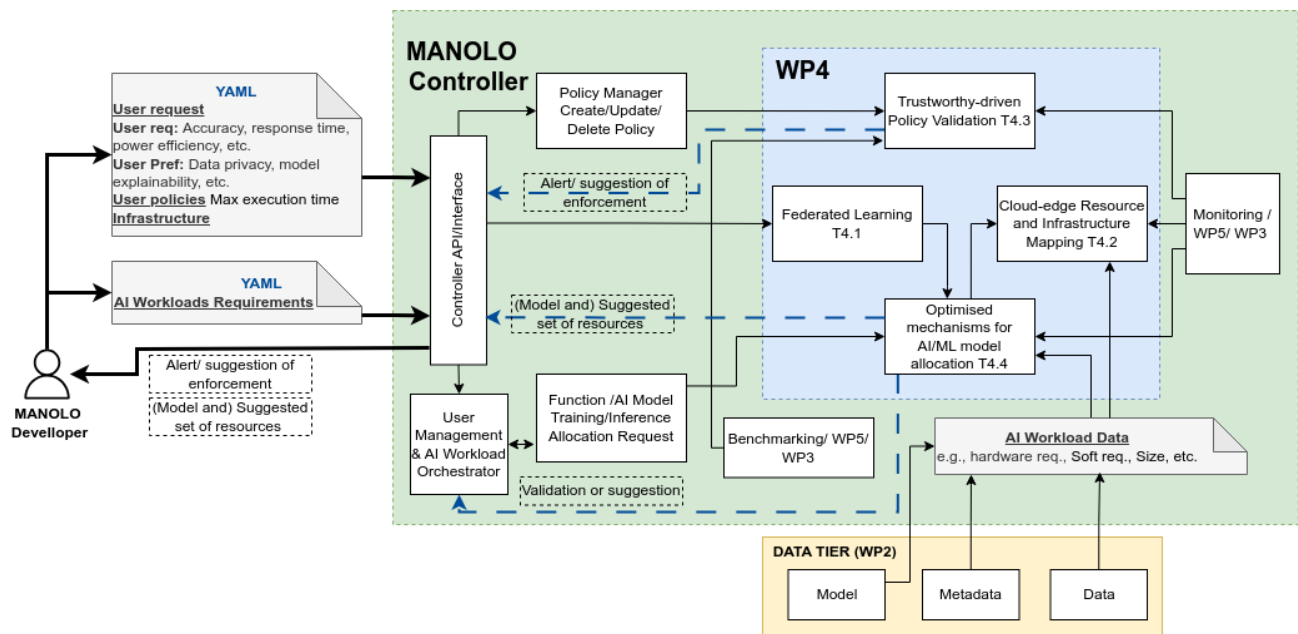


Figure 17. Cloud-Edge Continuum AI Model Function Allocation high-level architecture (highlighted in red).

As part of T4.3, the Trustworthy-driven Policy Validation component will be developed to manage the high-level requirements and policies which apply to the whole network and workloads; this is also contained in the same YAML file defined by the Admin User (top YAML in Figure 18). This will include for example overall constraints on energy consumption, performance thresholds, data handling practices, and algorithmic fairness and other data/network access policies and will all be registered as Policies within the system. These overall Policies will be enforced against the requirements of the Users' AI workloads (AI Workload Requirements YAML file) to ensure objectives, constraints, targets, or trustworthiness criteria are consistently met at a whole.

As the system needs to be dynamic to continuously adapt to the evolving needs of AI training, it will support Federated Learning functionality (Task 4.1) as an additional layer of efficiency of MANOLO. This will alleviate the burden of the learning process if required and allow for better generalisation of models and domain adaptation via distributed learning. More information about the WP4 internal architecture can be found in D4.1.



In the next four sections, we will zoom in on the components of the Cloud-Edge Continuum AI Model Function Allocation building block and describe the functionalities and workflows.

6.1.1 Federated learning

The component developed within Task 4.1 shall implement Federated Learning techniques targeting the minimization of sensitive data exposure, volume of data exchanged, and the reduction of energy consumption in distributed nodes during the execution of machine learning processes. Central to this system is a central server acting as the Aggregator API, initiating a round by sharing a global model across all client devices, henceforth referred to as Edge Devices APIs.

Each of the edge devices will have some local training on the model for its data without actually sharing that data with the central server. All the training in the local setup would just produce model updates, like weight gradients or parameter deltas, which, in turn, get sent back to the Aggregator API. On the server-side, these updates are aggregated by means of federated averaging or secure aggregation. After aggregation, the global model is then updated and redistributed again to the edge devices for another round of training. The process iteratively goes through several training rounds driven by the Aggregator API until the global model converges.

Aggregator API :

`/initiate_training`: Triggers the round of federated learning by sending the global model, along with its metadata, to all edge devices.

- `/aggregate_updates`: Collects model updates from edge devices, conducts secure aggregation techniques, and updates the global model.

`/distribute_global_model`: Sends the updated global model to all participating edge devices.

Edge Device APIs :

/receive_model: Receive the global model and conduct local training using device-specific data.

/send_model_update: Sends the model update which has been trained locally, including gradients or weight deltas, to the central aggregator.

/request_next_round: Indicate that itself is ready to take part in the next round for training, and also request the latest global model update from.

See Figure 19 for an illustration of this process. After the training finishes, the aggregated model will be sent to the “Trustworthy & Infrastructure Oriented Model and Learning Allocation” component described in Section 6.1.4.

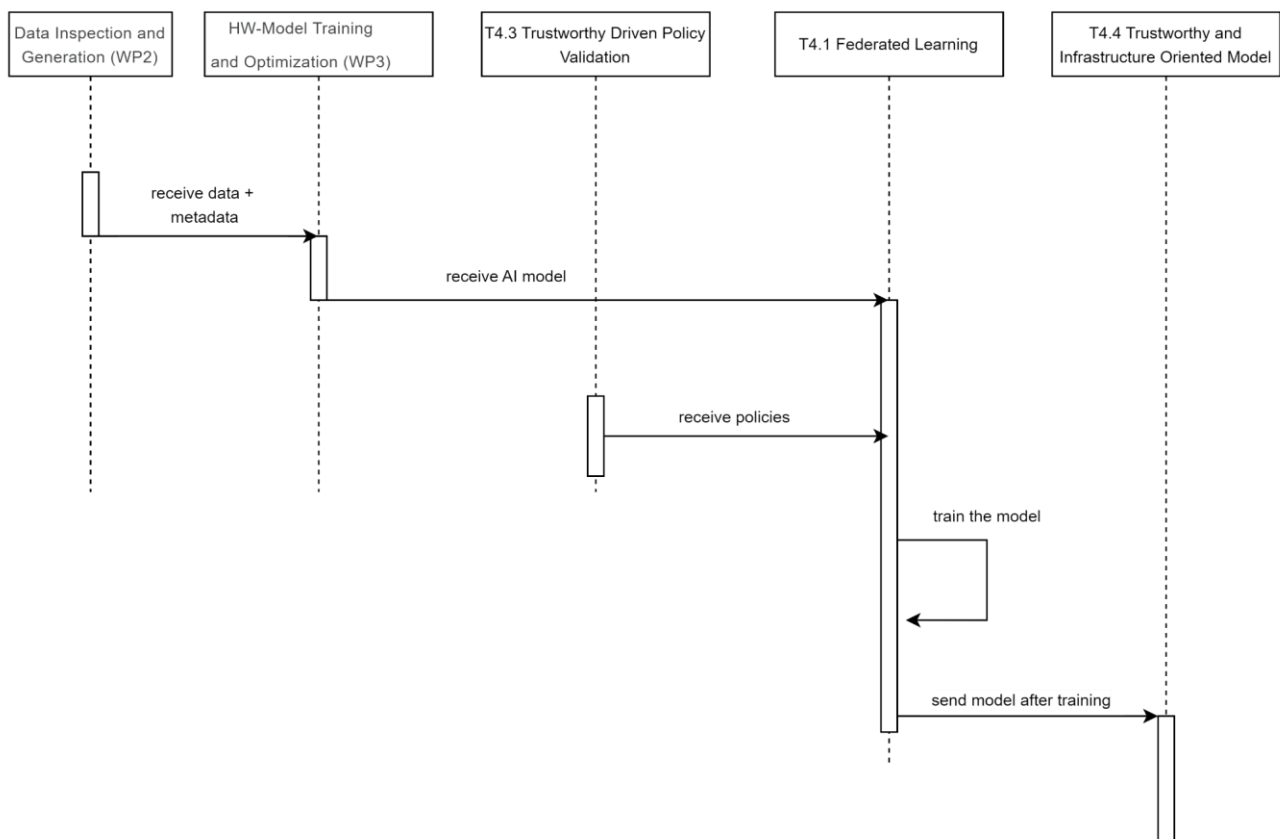


Figure 19. Federated learning functionality sequence diagram in MANOLO.

