Industrial Machine Learning for Enterprises

Deliverable D4.4

Final MLOps methodology and the IML4E framework

Project title:	IML4E
Project number:	20219
Call identifier:	ITEA AI 2020
Challenge:	Safety & Security

Work package:	WP4
Deliverable number:	D4.4
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	30.09.2024
Actual delivery date:	04.09.2024
Responsible partner:	Fraunhofer FOKUS

Contributors

Editor(s)	Jürgen Großmann (Fraunhofer FOKUS),
Contributor(s)	Jürgen Großmann, Dorian Knoblauch, Marek Feldo; Martin Große-Rhode, Abhishek Shrestha (Fraunhofer FOKUS), Johan Himberg (Reaktor), Mikko Raitikainen (University of Helsinki), Harry Souris (Silo AI)
Quality assuror(s)	Ihasalo Heikki (Granlund), Abhishek Shresta (Fraunhofer FOKUS), Csaba Kiss and Gábor Gulyás (Vitarex), Balázs Morvay (BUTE)

Version history

Version	Date	Description
1.0	04-09-24	Final version for publication

Abstract

The purpose of this document is to specify the final version of the IML4E MLOPs framework. It consists of the IML4E methodology that aims to offer governance and auditability to ML products. In addition, purpose of the document is to describe a software platform that can support and accelerate the development and operations of ML applications.

Keywords

MLOps, MLOps framework Governance, Audit, Monitoring, Deployment, ML/AI Architecture, MLOPs technology stack

Executive Summary

This document introduces the final version of the IML4E framework. IML4E framework describes methodologies that will govern the lifecycle of ML products in addition to the technologies and tools that will support the execution of ML pipelines. The IML4E framework aims to bring an end-to-end approach on working with ML in enterprises.

Table of contents

1 Introduction

1.1 Role of this Document

This document discusses the IML4E framework and methodologies (Section 3). The IML4E methodologies aim to provide a systematic process when developing and operating ML pipelines.

1.2 Intended Audience

The intended audience of the present document is composed primarily of the IML4E consortium for the purpose of capturing the baseline of the project that the project will advance. However, this document is public and can provide an overview of the current practices to a reader. This document describes technologies for the technically oriented audience rather than the general public or layman.

1.3 Applicable Documents

Table 1: Contractual documents

Reference	Referred document
[FPP]	IML4E – Full Project Proposal 20219
[PCA]	IML4E Project Consortium Agreement

2 Introduction to MLOps

Machine Learning Operations (MLOps) is an emerging discipline that integrates machine learning (ML) with DevOps principles to streamline the deployment, monitoring, and management of ML models in production environments. As organizations increasingly leverage ML to gain insights and drive decision-making, the need for robust, scalable, and efficient operational frameworks becomes critical. MLOps addresses this need by providing a systematic approach to managing machine learning models' lifecycle, from development and training through deployment and serving to maintenance and updates.

The core objective of MLOps is to bridge the gap between data science and IT operations, ensuring that ML models can be deployed reliably and monitored effectively. MLOps involves a range of practices and tools designed to automate and standardize processes, facilitate collaboration, and ensure compliance with organizational policies and regulations. Key components of MLOps include continuous integration and continuous delivery (CI/CD) pipelines for ML, automated testing, model monitoring, and performance optimization.

By implementing MLOps, organizations can achieve several benefits, including faster time-to-market for ML solutions, improved model performance and reliability, and enhanced collaboration between data scientists, engineers, and other stakeholders. Furthermore, MLOps helps manage the complexities associated with deploying ML models at scale, such as handling large datasets, ensuring reproducibility, and maintaining model accuracy over time.

As machine learning continues to evolve and become integral to various industries, MLOps will play a pivotal role in ensuring that ML models are efficient but also maintainable and scalable in production environments. This introduction provides an overview of the fundamental concepts of MLOps, highlighting its importance and the key practices that underpin successful ML operations.

3 The IML4E Framework

Machine learning (ML) and the use of software-based systems whose functionality is at least partially determined by ML are also becoming increasingly important in the European industry. Machine learning is being used to provide smart services and is increasingly becoming the basis for safety-critical functions. The use of machine learning (ML) as integral part of industrial application development has a massive impact on the entire software lifecycle and related quality assurance activities.

The IML4E framework is a collection of principles, methods, and technologies designed to simplify the adoption of MLOps in enterprises. It consists of a *technology layer*, called the "IML4E MLOps tools and techniques", a *methodology lay*er called "MLOps methods and principles", and a *platform layer* called the "IML4E OSS platform and reference architecture". All layers are supplemented by "IML4E Teaching material and playbooks". The layers of the IML4E framework are shown in Figure 1.

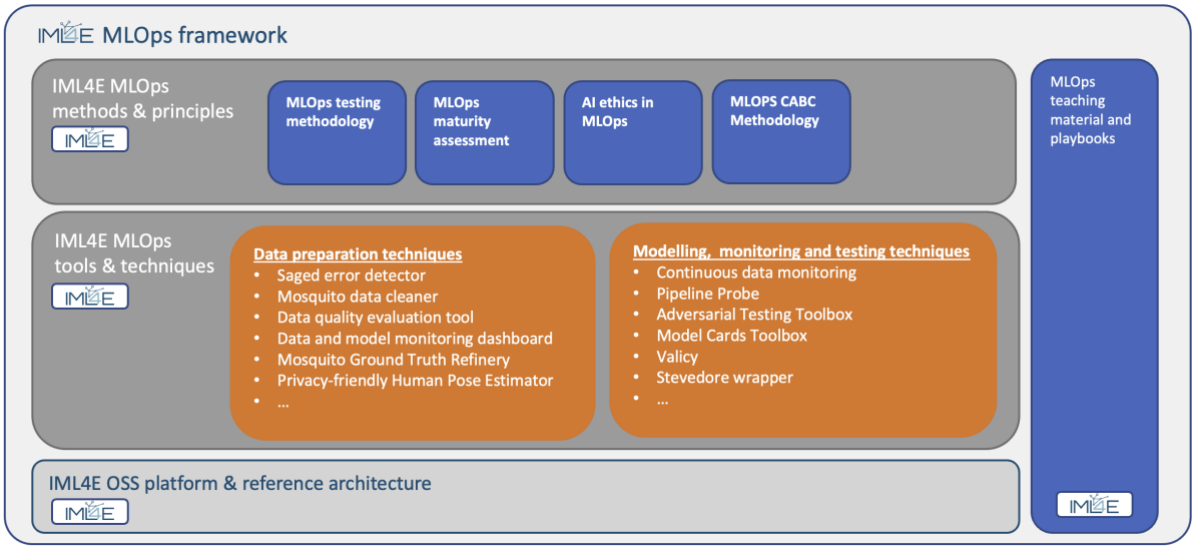

Figure 1 - The IML4E framework

While the methodology layer provides foundational support for setting up and managing MLOps in an organization, the technology layer provides individual technical solutions for concrete subproblems in MLOps. In addition, all solutions are demonstratable by integration in the IML4E OSS platform. The IML4E OSS platform is an implementation of the MLOps reference architecture and builds the operational basis for setting up an MLOps infrastructure.

3.1 IML4E MLOps principles and methods

The IML4E MLOps principles and methods are developed by individual partners as part of the work package 4. They are meant to provide guidance on implementing MLOps and MLOps related workflows in organizations. Among others IML4E has developed the methods given in Table 2.

Table 2: IML4E principles and methods overview

Name	Provider(s)	Topic(s) Covered	WP
The MLOps maturity assessment framework	Fraunhofer FOKUS	MLOps	WP4
The IML4E MLOps testing methodology	Fraunhofer FOKUS	Testing ML-based systems, MLOps	WP4
The IML4E CABC methodology	Fraunhofer FOKUS	Continuous-audit based quality assessment in MLOps	WP4

AI Ethics in MLOps	University of Helsinki	AI ethics in MLOps	WP4

3.2 IML4E MLOps tools and techniques

The IML4E technologies are developed in the IML4E project by different partners in the work packages 2 and 3. They provide self-contained solutions covering topics in Data Preparation, Data Versioning, Model Documentation, Model and Data Testing, Monitoring etc. An overview on these technologies is given by Table 3.

Table 3: IML4E tools and techniques overview

Name	Provider(s)	Topic(s) Covered	WP
Automated Data Cleaning for Tabular Data in ML Pipelines	Software AG	Data Preparation Automation	WP2
Data and model monitoring dashboard	Granlund	Data Preparation Automation	WP2
Mosquito Ground Truth Refinery and Quality Feedback Service	Basware Oy	Data Preparation Automation	WP2
Privacy-friendly Human Pose Estimator	Budapest University of Technology and Economics	Data Preparation Automation	WP2
Data Quality Evaluation Tool	Fraunhofer FOKUS	Data Quality, Automated Testing, CABC	WP2
Model cards toolbox	University of Helsinki	Model engineering, model maintenance	WP3
Stevedore wrapper class and generic API and Podman/Docker build automation for Python ML models	Reaktor	Model engineering	WP3
Pipeline Probe (former CABC-Mapper)	Fraunhofer FOKUS	Model training, MLOps lifecycle	WP3
Adversarial Test Toolbox	Fraunhofer FOKUS	Model training, model testing	WP3
VALICY – a tool for virtual validation of AI & complex software applications	Spicetech GmbH	Virtual validation of AI & complex software application, training of state dependent field data to train an AI model for prediction of states	WP3
Validation of pose estimation models	Vitarex Studio Ltd.	A tool that can be used to evaluate and assess the accuracy, reliability and performance of pose estimation models.	WP3

3.3 The IML4E OSS platform and reference architecture

IML4E's OSS platform targets to address the basic requirements enterprises are facing when developing and deploying AI/ML services. Specifically, the introduced MLOps platform enables accessibility to scalable compute resources, automation to the execution of different ML/AI workloads and versioning traceability and transparency to the results that were generated during the development of an AI/ML model. In addition to focusing on the operational phase of an AI model, the introduced ML/AI platform introduces capabilities for deploying and operating AI services in productions together with components for monitoring AI's models results.

3.4 The IML4E MLOps teaching material

MLOps framework consists of methods and principles, tools and technologies, and platform and reference architecture layer. Individual pieces of the framework are documented in general in this chapter and in specific IML4E documents. However, the material spans many different topics and technologies, so we provide some training material to better grasp the most essential features of IML4E framework in specific and some MLOps principles in general.

Many introductions to and textbooks about MLOps already exist, for example, O'Reilly series alone contains several items like "Introducing MLOps", "Practical MLOps" or "Machine Learning Design Patterns". The cloud providers typically have introductory material for their specific platforms. We intend not to compete with the extent of such general introductions to MLOps but highlight the IML4E framework via practical tutorials of IML4E OSS platform, ethics training material, and integrative teamwork thinking and planning case exercises. A couple of general syllabi on MLOps are included, though.

The training material syllabus and links to the materials are presented in more detail in the project deliverable 4.3. Here is a summary:

- Principles and processes of MLOps, MLOps capabilities
    - Introduction to MLOps with OSS MLOps Platform: one day training course aimed at Machine Learning Engineers and Data Scientists, focusing on the practical understanding of MLOps processes using the OSS MLOps platform, covering its application in real-world scenarios from experiments to pipelines.
    - Engineering of Machine Learning Systems is a 5 credit University of Helsinki course
    - A one-day intensive course on implementing and measuring compliance with the EU AI Act in AI systems, focusing on practical strategies, automation techniques, and continuous improvement in adhering to regulatory and ethical standards.
- OSS platform and general architecture
    - Learning material for the IML4E OSS platform
    - The IML4E OSS platform (Section 5) contains *practical tutorials* - that makes it easier to familiarise one with the platform.
- Integrative team exercises for the framework
    - Software is developed in teams, in case of ML software they include PO, developers, and ML specialists / data scientists among others. Our training material includes team *case exercises* which are suitable for a solution team of developers and ML specialists. As said, it is advisable to go first through some existing introduction to MLOps, unless the team is already familiar with the field. The exercises are meant to be done as team discussion and planning sessions, but one can go through them alone, for example, Helsinki University course uses the case exercise material as final exercises. Short case exercise playbook (case exercise themes) suggests key points in actions.
    - A solution team usually includes also UX/UI specialists, and (concept) designers. For designers, we have material "Principles for natural language user interfaces." This is specific to large language models.

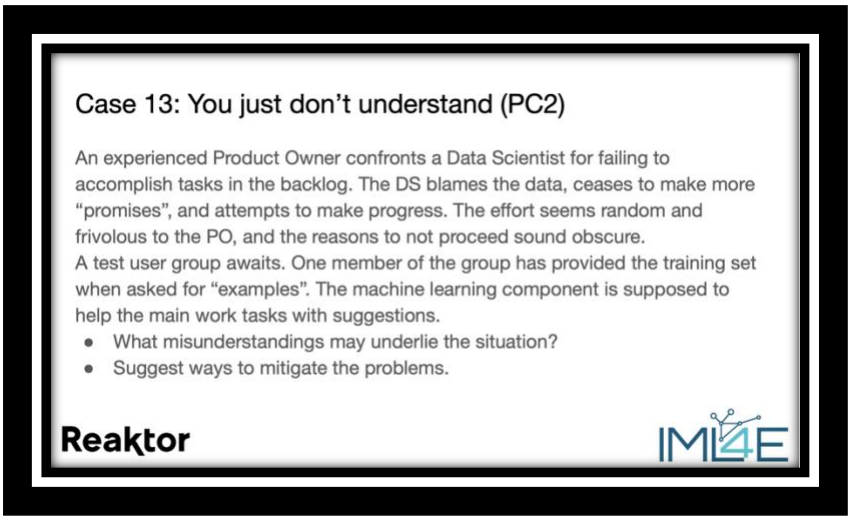

Figure 2 - Example of a teamwork exercise

3.5 MLOps capabilities

Machine Learning Operations (MLOps) capabilities are essential for ensuring the efficient and effective development, deployment, and maintenance of machine learning models within enterprises. These capabilities are critical for managing the complex lifecycle of ML models and for integrating them seamlessly into production environments. The IML4E framework, designed to support industrial machine learning applications, provides comprehensive MLOps capabilities to address these needs. The key MLOps capabilities provided by the IML4E framework include:

- **Model Development and Lifecycle Management**: This includes experiment tracking, model and artifact tracking, model performance validation, flexible deployment strategies, and automated model training. These capabilities ensure that models are developed systematically, validated thoroughly, and deployed efficiently.
- **Monitoring and Operations**: Performance monitoring, logging and auditing, and business monitoring and other feedback mechanisms help in maintaining model accuracy, ensuring compliance, and aligning model performance with business goals.
- **Data Engineering**: Capabilities such as data ingestion, data version control, data preparation, feature provision and reuse, and data validation and quality monitoring ensure that high-quality data is available for model training and that data processes are efficient, consistent and reproducible.
- **Scalability and Infrastructure Management**: Integration with CI/CD pipelines, support for containerization and virtualization, model serving flexibility, and distributed training capabilities enable scalable and flexible deployment of models across various environments.
- **Collaboration and Guidance**: Features that support collaboration among team members, such as sharing models, data, and experiments, enhance teamwork and ensure that knowledge is effectively shared and utilized.

Details of these capabilities are given in the tables below. Moreover, Annex A shows how the individual IML4E methods, tools, and techniques support the given capabilities.

Table 4: Model development and lifecycle management capabilities

Model Development and Lifecycle Management	
Experiment Tracking	Track and log different experiments along with their parameters, metrics, and outputs.
Model and Artifact tracking	Capabilities to track, version and store models, related artifacts and metadata in various versions.
Model performance validation	Validate model accuracy and other metrics to ensure they meet predefined thresholds before deployment.
Model deployment flexibility	Support for different deployment strategies e.g. A/B testing, Blue-green testing or Canary deployments
Model training automation	Automate the training process, including the management of compute resources and basic activities for training like optimize model parameters automatically for the best performance.

Table 5: Monitoring and operations capabilities

Monitoring and Operations	
Performance Monitoring	Track the performance of models over time to detect issues like model drift, data drift, or degradation in prediction accuracy.
Logging and Auditing	Maintain logs of model predictions and user actions for audit and compliance purposes.
Business Monitoring & Feedback	Track the performance of models with respect to their business value and based on user feedback.

Table 6: Data engineering capabilities

Data Engineering	
Data Ingestion	Ability to ingest data from various sources in different formats.
Data Version Control	Manage versions of both data and data preparation code.
Data Preparation	Tools for cleaning, preprocessing, and transforming data to make it ready for training.
Feature Provision and Reuse	Centralized data and feature provision and support for feature reuse across different machine learning projects.
Data Validation & Quality Monitoring	Validate and monitor data and features to ensure they meet predefined thresholds before training

Table 7: Scalability and infrastructure management capabilities

Scalability and Infrastructure Management	
CI/CD Pipelines:	Integration with continuous integration and continuous deployment tools to streamline the deployment of data, data preparation code, and machine learning models.
Containerization and Virtualization:	Support for container technologies like Docker and orchestration systems like Kubernetes.

Model Serving Flexibility	Capabilities to deploy models in various environments (cloud, on-premises, edge) and via different methods (APIs, batch processing).
Distributed Training	Data and workloads are distributed over worker nodes while the server node maintains globally shared parameters, like the weight's updates during training.

Table 8: Collaboration and guidance capability

Collaboration & Guidance	
Collaboration Support	Facilitate collaboration between team members by sharing models, data, and experiments.