6.1.2 Cloud-edge Resource and Infrastructure Mapping

This component will map, abstract, and categorise computational resources and AI assets across the distributed heterogeneous cloud-edge system infrastructure. Particular attention will be given to the different types of computing resources (processing, storage, energy, capacity, etc.), networking topology, and associated AI functions and assets which characterise the overall infrastructure in the cloud-edge environment and AI requirements.

This information will be used to define specific mapping parameters that, together with the policy validator in Task 4.3, are going to be used in the development of mechanisms for recommending allocation of different models across the continuum in Task 4.4. The workflow in Figure 20 illustrates the interactions between the elements involved in the resource and infrastructure mapping component.

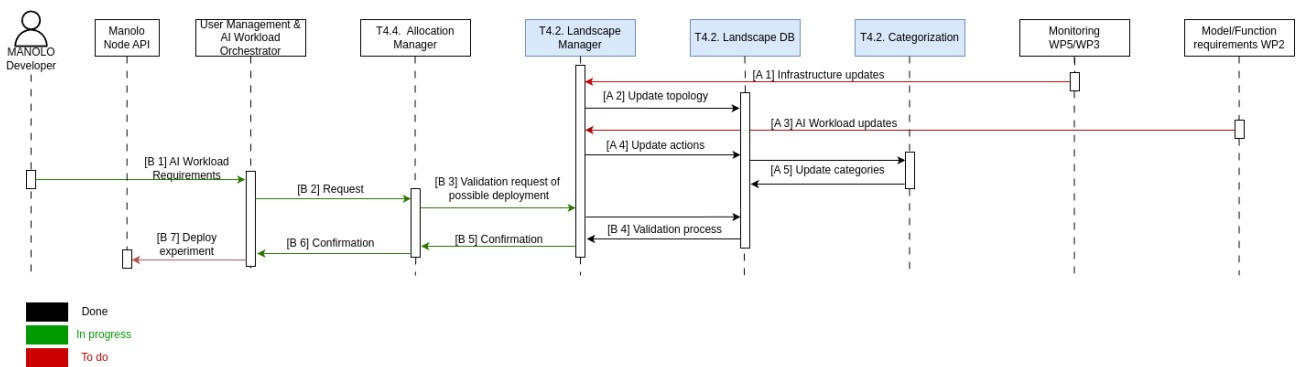


Figure 20. Cloud-edge Resource and Infrastructure Mapping component: Workflow.

The cloud-edge resource and infrastructure mapping component will be, according to Figure 20, composed of: i) a **landscape manager** responsible for interfacing with other MANOLO components, exposing APIs to receive AI requirements, benchmarks, and infrastructure updates. Furthermore, it will generate topology updates and store them in the Landscape Database, as well as trigger re-categorization (updates) of resources; ii) a **landscape database** storing all information about the topology, as well as the categories of resources based on AI model requirements and resource constraints; iii) **categorisation algorithms** responsible for classifying available resources into different classes based on resource constraints and AI model requirements.

In Figure 20 the entire Workflow of the cloud-edge resource and infrastructure mapping component is shown, together with the current state of integration. The internal communication of our tool in T4.2. is already fully integrated and done, while the communication with T4.4. is about to be finalized as well. T4.4. is working on establishing the connection to the AI Workload Orchestrator and over this to the MANOLO Developer

6.1.3 Trustworthy-driven Policy Validation

The Trustworthy-driven Policy Validation component is developed as part of Task 4.3 and implements the mechanisms that allow MANOLO stakeholders to monitor the compliance of operational requirements that AI/ML functions and models must meet. These requirements defined as policies to enforce during the runtime cover key aspects such as trustworthiness, performance, availability, energy efficiency, and infrastructure compatibility.

Policies are introduced into the system through developer-defined inputs and serve as a foundational mechanism to validate whether workloads behave as expected when deployed across the cloud-to-edge continuum orchestrated by MANOLO. Beyond their use in guiding runtime behaviour, these policies also support post-execution evaluation, helping developers and operators refine workload configurations and infrastructure choices based on historical compliance data.

The structure and ontology of these policies will be defined in collaboration with Task 4.2, where infrastructure capabilities, resource descriptions, and mappings are established. This ensures consistency between policy definitions and the actual resources available across the MANOLO environment. Policies are expressed using a declarative syntax (e.g., YAML), with clearly specified parameters such as model ID, dataset metadata, and rules like "maximum latency < 100 ms". This structured approach enables both automatic validation and traceability throughout the experiment lifecycle.

During experiment execution, policies are continuously evaluated by the Trustworthy-driven Policy Validation component, which operates within a broader observability framework built on Thanos. At the core of this system is the Policy Manager, which monitors compliance in real time by analyzing metrics collected from both workloads and the underlying infrastructure. This process is non-intrusive: the component does not enforce corrections but instead detects violations, triggers alerts, and provides developers with actionable feedback through a graphical user interface (GUI). The GUI supports both real-time monitoring and post-mortem analysis, giving full visibility into the status of policies, system behavior, and historical trends.

A typical example of a policy rule might be:

"If the average model confidence drops below 0.75 over a sliding window of 100 inferences, and CPU usage exceeds 80%, trigger a warning."

This rule-based approach allows MANOLO to support flexible, context-aware policy validation, which is essential for ensuring reliability and trust in distributed AI workloads.

The below in Figure 21 shows a Sequence Diagram about the main modules interact with the Trustworthy-driven Policy Validation.

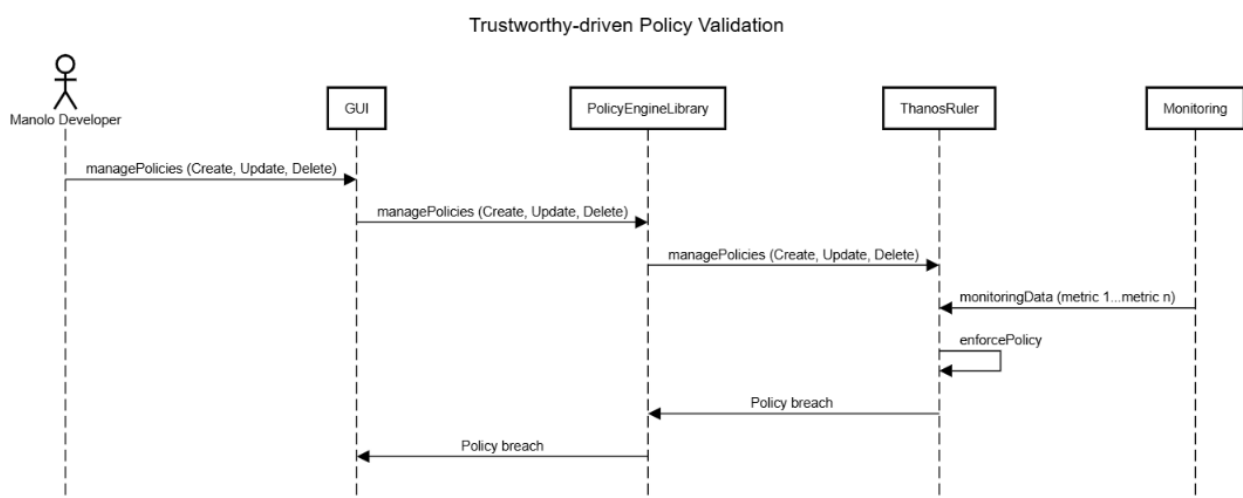


Figure 21. Trustworthy-driven Policy Validation Sequence Diagram.

6.1.4 Trustworthy & Infrastructure Oriented Model and Learning Allocation

This component is central within the main Cloud-Edge Continuum ML Model Function Allocation building block and is developed in Task 4.4. One of the main objectives is to design and develop optimised mechanisms for recommending the allocation of different AI/ML models and functions (training/inference) across the entire cloud-edge continuum. These mechanisms will take into consideration the specific requirements of the continuum infrastructure (heterogeneous and distributed computational resources with different capabilities and limitations) mapped in T4.2, and policy validation in T4.3, as well as providing functionality to enable federated training in T4.1 when and if required. It will have to model the set of parameters to be used to characterise resources in the continuum and associated policies, in order to recommend efficient and compliant allocation of ML models and functions. To this end, resources and models must be consistent, thus driving the proper mapping solution and so this task will be fed by the specific requirements defined in T4.1, T4.2 and T4.3.