4 IML4E Experimentation and OSS platform

The OSS platforms address the above requirements by onboarding different open-source technologies and by integrating them into one platform. The IML4E team decided to introduce leading open-source tools with an established user community aiming to cover specific MLOps requirements. The introduced tools and their purposes are the followings:

- **Kubernetes:** The backbone container orchestrator that enables AI/ML workloads to utilize different CPU and GPU resources. Kubernetes is mainly abstracting the complexity of introducing and offering hardware resources to compute workloads developed by data scientists and ML practitioners. Kubernetes is a general-purpose cluster management tool and thus it comes with each one complexity and set of commands. Interacting with Kubernetes requires a steep learning curve and often ML/AI platforms introduce tools and templates to hide away the complexity that Kubernetes introduces.
- **MLFlow:** Manages experiment tracking and model registry, enabling versioning of the different models developed by the data scientists. MLFlow is quite versatile and can capture any user defined metadata related with the development of ML/AI models.
- **Kubeflow:** Orchestrates ML workflows on top of Kubernetes. Kubeflow plays a pivotal role on compiling and deploying the ML/AI experiment workflows to the Kubernetes for execution. To distinguish Kubeflow with Kubernetes, Kubeflow enables users to easily access the Kubernetes resources for running their experiment workflows, while Kubernetes enables the deployed experiment workflows to scale to clusters of attached hardware resources. Kubeflow abstracts away the complexity of Kubernetes and Kubernetes abstracts away the complexity of managing and operating different hardware resources into one cluster.
- **KServe:** Facilitates model deployment and serving. KServe addresses the demands that enterprises have for a standard and automated way to deploy AI services that can scale from low to large usage/traffic. KServe is also capable of serving different AI models deriving from different AI frameworks such as TensorFlow, PyTorch and others.
- **Prometheus & Grafana:** Provide monitoring solutions with advanced visualization capabilities. With Grafana, enteprises can visualize predictions, fairness, accuracy, service utilization and other metrics. While Grafana is the front-end tool for visualizing AI metrics, Prometheus is the database that collects AI metrics at real-time.

The aforementioned tools are combined into one platform which has been open sourced. In Figure 3 more technologies are present but for the economy of the document we decided those technologies not to be described. If a reader is particularly interested in learning more about the platform and the technologies introduced the following links are relevant:

- link to the OSS platform hosted in GitHub: https://github.com/OSS-MLOPS-PLATFORM/oss-mlops-platform/blob/main/README.md
- link to the documentation describing the OSS platform in details: IML4E-D4.2-Initial MLOps methodology and the architecture of the IML4E framework.pdf

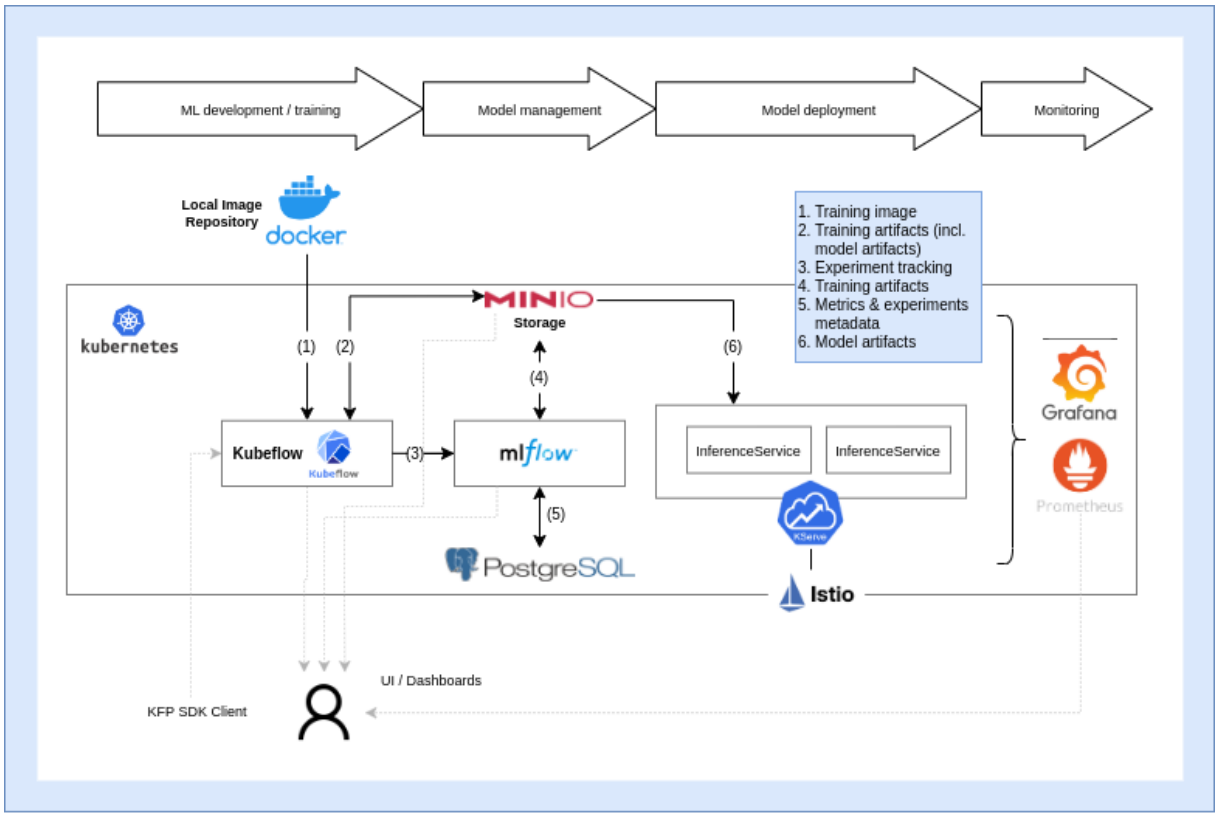

Figure 3 - The IML4E OSS platform

5 IML4E MLOps methods and principles

5.1 AI Ethics in MLOps

Though tooling and the integration of tools are required to ultimately achieve the automation that forms the basis of MLOps, AI ethics is still an emerging area of both research and practice. In this regard, *what* needs to be measured or monitored and *how* (real-time, at set times, or when needed because of changes/actions such as new model training or deployment) is still an open question. The practical implementation of AI ethics in general is still fuzzy as far as good practices and suitable processes are considered. As AI ethics related process maturity overall is not yet at a level where clear requirements for tools (or toolchains) can be defined, devising suitable tools is challenging.

AI ethics has typically been approached through *principles*. Principles describe, on a general level, what an AI system should be like in order to be considered ethical – at least according to the party behind the principles. One example of such a set of guidelines containing principles is the "Ethics guidelines for trustworthy AI", created by an expert group set up by the European Commission. These guidelines contain the following four principles: (i) Respect for human autonomy, (ii) Prevention of harm, (iii) Fairness, and (iv) Explainability. These four requirements further motivate seven requirements, which offer more tangible guidance for actual AI development. As an example of a principle, fairness focuses on issues such as bias and discrimination, as well as on fostering equality (equal access to technology, etc.). Operationalizing these principles remains a pivotal challenge in AI ethics overall and extends to measuring and monitoring them.

Most work in monitoring ethical requirements has focused on monitoring fairness. Arguably, this is likely a result of algorithmic fairness being more tangibly related to ML models than some broader issues related to other principles issues such as minimizing harm. Existing tools for de-biasing datasets and monitoring of fairness contribute towards realizing some aspects of AI ethics in MLOps contexts.

However, run-time monitoring is only one part of AI ethics, even in MLOps contexts.

If we consider ML development related metrics through the following typology, metrics related to AI ethics can be identified in all seven categories:

- Data metrics (e.g., data quality, preprocessing, diversity)
- ML model metrics (e.g., various performance metrics, training metrics; incl. data related model outputs)
- System/infrastructure metrics (e.g., hardware metrics, software metrics for system hosting ML models)
- Process metrics (e.g., DevOps or CI/CD metrics, code-related metrics)
- Business metrics (e.g., financial metrics for model training, positive/negative impacts of system)
- User metrics (e.g., system use analytics, user experience)
- Domain-specific metrics (e.g., patient data in the medical domain)

Continuing with the example of fairness, the training data needs to be assessed in terms of fairness (bias, diversity, etc.) in order to train models that are similarly unbiased (or less biased). Fairness metrics to determine model fairness need to be outlined. Some may only be needed periodically upon training new models, while in some cases *fairness drift* may need to be monitored during runtime. Data from users may be used to supplement the monitoring of fairness (e.g., whether users discern any issues that they may report through different channels; sometimes indirectly on social media). In this fashion, a single ethical principle alone may require a more holistic approach to monitoring and metrics.

Yet not all systems include fairness consideration to a notable extent. For example, an ML system monitoring machinery in a factory that has no interaction with humans is not particularly prone to face issues related to fairness. This highlights the context-specific nature of ethics, which poses problems for the practical implementation of AI ethics by making one-size-fits-all solutions difficult, if not impossible, to achieve.

In addition to tooling to support run-time monitoring of AI ethics in MLOps contexts, further support is required for helping organizations quantify (metrics) ethics in ML development overall. Not all AI ethics issues require

runtime monitoring, but simply determining what to measure in the first place, and when or with what frequency, is a challenge faced by organizations when seeking to "do" AI ethics. An ethical framework is needed as a basis for this consideration, in order to define what is ethical in the given system context. Such a framework may be a set of guidelines such as the trustworthy AI ones mentioned above. Even then, however, this requires context-specific deliberation from the organization(s) developing the ML system.

5.2 The IML4E MLOPs Maturity Assessment Framework V2

This section describes the second version of the IML4E Maturity Assessment Framework. The second version updates some of the classes and mappings of the first version published in Deliverable D4.2 after the first version was evaluated by the case study partners as part of the interim evaluation.

5.2.1 Overview

The IML4E Maturity Assessment aims to provide a schema that can be used to examine the current state of implementation of MLOps in an organization and to make informed decisions about which improvement actions may be useful additions in light of the current maturity and expansion goals. It helps organizations understand where they are in their MLOps journey, inspires discussion among stakeholders and identify areas for improvement. In its current state, the IML4E Maturity Assessment is designed as a self-assessment. Based on a maturity definition from Microsoft consisting of 5 maturity levels, we have created a collection of assignments between maturity levels and statements on process maturity for 7 different areas. Figure 4 shows the assessment areas of the IML4E MLOps Maturity Framework.

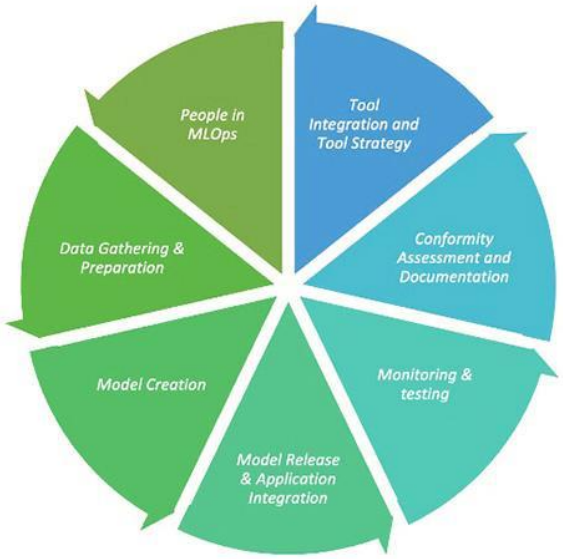

Figure 4 - Assessment areas of the IML4E MLOps Maturity Framework

Maturity levels and statements are provided in the form of an Excel spreadsheet so that an organization can provide scores for each area in a self-assessment. These are finally summarized and clearly presented with the help of a spider diagram. An initial internal project evaluation has shown that such a self-assessment stimulates a wide range of discussions and paves the way for the identification of targeted and systematic improvements. These self-assessment tables are introduced in the following sections.

5.2.2 People in MLOps

People are one of the most important aspects in MLOps. It must be ensured that all necessary roles are filled by suitable people and that the people are able to communicate meaningfully with each other. Especially relevant is the communication between data teams (e.g., Data Analysts/Data Engineers and Data Scientist), software development and operations teams (e.g., Software Engineers, IT-operations) and the business teams (e.g., Business Analysts, End User). The roles are defined as follows.

- **Business Analyst:** has deep in-depth knowledge of business domains and processes. They help the technical team understand what is possible and how to frame the business problem into an ML problem. They help the business team understand the value offered by models and how to use them. They can be instrumental in any phase where a deeper understanding of the data is crucial.
- **Data Analysts/Data Engineers**: work in coordination with product managers and the business unit to uncover insights from user data. They manage how the data is collected, processed, and stored to be imported and exported from the software reliably.
- **Data Scientist**: are responsible for analyzing and processing data. They build and test the ML models and then send the models to the production unit. In some enterprises, they are also responsible for monitoring the performance of models once they are put into production.
- **MLOps Engineers**: design, build, and run machine learning systems at scale. Provide data scientists and other roles with access to the specialized tools and infrastructure (e.g., storage, distributed compute, GPUs, etc.) they need across the data science lifecycle. They develop the methodologies to balance unique data science requirements with those of the rest of the business to provide integration with existing processes and CI/CD pipelines.
- **SW Engineer:** designs, develops, tests, and implements programs and applications. Works with data engineers and data scientists, focusing on the productionalization and integration of ML models and the supporting infrastructure. They develop solutions based on the ML architect's blueprints, selecting, and building necessary tools.
- **IT-Operations:** are responsible for operating applications/services and models in production.
- **End User:** is the user that directly interacts and benefits from the ML-enabled application or service.

The following table describes the various assessment statements this aspect.

Table 9: Level definitions for the assessment area "People"

	People					
	Topic[1]	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
1.	Involvement and integration of Data Analysts/Data Engineers (MS)		Data Analysts/Data Engineers are siloed, not in regular communication with the larger team.	Data Analysts/Data Engineers are working with data scientists.		Data Analysts/Data Engineers, Data Scientists, SW Engineers and MLOps Engineers are working together to manage inputs/outputs flexibly.
2.	Involvement and integration of Data Scientist (MS)		Data Scientist are siloed and not in regular communications with the larger team.		Data Scientists are working directly with Data Engineers and MLOps Engineers to convert experimentation code into repeatable scripts/jobs that allow for automated model training and evaluation.	Data Scientists are working with Software Engineers to identify markers and systematic feedback for Data Engineers
3.	Involvement and integration of Software Engineers (MS)			Software Engineers are siloed, receive model remotely from the other team members.	Software Engineers are working with Data Scientist to automate model integration into application code.	Software Engineers are working with Data Scientist to automate model integration into application code. Implementing post-deployment metrics gathering.
4.	Involvement and integration of IT-Operations	IT-Operations is siloed and not in regular communications with the larger team.	IT-Operations is in regular communications with SW Engineers to manage the deployment of application code.		IT-Operations is in regular communications with SW Engineers and Data Scientist to manage the deployment of applications and models.	IT-Operations is in regular communications with SW Engineers, Data Scientist and Data Engineers to allow for feedback on

[1] All topics/level definitions that are marked with (MS) are adopted from (Microsoft 2022)

						data and software related issues during operations.
5.	Involvement and integration of Business Analysts and End Users	Business Analysts are siloed and only in communication with Data Analysts/Data Engineers.		Business Analysts are in regular communication with Data Scientist to address issues with models and prediction results.	Business Analysts are in direct communication with the End Users to gather direct feedback regarding efficiency and effectivity of the model in production.	Business Analysts are in direct communication with all other roles to allow for immediate feedback on business related fails and issues.

5.2.3 Data Gathering & Preparation

During data gathering & preparation, data are gathered and prepared. It aims to identify the required data sources and the best data sets in the correct distributions. The main purpose is to collect the data in such a way that the model output can be provided as efficiently as possible, enrich the data by labeling, store the lineage of the data, verify the quality of the labeled and prepared data, establish specific metrics to measure the quality of the data, store and analyze the data. As a result, all relevant data sources are identified and evaluated regarding the available data, metadata formats and schemas for labeling and annotating data have been defined and documented, a strategy for long-term data management (data governance) has been defined and documented and a process for preparing the data is specified. The following table describes the various assessment statements of this aspect.

Table 10: Level definitions for the assessment area "Data Gathering & Preparation"

	Data Gathering & Preparation					
	Topic[1]	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
6.	Data pipeline automation	Data gathered and prepared manually.		The data pipeline collects data semi-automatically. Data processing is automated in smaller chunks and for partial aspects of the process (e.g., cleaning, labeling).	The data pipeline gathers and prepares data automatically and processes data on demand. This is usually based on batch processing.	Data pipeline gathers and prepares data continuously (e.g., by stream processing).
7.	Data reuse	Data are gathered and prepared for individual models/solutions.		Data are gathered and prepared/reused for multiple models/solutions.		Data are systematically managed for reuse (e.g., by dedicated feature stores).

8.	Data version control	Data and data related artifacts are not version controlled in a consistent manner.		Data, models and other artifacts are partially version controlled.	Data and preparation infrastructure is under version control.	Bind and store model predictions with the corresponding input data and with the model version that we have deployed and created during the training phase.
9.	Data quality assurance		Doing data sanity checks and checking if data falls within simple metrics like min-max ranges.		Data unit tests based on a set of predefined validation rules. Data units that violate these validation rules are displayed and further examined in a predefined process	Continuous analysis of data sets, i.e., examining gaps in data, missing values, existing trends, and so forth.
10.	Data unit testing		Data unit tests are done sporadically.		Data unit tests are done based on tools and automatically triggered for new data sets.	Data unit testing is fully automatized and synchronized with other MLOps activities like data preparation, model training and operations.

5.2.4 Model Creation

Model Creation aims at providing the best modelling solution for a given task. For doing so, data scientists parameterize and train model variants based on the available data and their characteristics. In practice, training runs (experiments) are performed with different model architectures and initial parameters (hyperparameters) and the results are compared. All resulting models are benchmarked, evaluating their properties in terms of accuracy and generalizability. For model creation, predefined frameworks usually support in creating the model code and implementing the algorithms. The following table describes the various assessment statements of this aspect.