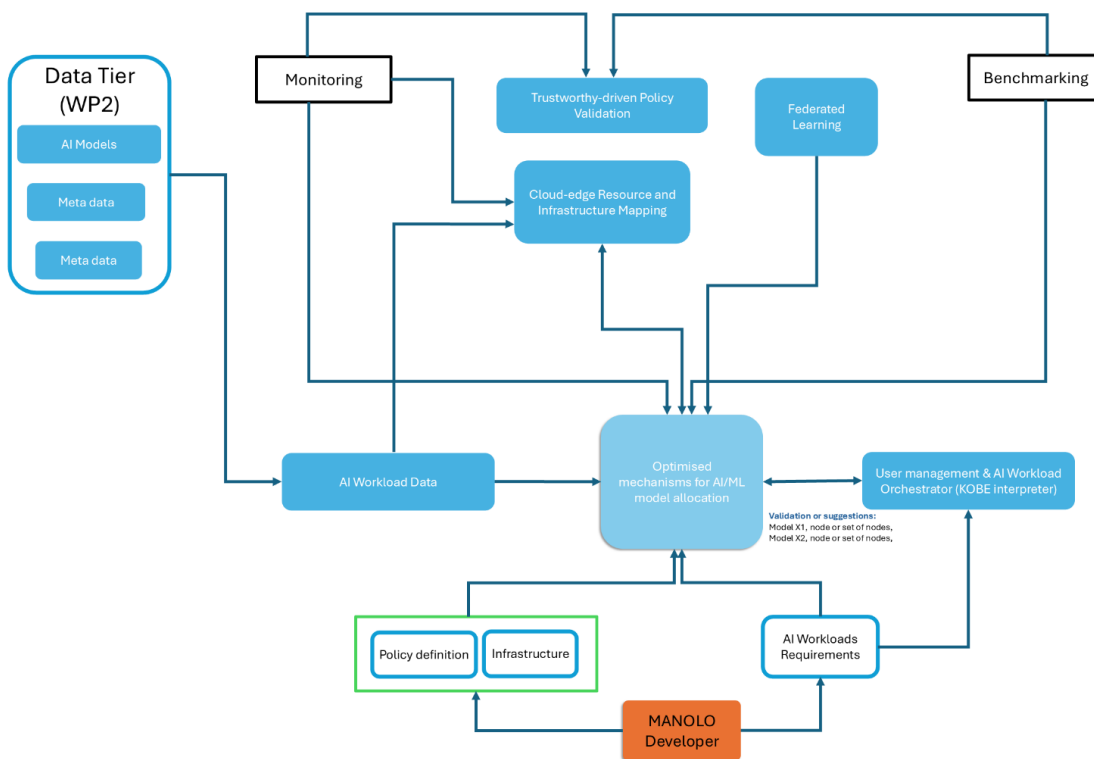


Figure 22. Trustworthy and infrastructure-oriented model and learning allocation component interactions

The interactions between MANOLO components and subcomponents of the learning and model allocation pipeline is shown in Figure 22, which introduces the following inputs for the Trustworthy and infrastructure oriented model and learning allocation component: set of user-defined requirements (AI Workloads Requirements), metadata from the data inspection and generation module described in Section 4, potentially the requirement of applying federated learning defined in Subsection 6.1.1, an AI/ML model from the training and optimization module described in Section 5, a landscape description from the cloud-edge resource and infrastructure mapping component described in Subsection 6.1.2, and the policies defined by the trustworthy-driven policy validation component described in Subsection 6.1.3.

The two main subcomponents in the learning and model allocation component are the manager and the optimal allocator. The manager receives a request from the user containing a set of requirements, along

with metadata, an AI model, resource and infrastructure mappings, and policy validations from the relevant subcomponents. Upon receiving this information, the manager activates the optimal allocator to provide the necessary resources. Once the optimal allocator is triggered, it performs an AI-based optimal allocation process. Considering the user's requirements and inputs from other subcomponents, it delivers the optimal model allocation solution by suggesting the appropriate set of resources. Finally, as outputs of the subcomponent, the pipeline outputs a set of suggested resources. This internal workflow is illustrated in Figure 23.

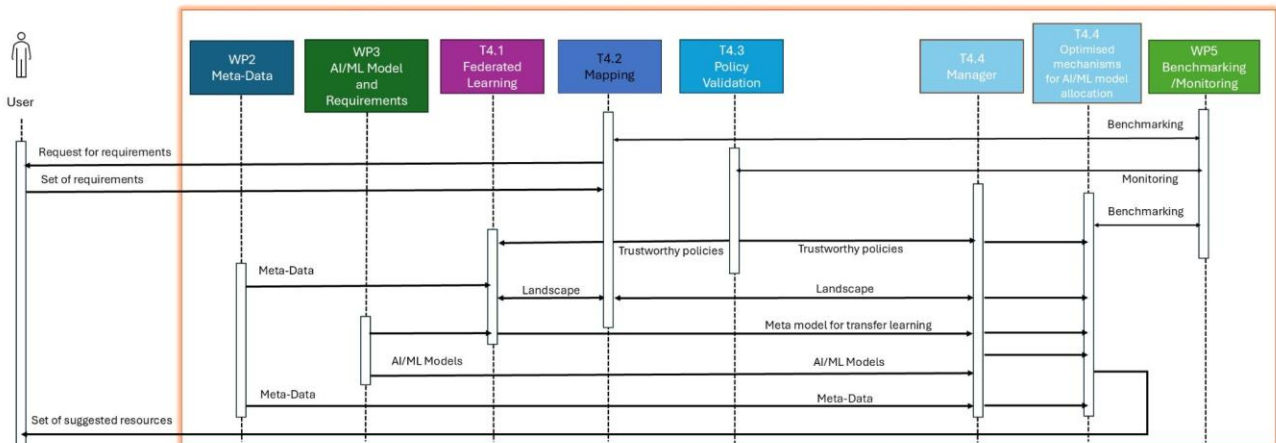


Figure 23. Trustworthy & Infrastructure Oriented Model and Learning Allocation component: Workflow

6.1.5 Workflow of the Cloud-Edge Continuum ML Model Function Allocation

The main functionality of the Cloud-Edge Continuum ML Model Function Allocation building block can be summarised in the workflow presented in Figure 24. The specific steps and more information about the workflow can be found in D4.1.

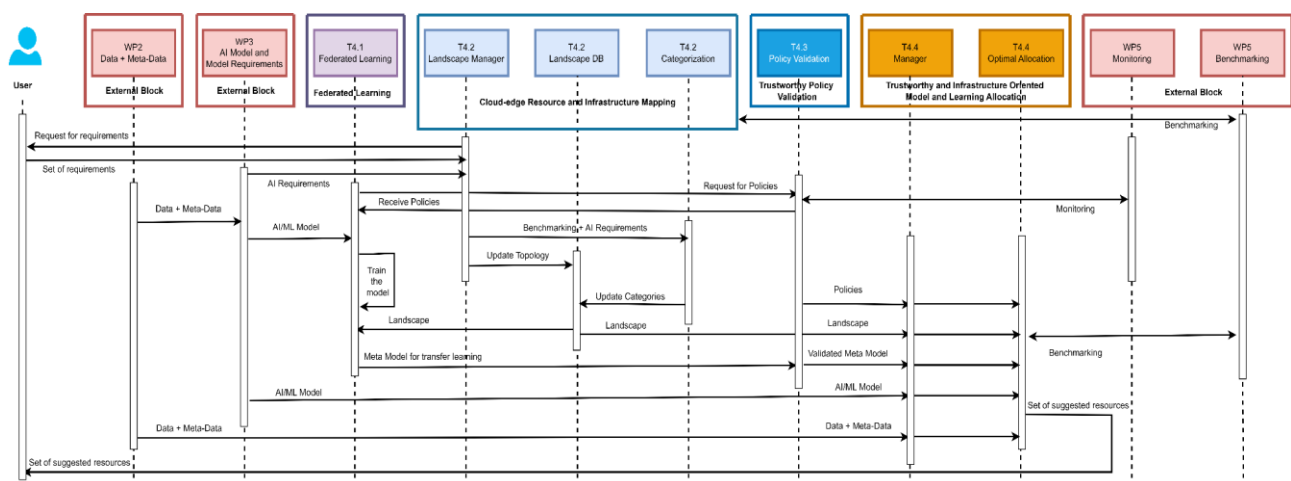


Figure 24. Cloud-Edge Continuum ML Model Function Allocation: Workflow.