Table 11: Level definitions for the assessment area "Model Creation"

	Model Creation					
	Topic[1]	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***

11.	Management level of compute during training (MS)	Compute is likely not managed.	Compute is managed as a single node instance that is unlikely to scale.			Compute is managed as a multi node cluster that scales dynamically.
12.	Experiment tracking (MS)	Experiments aren't predictably tracked.		Experiments are partially tracked. Not all experiments are reproducible.		Experiment results tracked automatically. All experiments are reproducible.
13.	Model versioning		Result may be a single model file manually handed off with inputs/outputs.		Both training code and resulting models are version controlled.	Both training code and resulting models are version controlled. Retraining is triggered automatically based on production metrics.
14.	Automation of hyperparameter and architecture determination, Auto ML		Manual hyperparameter and architecture determination.	Automated hyperparameter and architecture optimization for a given training pipeline.	Automated hyperparameter adjustment for each model iteration.	Fully automated hyperparameter and architecture determination for each model iteration.
15.	Training pipeline integration and deployment		Training pipeline is integrated and deployed manually.		Training pipeline is continuously deployed (CI/CD of training pipeline).	Training pipeline is continuously deployed (CI/CD of training pipeline).
16.	Model debugging	Sporadic and Real model debugging.		Model debugging is done based on a predefined strategy with adequate tools that allow to gain model insights and transparency.		Model debugging is done as part of the failure response process for each of the failures that needs explanation and correction. Model debugging is done with adequate tools that allow to gain model insights and transparency.

5.2.5 Model Release & Application Integration

During Model Release and Application Integration the model is transferred in its operational setting. It is either integrated with a larger application or directly released in its operational environment. The following table describes the various assessment statements regarding this aspect.

Table 12: Level definitions for the assessment area "Model Release & Integration"

	Model Release & Integration					
	Topic[1]	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
17.	Scoring and evaluation of models (MS)		Scoring script and release KPIs are manually created after experimentation phase. They are likely version controlled.			Scoring script and release KPIs are continuously updated, and version controlled with tests.
18.	Responsibility for model release (MS)	Release handled by Data Scientist or Data Engineer alone.	Release is handed off to Software Engineers.	Release managed by Software Engineers and MLOps Engineers.	Release managed by continuous integration pipeline.	Release managed by continuous integration and CI/CD pipeline. Releases are triggered automatically based on training and production metrics.
19.	Testing the model	No dedicated testing of the model.		Basic integration tests exist for the model.		Unit and Integration tests for each model release
20.	Expertise for model release (MS)	Heavily reliant on data scientist expertise to manually implement and release the model.		Heavily reliant on data scientist expertise to implement model. Release is based on predefined KPIs and quality gates.		Less reliant on data scientist expertise. Training and release are automated based on dedicated KPIs and quality gates.
21.	Degree of release automation	Manual releases each time.		Release is based on defined KPIs and release criteria. However, integration and	Models are continuously integrated with the prediction pipeline (CI of model).	Models are continuously deployed (CI/CD of model).

				deployment are still manual.		
22.	Prediction pipeline testing			Prediction pipeline has unit tests.		Prediction pipeline has unit/integration tests.
23.	Rollback of release	No rollback possible.		Model can be rolled back with a certain degree of manual intervention.		Model is continuously checked against older version and automatically rolled back if model performance becomes worse than the older models.

5.2.6 Monitoring & testing

Monitoring and testing ensure that the performance and quality of a model can be continuously checked and validated. This is usually done by checking data, models, and the pipelines against dedicated test suites and KPIs. Testing, monitoring, and validation activities span over the whole lifecycle of ML-based software. It starts with validating the requirements, data and the model itself. But it also includes debugging, testing, and validation activities at different levels of integration up to the deployment/release. MLOps aims to neatly integrate the testing and validation activities in the overall model development process and aims to automate as much as possible. Monitoring provides insight into data and ML models during training and in production to ensure they are working as intended. Monitoring is also a prerequisite to derive meaningful business value from models. Without understanding how your model's predictions impact downstream business KPIs and revenue, it is impossible to make further improvements and optimizations to the modelling and training pipeline. In addition, any model that is transferred to operational use can start to lose performance. Such performance loss must be detected, documented, and communicated. There must be a way to continuously report to stakeholders in the organization on whether and how the machine learning solution provided solves the defined problems and meets its targets. Monitoring and appropriate visualization help to make this process as efficient as possible. The following table describes the various assessment statements for this aspect.

Table 13: Level definitions for the assessment area "Monitoring & Testing"

	Monitoring & Testing					
	Topic	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
24.	Monitoring scope & integration	Monitoring of some technical KPIs during data preparation, training, deployment, and operation.		Integrated monitoring covering aspects of data preparation, training, and deployment in a systematic manner.		Comprehensive monitoring and control of relevant technical and business related KPIs in an integrated manner.

25.	Cross application monitoring	Individual monitoring of some technical KPIs for only one deployment or project.		Monitoring of relevant technical KPIs covering potential multiple deployments.		Comprehensive monitoring and control of relevant technical and business related KPIs in an integrated manner covering potential multiple deployments.
26.	Test automation	Manual testing of model and application.	Automated testing of application code.	Automated testing of the model code. Systematic model testing.	Automated testing of the training pipeline and the model.	Integrated test process targeting systematic test automation in all phases of the training and deployment process.
27.	Test scope	Testing of application code and model separately.	Testing of application code and model in a systematic integration process.	Testing that the ML model successfully loads into production serving and the prediction on real-life data is generated as expected. Testing for model degradation over time, testing for sudden performance issues, Model API testing, testing algorithmic correctness.	Systematically targeting model performance differences in training and production environments by applying dedicated test suites. Integrated A/B testing of model performance for deployment.	Testing in different deployment schemes like canary or blue-green deployments. Integrated Quality Assessment that combines and supplements monitoring and testing in a systematic manner (e.g. provides tests for critical areas that are not easy to monitor, providing dedicated tests in case monitoring reveals potential issues).

5.2.7 Conformity Assessment and Documentation

According to the European draft AI law, "high-risk" AI systems must undergo a conformity assessment before they can enter the European market. The conformity assessment targets the trustworthiness (including safety and security, explainability, transparency, reliability, accuracy, maintainability, fairness, degree of autonomy and controllability by humans and accountability) of the system and is carried out by a third party or is based on internal control; the main subject of the assessment is a technical documentation of the AI system. The coordination and consolidation of the results of a conformity assessment is an important step in the operationalization of a conformity assessment and is addressed by the following criteria.

Table 14: Level definitions for the assessment area "Conformity Assessment & Documentation"

	Conformity Assessment & Documentation					
	Topic	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
28.	Conformity Assessment	Conformity to policies and related quality attributes are checked manually for the application.		Relevant compliance policies and related quality attributes are checked based on automated tools integrated in the CI/CD pipeline. Training process is checked for compliance (e.g., DSGVO)		Continuous audit of compliance policies and related quality attributes over the whole MLOps life cycle.
29.	Documentation	No documentation.	Manual documentation creation for some aspects of the model.	Manual documentation creation of data and model.	Semi-automatic documentation stored and deployed with the model.	Fully automated documentation generation covering all aspects of model and code as well as continuous documentation of up-to-date KPI fulfillment based on monitoring information.

5.2.8 Tool Integration and Tool Strategy

Tools and their appropriate integration are the basis for a high degree of automation. The reliance on tools over the entire lifecycle of an ML-based application makes tool integration particularly challenging in the case of MLOps, since tools from the areas of data engineering, software engineering, operations and the business units must be properly chosen and integrated. The following table describes the various assessment statements of this aspect.

Table 15: Level definitions for the assessment area "Tool Integration & Tool Strategy"

	Tool Integration and Tool Strategy					
	Topic	***Level 0***	***Level 1***	***Level 2***	***Level 3***	***Level 4***
30.	Automation by tools	Individual processes are supported by tools.		Tools are integrated for individual phases of the MLOps life cycle (e.g., data preparation, model training, release etc.)		Fully automated tooling supports the MLOps process.
31.	Tool strategy and harmonization	No harmonization of tools	Tools are harmonized over multiple projects.		Harmonized tooling for all relevant phases of the experimentation, training, deployment, and operations.	Tool strategy that allows for a sustainable management of harmonized tooling over all relevant phases of the MLOps life cycle.
32.	Visualization and dashboarding	No visualization.	Individual visualization techniques to support data exploration, verification, validation, and training.		Integrated visualization dashboards to systematically guide data exploration, verification, validation, and training.	Comprehensive visualization dashboards to monitor and control technical and business related KPIs by business experts.

5.3 The IML4E MLOps Testing Methodology

The IML4E MLOps testing methodology aims to provide a schema to systematically apply testing to MLOps processes and thus increase the quality of ML-based application by maintaining efficiency through targeted testing. It is a comprehensive framework that incorporates testing in all phases during developing, integrating, and operating ML-based systems by combining classical software engineering with data science activities to ensure the quality and reliability of ML-based systems. It addresses the quality assurance challenges specific to ML systems, ensuring that both the software and the ML components meet the required performance, reliability, and compliance standards throughout their lifecycle.

5.3.1 A workflow perspective for developing and operating ML-based systems

In the context of identifying and locating important quality assurance activities, this methodology introduces a workflow model that encompasses both the perspective of classical software engineering and the data science activities of machine learning. When defining the workflow model, design activities for the overall system and individual components were not mapped. Instead, software development activities that are required for the provision of highly automated training infrastructures are considered. Figure 5 shows the abstract definition of the workflow including classical software development activities, as well as typical data science activities like data preparation, training, and validation. The workflow is based on the activities known from the established workflow models for traditional software development and data science mentioned above. It describes the main activities and artifacts from both domains and as such describes the development, integration, and operation of an ML-based application as an integrated software product consisting of ML models and traditional software.

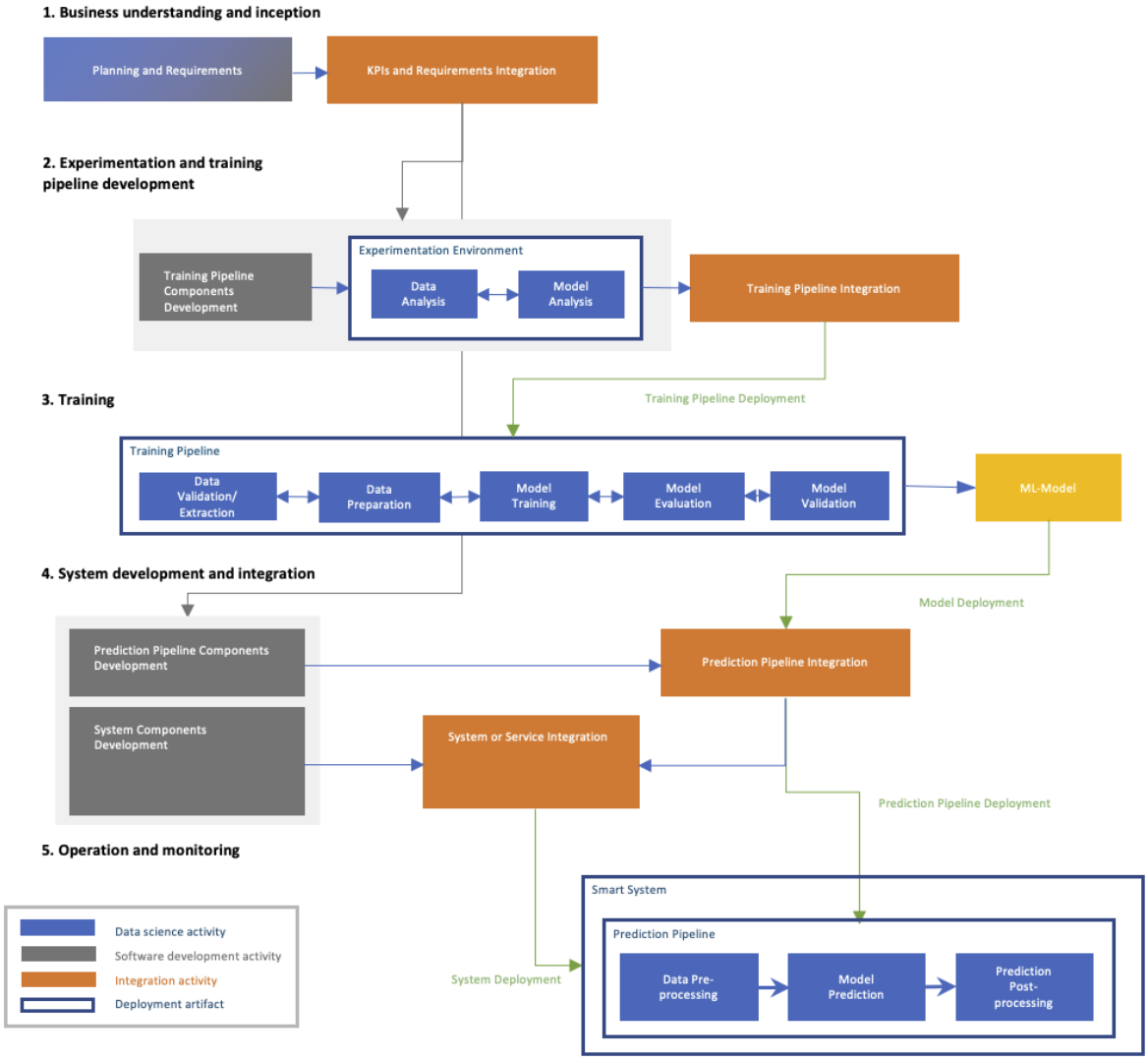

Figure 5 – Development and training workflow to develop, train and deploy ML-based systems or applications

On the high-level, the workflow distinguishes five different phases that are differentiated and detailed in Figure 5 and explained below. Each of these phases are defined by a set of activities that are roughly assigned to the field of data science (blue), software development (grey), and integration (orange).

- **Business understanding and inception** aims to derive a basic understanding of the overall objectives and requirements of the ML-based system. For this purpose, it is necessary to understand the business and technical context of the system and to obtain a basic understanding of the data available for modelling.
- **Experimentation and training pipeline development** aims to evaluate the data and modelling approach and to build a modelling infrastructure. In this phase, PoC systems are developed and evaluated for their basic applicability. Depending on the modelling approach, the training and data preparation pipeline is developed and integrated.
- **Training** aims to create new models based on the modelling approach and with the help of the training pipeline. Depending on the degree of automation available, activities for data preparation, training, including the tuning of hyperparameters, validation, and quality assurance of the model are executed more or less automatically.

- **System development and integration** aims to integrate the ML model into a software environment. The complexity of the integration depends on the application context and ranges from the simple provision of a user interface to complex integration with other models, sensor systems and complex control software, such as in automated driving.
- **Operation and monitoring** is finally the phase in which the integrated ML-based system is being executed and **monitored** in its operating environment. Depending on the application context, various operating environments are possible, ranging from a simple cloud deployment to a distributed edge deployment.

Most of the phases end with a dedicated integration activity (depicted in orange), integrating the key work products and as such defines the main artifact that is propagated or deployed to the next phases (green arrow).

5.3.1.1 Overview on test methods for testing ML-based systems

The work products of a given workflow phase and their systematic integration are usually the subject of systematic testing. Testing is considered here as the process of evaluating a software system or component to determine deviations between expected and actual behaviour. The main objectives of testing are to detect bugs, verify functionality, and ensure that the software meets the specified requirements.

Testing is usually performed during the development phase or as a special quality assurance measure prior to deployment (Phase 1 – 4 in Figure 5), but can also be performed during operation (Phase 5 in Figure 5). The latter becomes necessary especially for systems with strong dynamics or for systems with a high dependency on the environment. Basically, a distinction can be made between dynamic and static testing methods.

1. In **dynamic testing**, the system is executed. Specific inputs or test cases are applied as inputs to the running system and the observed results are compared with the expected results.
2. In **static testing**, the system is not executed or artifacts that cannot be executed are examined. These include specifications, architectures as well as data. Static testing can be done automatically with the help of dedicated **analysis tools** or manually through **review**.
3. **Monitoring** is a testing method that does continuous observation and measurement of a software system during its runtime. It involves the collection and analysis of real-time data about the system's performance, behaviour, and various **operational metrics**. Monitoring helps to identify potential problems, bottlenecks, or anomalies that may affect the availability, performance, safety, security of the system. It provides insights into system health, usage patterns, resource utilization, and other relevant aspects.

In summary, review, analysis, dynamic testing and monitoring are all considered useful testing methods to test ML-based systems. Static and dynamic testing often focus on assessing the correctness, functionality, and compliance of software systems before deployment, while monitoring concentrates on real-time observation, measurement, and analysis of the system's performance during runtime. All these activities are considered crucial for maintaining software quality, reliability, and overall system health for classical software systems as well as for ML-based systems.

5.3.1.2 Considerations in defining adequate test items for testing ML-based systems

The term test item describes the item to be tested by a particular test method. In the case of dynamic testing, this is normally referred to as System Under Test (SUT), which somehow highlights the dynamic nature of the test item. However, analogous to the ISTQB (International Software Testing Qualifications Board), we use the concept of a test item in the following, which includes any work product in the life cycle of an ML-based system, to clarify that we deal with both static and dynamic test procedures.

Although our primary test item, as the name of this report suggests, is the ML-based system, we obtain several other test items that can be tested individually or partially integrated considering the development of an ML-based system as well as its systematic integration from individual components.