6.2 Functional Requirements

Requirement ID	Description	Linked NFRs	Importance
FR_C6_01	Federated learning: MANOLO will provide federated learning functionality that will allow users to leverage a distributed training paradigm on multiple nodes	NFR02	Must have
FR_C6_02	Environment mapping: MANOLO will offer h/w and network usage metrics to allow monitoring and recommend AI/hw allocation.	NFR02, NFR04	Must have
FR_C6_03	Distributed processing: MANOLO will provide the functionalities to support the processing, storage, and preservation policies data and models across the different elements in the cloud-edge continuum.	NFR02, NFR03, NFR04, NFR06	Must have
FR_C6_04	Resource allocation: MANOLO will implement methods able to recommend task allocation across Cloud and Edge devices. Additionally, Standard Operating Procedures will be established for efficient resource management and dynamic allocation.	NFR02, NFR04, NFR05	Must have
FR_C6_05	Containerization: MANOLO will offer prepackaged containerization for different installation needs including reduced versions of the MANOLO library to resource constrained edge devices.	NFR02, NFR02	Must have

Table 14. Cloud-Edge Continuum ML Model Function Allocation: Functional requirements

6.3 Technologies

Table 15 shows an initial mapping of all the technologies that will be utilised for the component's development, the project task associated with the development and the responsible partners that contribute to the implementation of the components.

Component	Technology	Task	Responsible Partner
Federated Learning	Flower, TensorFlow Federated	4.1	FDI
Cloud-edge resource and infrastructure mapping	EdgeX Foundry, OpenNebula, FogFlow	4.2	TUBS
Infrastructure Oriented Model and Learning Allocation	Kubeflow, Yatai, BentoML, Docker compose, Docker Swarm, Kubernetes	4.4	UPC

Table 15. Initial mapping of proposed technologies.

6.4 Development Cycle

The Cloud-Edge Continuum ML Model Function Allocation building block developed in WP4 comprises four components that are closely dependent on each other, as the output of one becomes the input for the next. Therefore, the overall approach of this component is to ensure close collaboration with each component leader. Throughout this building block, the trustworthiness of the components (especially in trustworthy policy validation) will be iteratively assessed with Z-inspection®.

Step 1:

- Initial workflow design agreed upon by all task leaders.
- After system architecture design, an initial idea of the roles of each component has been outlined in M9

Step 2:

- With roles defined, the development of each component begins in parallel until M18
- Build the federated learning system to enable the local model training on distributed client's individual datasets M18.
- The Trustworthy Policy Validation system is developed to ensure compliance with privacy, security, and legal compliance, along with validating that federated learning and resource mapping adhere to these policies M24.
- Design and develop optimal mechanisms to allocate the AI/ML models across the cloud-edge continuum, ensuring that resources are allocated efficiently while considering both model requirements and policy constraints M24

Step 3:

- Monitoring and benchmarking for this component will be done in the last phase M30.
- Continuous monitoring to track resource usage, model performance, and policy compliance during and after allocation M30.
- Ensuring benchmarking performance by comparing the performance of AI/ML allocation for training and inference across the cloud continuum based on various metrics M30.

7. Trustworthy Efficiency and Performance Benchmarking (ATOS)

7.1 Component Description and Interactions

This section presents the different components conforming the Trustworthy Efficiency and Performance Benchmarking building block and framework as part of the MANOLO suite. Figure 25 depicts the high-level architecture of this framework (highlighted in red). Table 16 summarises the tasks within WP5 allocated to develop the main components integrating the MANOLO benchmarking framework.

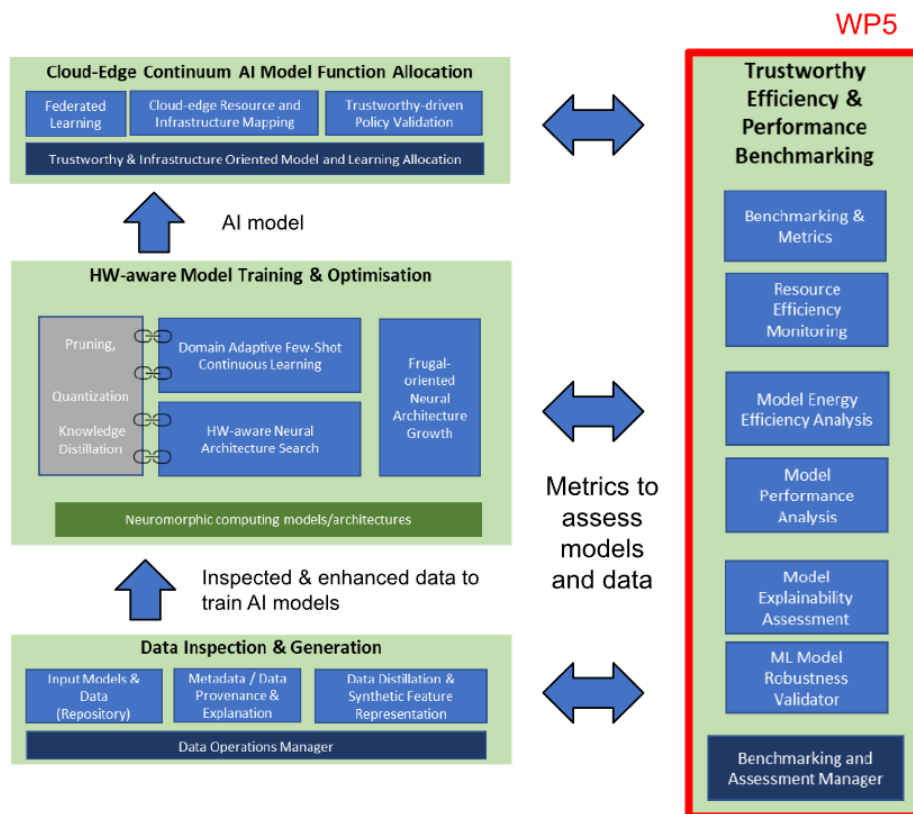


Figure 25. Trustworthy efficiency and performance benchmarking framework architecture (highlighted in red).

Task	Component	Main Effort / Task Lead	Planned integration in MANOLO
T5.1	Benchmarking Framework Platform	Demokritos	MANOLO suite
T5.2	Energy efficiency and Model Performance Monitoring and Analysis	ATOS	MANOLO Suite & open-source library
T5.3	Explainability and Robustness	FDI	open-source library

Table 16. Trustworthy Efficiency & Performance Benchmarking framework: Components, associated tasks, responsible partner and planned integration

Figure 26 presents the various components, the interactions between them, and initial suggestions for the components and tools to be used in each of the main blocks. The following sections provide an introductory explanation of the three components composing MANOLO's benchmarking framework, aggregated in functional blocks.

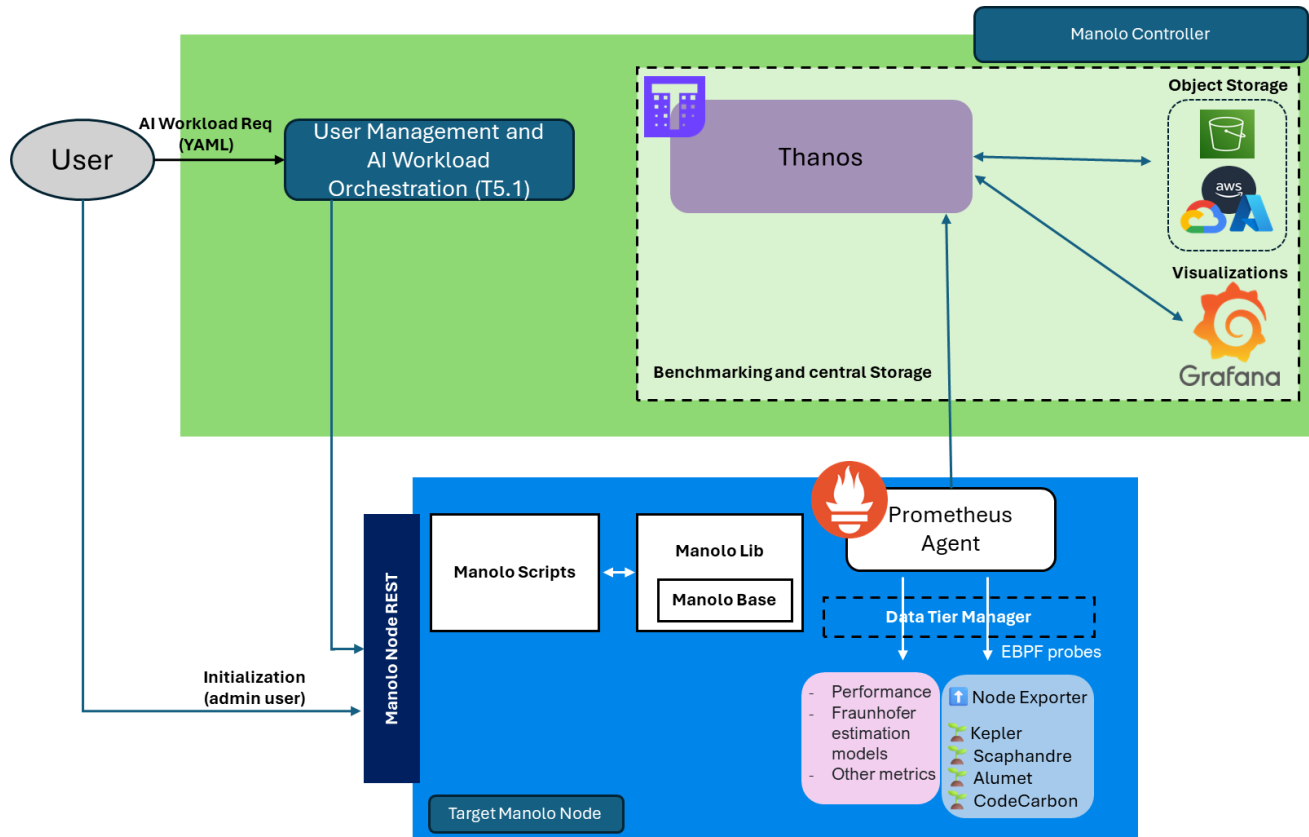


Figure 26. Benchmarking Framework: Main components and interactions (based on Figure 9)

7.1.1 Benchmarking Framework Platform

The MANOLO Benchmarking Framework (see Figure 26) is a platform designed to monitor, evaluate, and improve the performance, efficiency, and reliability of AI workloads running across distributed cloud-edge environments. It uses FastAPI to expose two REST APIs: the MANOLO Controller API, which is the main interface used by the User to configure and launch workloads, and the MANOLO Node API, which is called by the Controller to execute workloads at selected nodes. All interactions happen via standard HTTP protocols, with secure user authentication mechanisms in place. The Admin user manages access control policies for the Controller and sets up trusted communication channels between the Controller and the Nodes.

The User interacts with the MANOLO Controller API by submitting two YAML configuration files. The first is the Infrastructure & Policy Definition YAML, provided by the Admin, which defines the initial list of nodes in the network, their endpoints, hardware specifications, and high-level policies related to infrastructure, data, and models. The second is the AI Workloads Requirements YAML, which specifies the workload's execution logic, artifacts, and the list of target nodes. These are received by the User Management & AI Workload Orchestrator. This component collaborates with the Infrastructure & Policies module to validate and process

the configurations, store infrastructure metadata in the Data-Tier, detect inconsistencies from real-time network updates, and compare workload requirements with system policies and current infrastructure state. Based on this analysis and prior benchmarking data, it provides optimization recommendations to the Orchestrator.

Internally, the Orchestrator serves as the experiment coordinator within the MANOLO Benchmarking Framework. It begins by parsing the AI Workloads Requirements YAML to extract the workload structure, including models, datasets, execution steps (e.g., training, testing), and the metrics to be collected. It then verifies the feasibility of the workload against the infrastructure capabilities and policies defined in the Infrastructure & Policy Definition YAML. If the configuration is valid, it assigns a unique AIWorkflow ID, then orchestrates API calls to the MANOLO Node REST APIs to configure the selected nodes, upload scripts and artifacts, and initiate workload execution. On each node, the MANOLO Library is used to run the task under the assigned identifier. The Orchestrator dynamically loads the appropriate connectors—adapters that abstract away system-specific APIs for various ML frameworks—and verifies that all datasets are present and correctly formatted using the MANOLO-standard .npy and .json files. As workloads are executed, the Benchmarking Framework continuously monitors performance at the node level and aggregates this information at the Controller. It collects infrastructure metrics (CPU/GPU utilization, memory usage, disk I/O, latency), efficiency metrics (energy consumption, training/inference time), and accuracy/robustness indicators during development (e.g., precision, recall, ROC curves), as captured by the MANOLO Library. At deployment time, the framework tracks system uptime, error rates, and model degradation over time. It also identifies when models need retraining, managing model updates with new AIWorkflow IDs to handle challenges like data drift while ensuring continuous benchmarking traceability.

Data and metadata generated by each workload (e.g., models, logs, samples) are indexed using the AIWorkflowID + NodeID combination, and stored both locally (Node) and centrally (Controller) in SQL and Object databases, coordinated by the Data-Tier. Users can access benchmarking results and metadata via the Controller API, as well as through visual interfaces like MLFlow and Grafana. These tools allow for real-time inspection, alert management, and direct control over running workflows. Users receive recommendations for improving workload setups, making the Benchmarking Framework not just a passive monitoring layer, but an active optimization engine.

7.1.2 Energy Efficiency and Model Performance Monitoring and Analysis

To support the benchmarking process, a monitoring layer is deployed across the infrastructure to provide continuous insight into both system performance and energy efficiency. This layer not only feeds the benchmarking platform with real-time data but also helps estimate the environmental impact of the algorithms being tested. The monitoring setup builds on well-established open-source tools. Prometheus is used to collect performance metrics such as CPU usage, memory consumption, and network activity, typically gathered through Node Exporter, which runs on each node with minimal overhead. For long-term storage and scalability, Thanos extends Prometheus by enabling metric federation and integration with S3-compatible storage, allowing historical trend analysis over time.

In addition to performance monitoring, energy efficiency is a key focus. Several tools are integrated to provide visibility into power consumption and carbon emissions:

- Kepler collects energy metrics at the pod or container level using a combination of kernel telemetry, hardware counters, and machine learning. It's particularly well suited for Kubernetes environments.

- Scaphandre complements this by exposing power usage at the process or VM level and can be deployed alongside other exporters.
- Alumet adds an application-level view, correlating resource usage with energy data to help identify inefficient workloads and recommend improvements.
- For experiments involving AI or intensive computation, CodeCarbon tracks estimated carbon emissions, encouraging more sustainable development practices.

Beyond system metrics, Prometheus will also be configured to gather additional data sources (metrics) relevant to the benchmarking framework model. This includes performance-related outputs generated by the algorithms themselves, estimation models developed by Fraunhofer, and other task-specific metrics relevant to the evaluation of workload behaviour. These extended metrics will enrich the observability layer, allowing for a more nuanced and complete analysis of both functional performance and system-level impact.

All collected metrics—whether related to system performance, algorithm behaviour, or sustainability—are sent to a centralized backend via Prometheus and Thanos. This unified monitoring approach ensures consistent observability across all environments where testing takes place. It enables real-time analysis, historical comparisons, and informed decision-making not only around performance tuning but also around energy-aware optimization (see Figure 27).