Due to the high importance of the data and/or the training process, the literature explicitly distinguishes between test items of the training phase, which are crucial for the quality and properties of an ML model, and the

development and runtime artifacts, which are relevant for the composition and integration of an ML-based system based on individual components. Zhang et al., 2019 for example distinguishes on a high-level between testing data, testing the learning program (i.e., the training infrastructure) and testing the ML-framework (i.e., the libraries and building blocks that are used to define models).

"Thus, when conducting ML testing, developers may need to try to find bugs in every component including the data, the learning program, and the framework." (Zhang et al., 2019).

As already mentioned before, test items are normally the work products of a given workflow activity or phase. At this place we will start with a more general overview on considerable relevant test items.

Test items are:

1. **Specifications, requirements, and planning documents:** Before a system can be meaningfully constructed or optimized, it is necessary to determine what the system is supposed to accomplish, how it is to be structured, and how the necessary processes are to be planned. Testing these specifications, requirements and planning documents is mainly done by reviews and has to consider the different viewpoints and terminologies in software engineering and data science.
2. **Data:** Unlike in traditional software development, data and its provision as datasets for training, testing, and validation are one of the most important artifacts in machine learning. Testing of the data can be realized via different methods. These include reviews, static and statistical analysis, directed data testing by operationalizing the data through test and analysis models. This involves testing the data structure, its markup and metadata, as well as its meaning.
3. **Development, modelling, and training infrastructure:** Especially with regard to the automation of particularly complex processes such as data preparation and the tuning of relevant hyperparameters, as well as training, automation and tool support are usually relied on. Nowadays, we speak of pipelines when there is a toolchain that automates more complex processes. Since these infrastructures have a high impact on the quality of an application or a product and usually have to be rebuilt and tuned for new products and applications, the testing of these infrastructures is a necessary requirement.
4. **Models:** Models are the main result of the training phase. The testing of models ensures that a model meets the requirements placed on it. Requirements are usually formulated by KPIs along various quality dimensions. Testing of a model is usually done dynamically by feeding a variety of test data into the model and comparing the actual results with the current results. Errors are usually quantified and qualified using statistical measures.
5. **The ML-based system:** Finally, the resulting software system shall be tested across its integration stages. This includes the individual software components, their partial integration, and the integrated system in the various execution environments. In view of the fact that this is an ML-based system, ML components such as the model or the integration of the model with its pre- and post-processing components (prediction pipeline) are mentioned separately. For testing, a variety of test methods are used, i.e. dynamic testing, static analysis, reviews as well as various monitoring activities at runtime.

5.3.2 Detailed test item identification and definition of test activities

Test activities range from testing the individual test items and their integration to larger items in the integration phases. Nearly all phases of the workflow depicted in Figure 5 end with a dedicated integration phase having work products associated that are subject to dedicated testing activities. However, also intermediate work products are of interest for testing. Figure 6 shows the development and training workflow specified in Figure 5 extended by dedicated testing and monitoring activities.

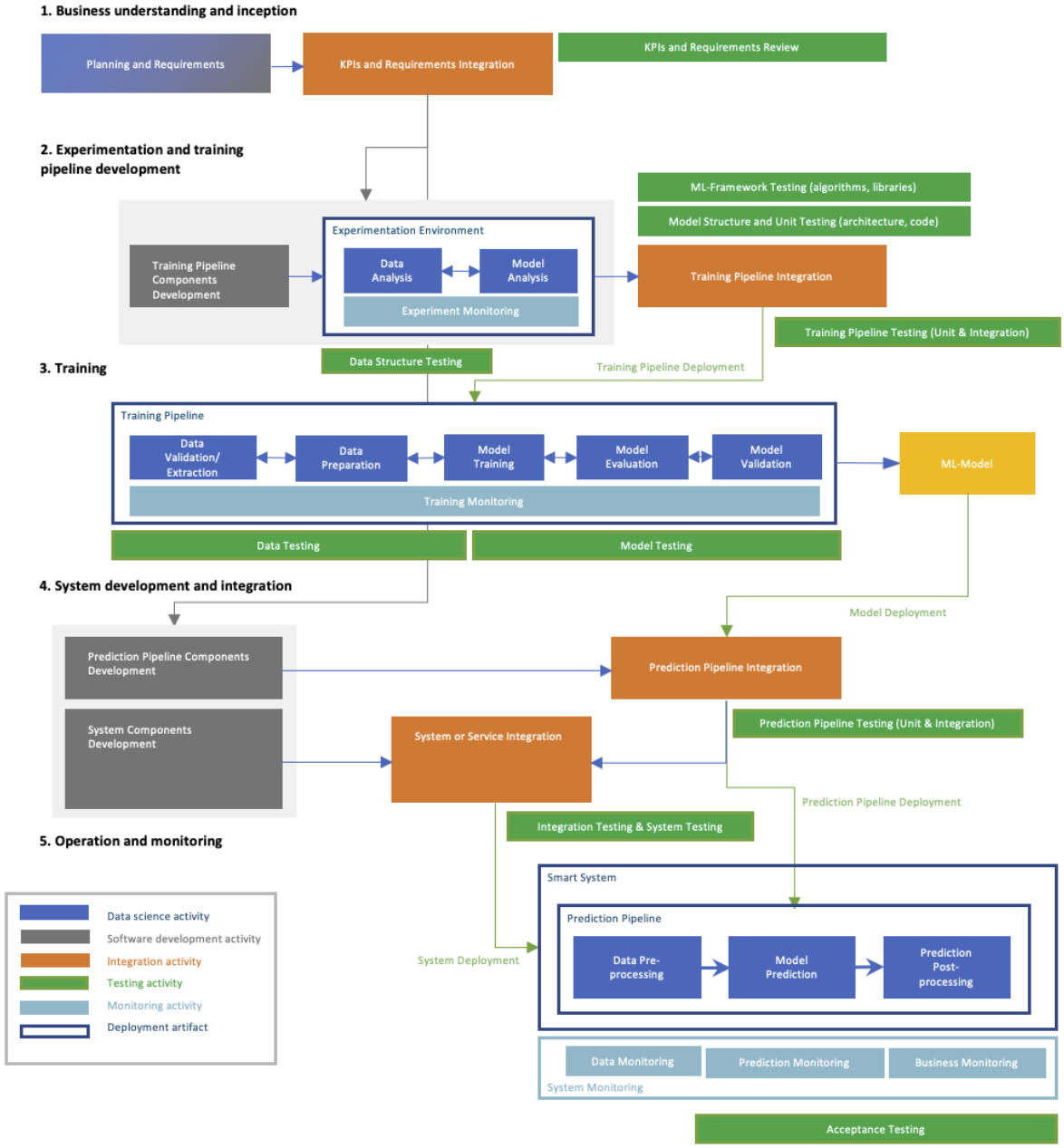

Figure 6 – Development and training workflow extended by testing and monitoring activities

Testing activates are denoted in green and monitoring activities are denoted in light blue. Please note that typical data science activities like Data Validation, Model Evaluation, and Model Validation include dedicated testing activities. These activities will be discussed in relation to the general testing activities denoted in green, since they are sometimes the same and show a larger amount of overlap in approaches, methods, and results.

The rest of the text identifies the key work products and acceptance criteria for each phase of the workflow. Each work product can then be considered as an independent test item to which suitable test methods and objectives are assigned. Finally, each combination of test item, acceptance criteria, and test method can then be assigned to the testing activities in Figure 6.

5.3.2.1 Test items of the business understanding and inception phase

Figure 7 - Testing activities in the phase of business understanding and inception.

The **Business understanding and inception** phase aims for deriving and integrating the major KPIs and requirements of the application, service, or system. Major work products are the business related KPIs, the technical KPIs and the overall requirements and quality criteria. The activity *Planning and Requirements* address general requirements management and planning activities while the activity *KPIs and Requirements Integration* addresses in particular the harmonization of KPIs and requirements with regard to completeness consistency, absence of contradictions and other cross-cutting concerns. Considering the iterative character of ML, KPIs, and requirements need to be adapted in the following phases.

KPIs and Requirements Review is considered a testing activity that checks individual KPIs and requirements for correctness, realizability, completeness, etc. and sets of KPIs and requirements completeness, consistency, and absence of contradictions and other cross-cutting concerns.

Table 16 provides an overview on the major work products of the **business understanding and inception** phase, the related acceptance criteria and items.

Table 16: Work products, acceptance criteria and test types for the business understanding and inception phase.