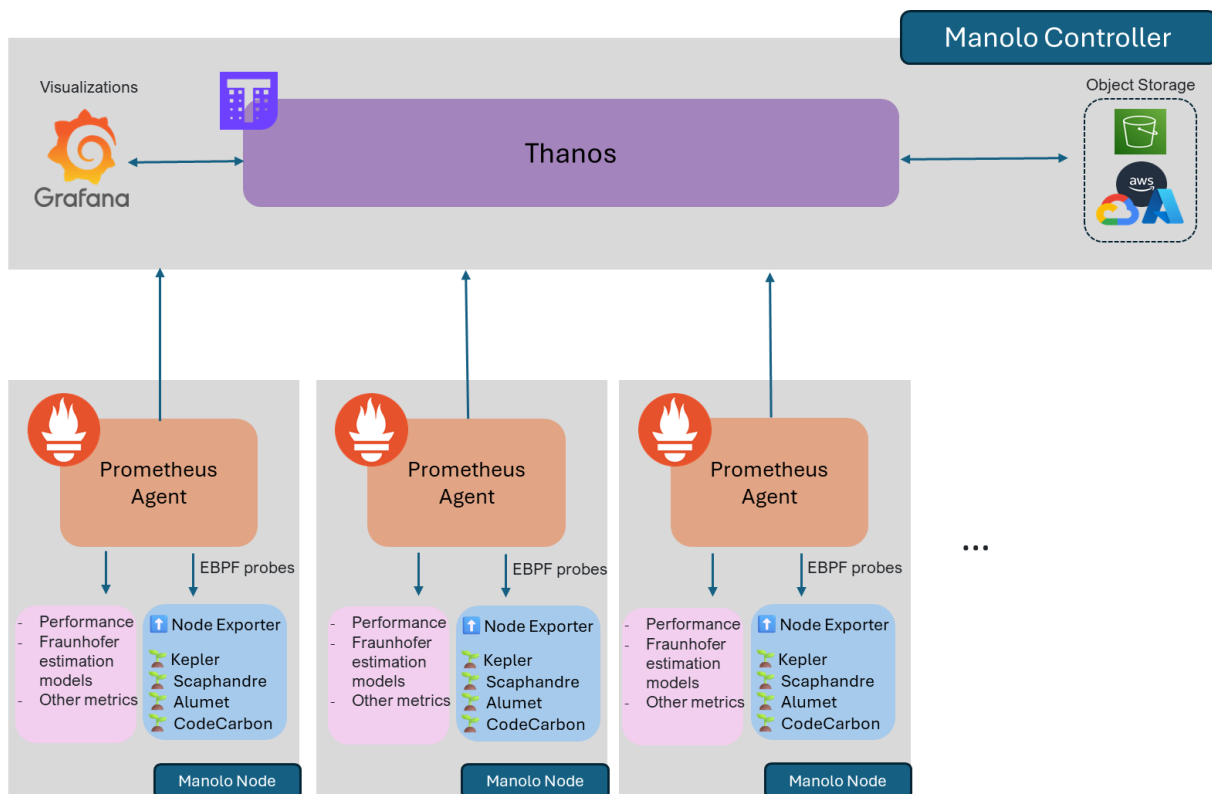


Figure 27. Energy Efficiency and Model Performance Monitoring and Analysis.

7.1.3 Explainability and Robustness Validation

Explainability is vital in ensuring AI models are transparent, interpretable, and trusted in production environments. By incorporating explainability techniques such as SHAP, LIME, or feature attribution methods, the system provides insights into the reasoning behind a model's predictions. This transparency lets stakeholders understand how key features contribute to predictions and make informed decisions, enhancing the system's trustworthiness.

Robustness complements explainability by ensuring the model's performance remains consistent under varying conditions, including data drift and unexpected changes. Through continuous validation and monitoring, any issues caused by shifts in data distributions or outliers can be detected early, minimising performance degradation. The combination of explainability and robustness ensures that models remain reliable, interpretable, and aligned with real-world requirements, especially in critical applications like those supported by MANOLO pipelines.

This dual approach of explainability and robustness strengthens the model lifecycle, allowing for both the detection of anomalies and the ability to explain their impact on model outputs.

Data Drift Monitoring System continuously analyses incoming data during model inference, comparing it to the original training dataset. This module helps to identify significant changes in data structure, features, or distributions, ensuring model robustness over time and protecting AI performance in production environments.

The Domain Adaptive Few-Shot Continuous Learning component (Task 3.2) will include statistical functions and predictions to estimate performance and identify issues in models over time. These functions will be used in this component in WP5 to ensure that the model metrics in MANOLO pipelines remain robust over time.

This component will aggregate the above functionality and will be incorporated in the MANOLO Core library.

7.2 Functional Requirements

Requirement ID	Description	Linked NFRs	Importance
FR_C7_01	Experiment orchestration: the MANOLO framework will orchestrate experiments in different virtual machines through KOBE	NFR9	Must have
FR_C7_02	Experiment resource consumption evaluation: MANOLO will probe the efficiency of AI experiments by using eBPF	NFR07, NFR08, NFR9	Must have
FR_C7_03	Model explainability: MANOLO will monitor and produce metrics portraying the explainability of AI models.	NFR07, NFR08, NFR9	Must have

FR_C7_04	Model robustness: MANOLO will monitor AI model robustness	NFR07, NFR08, NFR9	Should have
FR_C7_05	Data drift: MANOLO will provide continuous monitoring of data and model performance drift.	NFR07, NFR08, NFR9	Must have
FR_C7_06	Monitoring dashboards: the MANOLO suite will provide visualisation tools to facilitate human oversight and evaluation of the different benchmarks collected.	NFR07, NFR08, NFR9	Should have

Table 17. Trustworthy efficiency and performance benchmarking framework: Functional requirements.

7.3 Technologies

Table 18 shows an initial mapping of all the technologies used, as well as potential 3rd party technologies that will be utilised for the component's development, the project task associated with the development of the components, and the responsible partners that contribute to the implementation of the components.

Component / Module	Technology	Task	Responsible Partner
Data Drift Monitoring	NannyML, River, Frouros, Keras, Tensorflow, PyTorch	5.3	CeADAR, ATOS , FDI
Explainability	SHAP, LIME	5.3	FDI
Monitoring processes	Kepler	5.1	ATOS
Energy performance	Kepler models, PyTorch	5.1	ATOS, CeADAR
Data storage / Metric management	Prometheus/Thanos	5.1, 4.2	ATOS
Orchestration	Kubernetes, Docker Swarm, Docker compose	5.1	ATOS
Visualisation	Grafana	5.1	ATOS
Benchmarking	KOBE	5.1	NCSR

Table 18. Trustworthy efficiency and performance benchmark: Initial mapping of technologies per components.

Most importantly, CPU, memory, network and other type of metrics for monitoring will be automatically collected using OpenSource CNCF projects such as kube metrics and Prometheus node exporters. This will be key to leverage information for further processing in the benchmarking platform.

7.4 Development Cycle

The general approach for WP5 is to refine and extend the baseline benchmarking framework, which will serve as a foundation for improved resource monitoring and filtering processes. The development cycle evolves from this baseline, focusing on finding the right customization and configuration with parameters, as well as implementing additional tools on top of the framework. This will enhance resource monitoring and improve filtering processes to achieve an optimised benchmarking platform.

Regarding benchmarking, MANOLO will assume as a starting point NCSR's KOBE Open Benchmarking Engine, extending it to port existing KOBE functionalities to the MANOLO library and to integrate post-deployment monitoring functionalities developed in MANOLO.

Step 1:

- Final decision on functionality from KOBE Open Benchmarking tool to bring into MANOLO Benchmarking platform. Agree on set of metrics for benchmarking with the consortium.

Step 2:

- Select monitoring tools, set them up and test functional unit separately. Test integration between components using the same kind of ephemeral clusters. This will include monitoring workflows and first metric analysis.

Step 3:

- Test into a cluster (potentially Kubernetes) with internet connectivity in a lab environment with cloud and edge devices, analyze real-environment metric values and integrate all the tooling required.

Step 4:

- Configure multiple experiments via KOBE and test benchmarking platform with a variety of AI algorithms and hardware environments. Implement a human oversight mechanism over the gathered experiment/data.

Step 5:

- Complete integration with functionality of the MANOLO suite into local deployments. Implement cluster analysis and visualization tools to better explore and present the benchmarks to the user

8. Development Process, Integration, and Validation Environment (CeADAR & UPC)

This section introduces the preliminary approach for system integration and stress testing of the MANOLO framework in lab environments. This work is conducted in Task 5.4. This section provides preliminary guidance for the technical partners on:

- The initial technology choices for implementing different system components.
- A strategic approach to component integration and testing.
- Open-source license

8.1 Development and Integration Technologies

Due to the number of diverse components that will be developed over the course of the MANOLO project, their integration into a singular system needs to be accomplished in a structured and efficient manner. Most parts of MANOLO will be open-sourced (see Section 8.4 for open-source license). An interface to access proprietary technology/components (i.e., Neural architecture search module and the Neuromorphic module) will be included in the MANOLO open-source library and framework, but the usage of these modules will not be open-source and would rely on a web-based service.

Table 19 shows an initial mapping of technologies to be used for integration of MANOLO, which includes frameworks for containerisation, orchestration of containers and experiments and Continuous Integration and Continuous Delivery (CI/CD).

Activity	Technology
Containerisation & MLOps	Docker, BentoML, MLFlow, DVC
Experiment Orchestration (suggested)	KOBE, Yatai [4], Kubernetes (using helm charts to load docker containers), Docker Swarm, Kubeflow
CI / CD	GitHub actions

Table 19. Mapping of technologies for overall orchestration and integration of MANOLO framework.

8.2 Development Cycle and Integration Process

MANOLO aims for continuous Software development during the whole project lifetime with named milestone software releases. With this in mind, the integration is based on Continuous Integration and Continuous Delivery principles using GitHub Actions.

This project will provide incremental updates in the MANOLO repository. Each technical WP will detail releases for the work performed in their deliverables and make them available at the main branch of their sub-repositories when ready. Specific development cycles and methodologies for each WP have been detailed in their respective sections in this deliverable.

As shown in Figure 28, contributors will fork and pull the required MANOLO components/modules to their local machines, where they will develop new features or work on existing parts of the library.

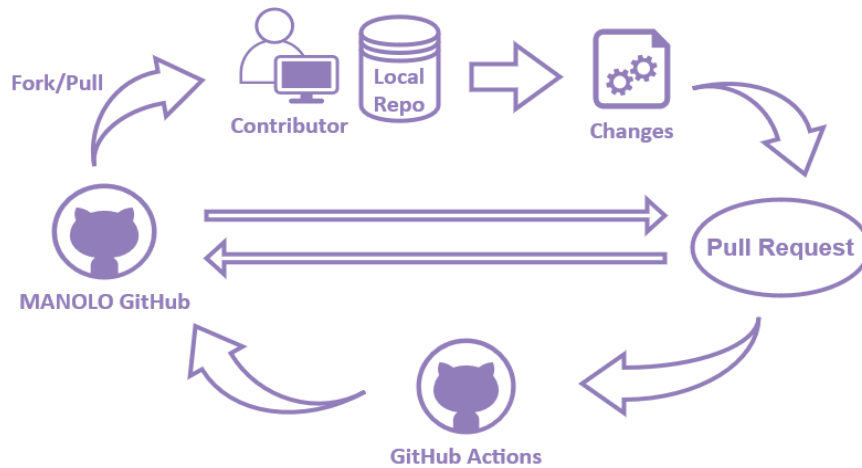


Figure 28. MANOLO integration process.

Once these changes are finished, the contributor will generate a pull request that will trigger the GitHub Actions workflow. This workflow consists of a battery of tests for loading models, inference in small models, mock runs of training pipelines, etc., to ensure that the changes do not disrupt existing functionalities. After these tests are passed, the code is integrated into the MANOLO repository and becomes part of the library. Developers of each module and component are responsible to prepare the appropriate tests which will be incorporated into the GitHub Actions workflow.

8.3 Testing and Validation Environments

Task 5.4 is responsible for organising the overall integration of MANOLO and testing of the library and suite in lab premises of multiple partners in the consortium with access to different cloud-edge environments, thus driving the MANOLO release. This task will produce a plan of integration and stress-testing procedures at in-lab premises of the selected partners. The set of stress testing will be decided later and reported in a next deliverable about integration, but they will be including a variety of software and ML testing via a set of experiments, including aspects such as communication among modules, scalability, and overall performance.

Initially, the infrastructure environment provided by technical consortium partners in their labs will be used to run experiments and test the system. The infrastructure will accommodate a wide range of computing and networking needs. A larger number of consortium partners involved in the development of MANOLO, including use case partners, will finally validate the library in their premises with different technologies, such as HPCs, servers and edge devices, and the validation results will be aggregated.

8.4 Licensing

For easier adoption and adaptability to the users' needs, the majority of the software components in the MANOLO suite and library will be open source under an Apache 2.0 license. This license permits the use of the Software for any purpose: distribution, modification, and distribution of modified versions. By adopting the 2.0 version of the Apache License, we aim to leverage collaborative development to make the MANOLO suite and library a long-lived open-source software.

The following is an initial suggestion for the license file to be included within the MANOLO suite:

```
Copyright 2025 MANOLO project (https://manolo-project.eu/)
```

```
This work has received funding from the European Union's HORIZON research and  
innovation programme under grant agreement No. 101135782.
```

```
Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at
```

```
http://www.apache.org/licenses/LICENSE-2.0
```

```
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.
```

All repositories should also include the following in their README file:

This work has received funding from the European Union's HORIZON research and innovation programme under grant agreement No. 101135782.

As described in the MANOLO Grant Agreement and in D7.4 Exploitation and Sustainability Plan v1, certain Key Exploitable Results (KER) will be protected through Copyright under a suitable Commons license. The Neural Architecture Search component and the Neuromorphic component will be protected by the standard Fraunhofer licensing terms. More details about this will be provided in the D7.5 Exploitation and Sustainability Plan v2.

9. Conclusions

This document outlines the second version of the architecture of the MANOLO system, which is a joint deliverable between the tasks “T1.3: Trustworthy Efficiency-Performance Assessment Framework co-design” and “T1.4: System Requirements and Architecture co-definition”.

All partners in the project collaborated to co-define the architecture using a structured methodology. This process began setting non-functional requirements at a MANOLO high level and then specific functional requirements for them, with a strong link to trustworthiness and efficiency in AI systems. Each component-specific section in the deliverable has gone into detail about specific component requirements and their degree of importance using a methodology based on the MoSCOW method. The collaboration involved various co-creation activities with multi-disciplinary teams, such as virtual and in-person meetings, brainstorming sessions, and the proposal and review of models and diagrams.

Functional requirements have been then converted into MANOLO system usage scenarios detailing the interactions between the different MANOLO components and detailing the core functionalities and value that MANOLO offers to its end-users, both as individual components and as the MANOLO integrated framework.

MANOLO, both in terms of development and usability, can be structured in two main internal parts, **library** and **suite**, as well as other external technologies. The library contains the base functionalities for AI model optimisation and trustworthiness, while the suite offers a containerised environment with extra functionality to optimise the execution of models in the CEC.

Besides these two representative implementations, MANOLO components are conceptually classified according to the following building blocks:

- **Data Inspection & Generation:** focusing on management, distribution, collection, provenance, compression, and generation of data and contextual information.
- **HW-aware Model Training and Optimization:** including algorithms designed to enhance models for a later efficient deployment across the CEC.
- **Cloud-Edge Continuum ML Model Function Allocation:** featuring modules for resource allocation on specific model inference and training requirements.
- **Trustworthy Efficiency and Performance Benchmarking:** creating a benchmarking system tailored for the CEC and evaluating AI systems with respect to efficiency, performance, and trustworthiness.
- **Human Oversight Trustworthy Assessment:** providing processes and tools for co-designing, monitoring, assessing, and ensuring that the developed AI system complies with ethical and legal standards and meets societal expectations.
- **Integration and Validation Environment:** integrating and deploying the full MANOLO suite within existing CEC infrastructure.

In this document, we have presented the i) the technological stack to be used for the implementation, integration, and testing of the system, ii) the partners responsible for each component, and iii) the integration process that will connect the individual components into the releases of a unified MANOLO system. We do this to organise the technical work in WP2, WP3, WP4, and WP5; the core functionalities of the MANOLO system will better support the design and planning of the use cases’ execution and evaluation.

The architecture in this deliverable is serving as the reference foundation for technical work in the project development within each of the technical work packages. A first validation has taken place and further co-refinement of the architecture has followed in the form of hybrid and physical workshops that involved the MANOLO consortium and presentation to the members of the Ethics and Industrial Advisory Board.

As a final remark, we expect the architecture to dynamically evolve with the project, as implementation and validation will feed back into the design phase. Additional feedback from the evaluation of the system in the project's use cases (WP6) will also contribute to its enhancement.

Appendices

Appendix 1: Z-Inspection® Trustworthy AI Co-design process

In Z-Inspection®, socio-technical usage scenarios are used by a team of experts, to identify a list of potential ethical, technical, and legal issues that need to be further deliberated. Scenarios are used in MANOLO as a participatory design tool in the design phase.

The documentation of the Trustworthy AI co-design using the Z-Inspection® process is made accessible to the users in MANOLO. A template (parts included below) has been designed to assist MANOLO partners in creating and analysing Socio-Technical Scenarios for various AI Systems that will be designed, developed, and used within the MANOLO Project.

With “AI Systems” we include any AI product, service, modules, models, and use cases of the MANOLO project. In Figure 29 and the step list below, we show the steps partners should follow in order to establish socio-technical scenarios.

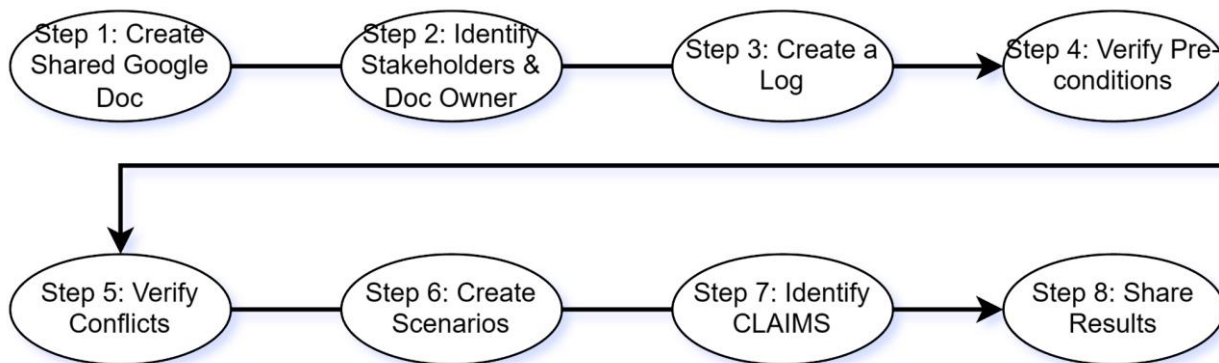


Figure 29. MANOLO Template steps for creating socio-technical scenarios.

Step 1. Create a shared Google doc, with the name of the AI System (say XX), using this standard name: XX.Socio-Technical Scenarios.

Step 2. Identify the *stakeholders* who will have access to the shared Google Doc (or other shared document) and be involved in the creation and analysis of the socio-technical scenarios. Among these stakeholders, identify the “owner(s)” of the shared document who will grant access to the document to other stakeholders.

Step 3. Create a *Log* in the shared google doc which lists all meetings online and in presence, and more. Use this template below:

Begin Log.....

Workshop(s): date(s), Location(s)

For each workshop list of the names, affiliation and roles and expertise of each participant

Links to any recordings (YouTube) of the meetings (if any) private (or if you decide public) to the members of MANOLO *to be placed at the end of the document of the socio-technical scenarios.*

Links to copy of the chat for online meetings *to be placed at the end of the document of the socio-technical scenarios.*

For each workshop List of the *Progress achieved* (see Step.5)

List of possible *Conflict of interests* you and/or some external members (e.g. members of the Ethical Advisory Board) discovered (see Step.5)

Actions taken when you have discovered a possible conflict of interests. (see Step. 5)

.....end Log

Step 4. Verify Pre-conditions

We developed a catalogue of questions that we recommend you should answer before starting the co-design process.

1. Who requested the co-design process?
2. Why carry out a co-design process?
3. For whom is the co-design process relevant?
4. Is it recommended or required (mandatory)?
5. What are the sufficient vs. necessary conditions that need to be analysed?
6. How are the co-design process results to be used?
7. Will the results be shared (public) or kept private? In the latter case, the key question is why is it being kept private?
8. How do you intend to handle possible conflict of interests? (see Step. 5)

Step 5. Verify possible conflict of interests.

One of the challenges in co-design through a self-assessment process is managing potential conflicts of interest, both direct and indirect. To address this, we recommend that an external, neutral stakeholder verify the absence of conflicts of interest before the co-design process begins and continue monitoring throughout.

During the co-design process potential conflicts of interests may arise, and therefore this necessitates constant monitoring during the entire duration of the process.

A protocol (refer to Step 3. log) should be established and maintained for the entire duration of the co-design process to document the progress over time.

Here are some pointers to consider:

- Partners should encourage team members to identify and share interests that might influence their input toward the project's goals and keep a record of these for clarity and reference.
- Partners should foster an environment where team members can openly share thoughts and concerns, ensuring all discussions are shared with the group to maintain transparency.
- To ensure accountability, create attributional records of who said what and when during discussions to track contributions and decisions, and continuously monitor the project's progression and stakeholder interactions to address any issues promptly.
- Promote decisions aligned with all partners and team members, using input from a diverse group of stakeholders (managerial/ethical/engineering/business). Document how decisions were reached and how different perspectives were considered to ensure well-informed and

transparent decision-making. Monitor changes to decisions and document these as the project moves forward.

In the Log, it is important to record **the actions taken upon identifying potential interests**. The transparency of this process, ensuring accessibility at least to members of the Ethical Advisory Board and EU Officers, and subsequently to reviewers, **is a crucial aspect of the co-design process**.

Step 6. Create Socio Technical Scenarios.

Socio-technical scenarios are created by using *a predefined set of questions*.

Step 7. Identify CLAIMS

Once socio-technical scenarios are created, the next step is to identify any sentence that sounds like a *Claim*, for which *Arguments* (pros and/cons) will be listed, and some *Evidence* supporting each argument will have to be provided.

These steps are part of the MANOLO early AI co-design phase which helps produce a (Re)Definition of the AI design's goal/purpose taking into account the Trustworthy AI requirements. (see Figure 30).

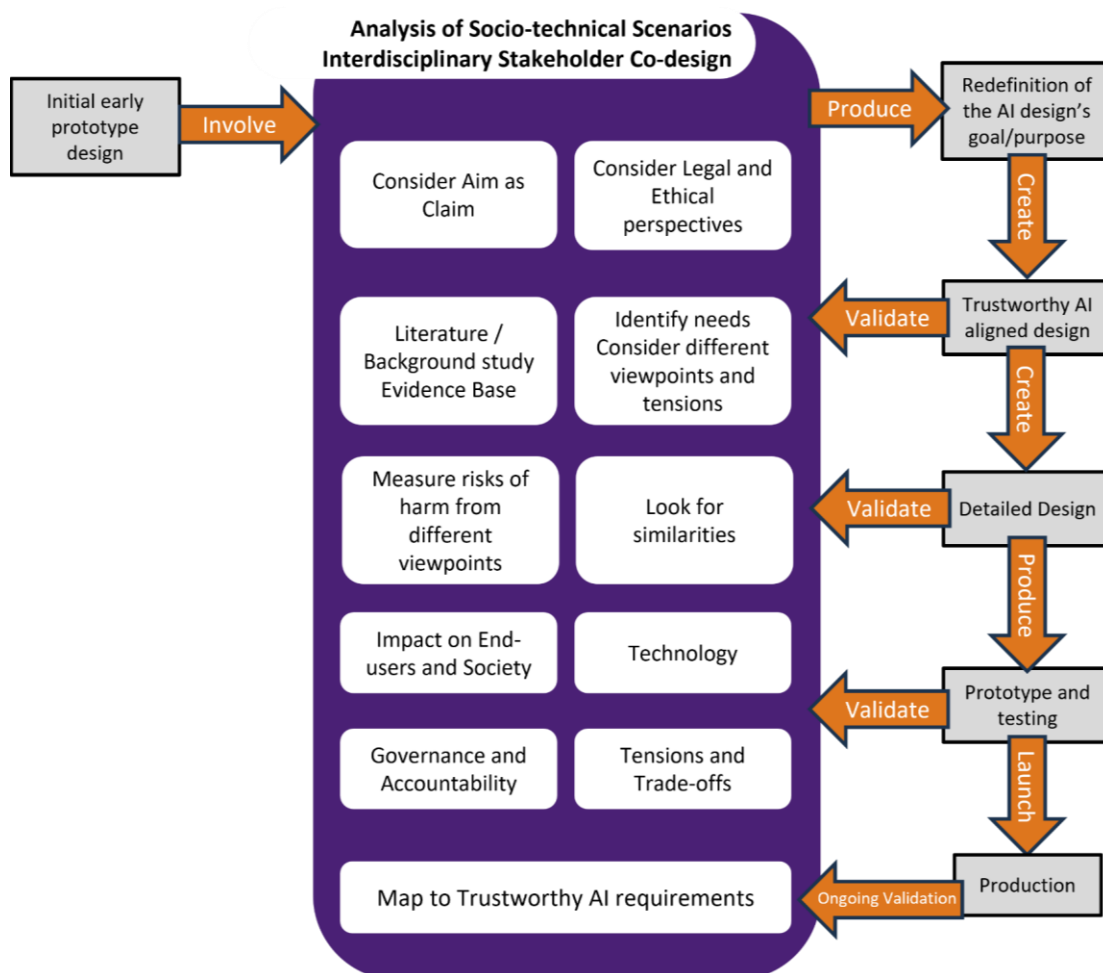


Figure 30. Trustworthy AI system Co-design.



MANOLO

CLOUD-EDGE EFFICIENT & TRUSTWORTHY AI

GA 101135782

Partners



EVIDEN



Fraunhofer IIS



KU LEUVEN



arx.net
MOBILE INNOVATIONS



@manolo-project