Work product/test item	Acceptance criteria	Test method/test objective
Business KPIs	– Business KPIs are correct, complete, consistent, unambiguous, measurable, traceable, feasible and validated.	– Review of business KPIs
Training KPIs and acceptance criteria for training	– Training KPIs and acceptance criteria for training are correct, complete, consistent, unambiguous, measurable, traceable, feasible and validated.	– Review of training KPIs
Requirements and quality criteria	– Requirements and quality criteria are atomic, correct, complete, consistent, unambiguous, verifiable, traceable and validated.	– Review of data quality criteria

5.3.2.2 Test items of experimentation and training pipeline development phase

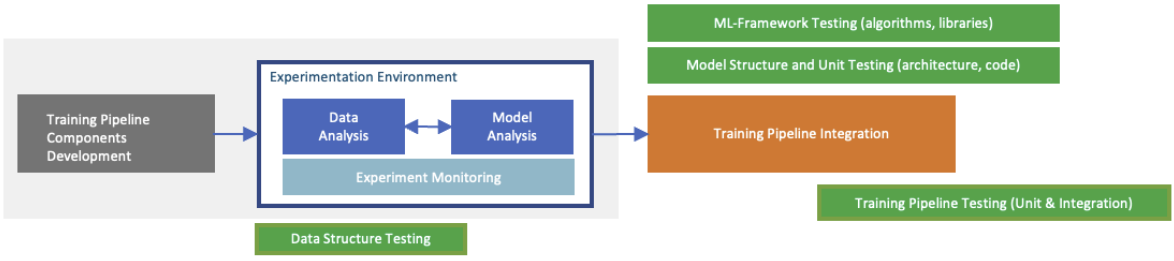

Figure 8 - Test activities in the phase of experimentation and training pipeline development.

The experimentation and training pipeline development phase consists of extensive activities in the area of Data Analysis and Model Analysis. The purpose of these activities is to identify suitable modelling approaches and data

preparation procedures that can be used to meet the KPIs and requirements derived from the first phase for the given data set. In the course of the activities, a suitable model architecture including layers and model code will be realized and the necessary software components for data preparation and training will be implemented and integrated into a functional pipeline. Major work products of this phase include the adequate data format for training data, samples of the training data, feature definitions and feature selection criteria, the model architecture and code as well as all algorithms, libraries, and components required for the training.

Data Structure Testing and Feature Testing: Data structure testing is a test activity in which syntactic properties of data and data sets are checked. These include the correctness and properties of data formats and data types, the metadata and its availability, and annotation formats for labels and other data annotations. Feature testing includes testing of feature relevance, compliance, ranges as well as tests for the general availability and costs for certain features.

ML-Framework Testing: ML-Framework Testing is considered an activity that tests the functionality, reliability, and scalability of the training environment. This includes testing of libraries that provide training algorithms like loss functions and optimizers as well as the code that allows to compose models out of predefined building blocks.

Model Structure and Unit Testing: Includes the test of the synthesized model structures and model code. This includes the code of the individual layers including their functions, the integration of the layers and the data flows and data type compatibilities between the layers as well as the integration of the model into the ML framework.

Training Pipeline Testing: This testing activity includes testing of all components that are part of the training process. This includes testing the relevant components for data gathering, data preparation, and feature generation/extraction, testing the ML-Framework and the model structure and code as mentioned above, and testing the monitoring and validation components, that are meant to safeguard the training process in the training phase. Testing covers all integration stages, starting with unit/component testing, through integration of individual components, to testing of the entire pipeline.

- *Experiment Monitoring*: Experiment monitoring is used to capture information gained during data and model analysis to ensure systematic decision making and traceability in the transition of POC models and infrastructures towards an efficient production environment.

Table 17 provides an overview on the major work products of the **experimentation and training pipeline development** phase, the related acceptance criteria and testing types.

Table 17: Work products, acceptance criteria and test types of the experimentation and training pipeline development phase

Work product/test item	Acceptance criteria	Test type/test objective
Training data format and samples.	– Quality criteria for data quality are completely defined. – Training data are suitable for purpose (training and inference) – Training data are available. – Training data are processable	– Review of data quality criteria. – Testing initial samples of training data for major data quality attributes. – Review of data sources and their availability. – Testing training data formats and metadata.
Features and feature selection criteria	– Features are identified. – Features are sufficient to allow for reliable inference. – Features are available in training and inference data.	– Redundancy – Ranking/Usefulness
Label structure and label adequacy	– Labels are identified.	– Review label structure and format. – Testing label completeness.

	– Label structure and format is adequate	– Testing label adequacy.
Model architecture, layers and algorithms	– The basic model architecture, layers and algorithms are defined and evaluated with the data that are available for training and inference.	– Review of architecture and layer interfaces.
Training algorithms (Loss Function, Optimizer), libraries and interfaces	– Algorithms used for training are working correctly. – Test the libraries and interfaces used for training and model set up are compatible with each other and the machine learning model being developed.	– Review of algorithms. – Code review. – Functional testing of algorithms and libraries. – Compatibility reviews and tests of training and library interfaces.
Model Code	– Model code is sufficiently tested with respect to training and inference capabilities and layer integration.	– Code review of model code. – Layer and sub model testing (unit testing). – Functional testing of model software behaviour during training and inference.
Hyperparameters	– Major hyperparameters are defined and tuned for the given data and model architecture.	– Cross Validation to test the performance of the model on different subsets of the data and with different hyperparameters.
Basic model performance	– ML-Model performance is sufficient as a candidate model for exhaustive training. – ML-Model is robust and generalizes well. – The ML-model is free of unwanted bias.	– Model performance testing and evaluation. – Model robustness testing. – Model bias testing.
Training pipeline components	– Functionality of the pipeline components. – Integration of the pipeline components. – Software-hardware embedding of the training pipeline.	– Unit/component testing of pipeline components (classical software testing). – Integration testing of pipeline components (classical software testing). – System testing of the training pipeline (classical software testing). – Testing of Software-hardware embedding (e.g. GPU integration) of the pipeline. – Test the API of the pipeline to ensure that it is easy to use and integrates well with other systems.

5.3.2.3 Test items of the training phase

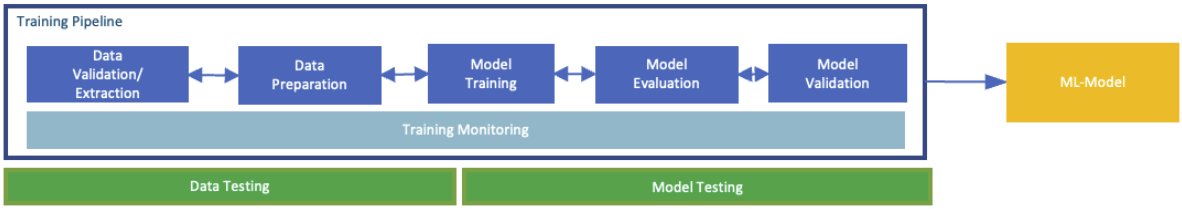

Figure 9 - Test activities in the phase of model training

The training phase is responsible for training ML models for production based on the modelling approaches and data preparation activities identified in the experimentation phase. If possible, this is done in an automated way and with the help of a predefined training pipeline. In the pipeline, all necessary activities from data validation and extraction, data preparation, model training, model evaluation, and model validation are performed. The result is the delivery of an ML model that best meets the requirements and KPIs from phase 1.

Data Testing: Data testing is an activity to detect errors in the data, the composition of the datasets, and the distributions of properties, features, or other characteristics in the data. Data testing can be very diverse and includes tests with different data compositions, statistical and structural analysis of the data, and monitoring of predefined KPIs for different quality characteristics of data.

Model Testing: Model Testing is the activity to identify deviations of the actual model performance from the expected model performance as well as to identify systematic errors in the model. This includes activities like measuring the accuracy and robustness by train/test split, cross validation, and other methods. Often there is an overlap in methods and approaches with the model evaluation and model validation phases. However, the latter are meant to select the best models and architectures from a given set of models, while model testing tries to check if the acceptance criteria for a given model is met.

Training Monitoring: is the activity to collect data during data preparation and training. These data are used to track dependencies (traceability) between data, hyperparameter settings and the resulting models. Moreover, these data can be used to continuously track quality related data and thus serves as a data source for localizing errors and track the state of certain quality attributes (e.g. number of training data failures and deviations etc.)

Table 18 provides an overview on the major work products of the **training** phase, the related acceptance criteria and testing types and test objectives.

Table 18: Work products, acceptance criteria and test types of the training phase

Work product/test item	Acceptance criteria	Test type/test objective
Training data	– Data and datasets are correct, – Data distribution and data splits are defined correctly. – Data are free of unwanted bias	– Test data format and type correctness. – Test data correctness and consistency. – Test data sets for missing data, duplicates, outliers, inconsistencies. – Test data set distribution and data skewness (e.g., any kind of imbalance regarding features and labels) – Test for correlated features. – Test data for unwanted bias
Hyperparameters	– Hyperparameters are fine-tuned	– Cross Validation to test the performance of the model on different subsets of the data and with different hyperparameters (e.g., different learning rates, batch sizes, regularizations, etc.).

ML-Model	– ML-Model performance is sufficient for production. – ML-Model is robust and generalizes well. – The ML-model is free of unwanted bias.	– Model performance testing and evaluation (i.e., evaluating various performance measures such as accuracy, precision, recall, F1-score, AUC-ROC, mean average precision, or any other relevant metrics specific to the problem domain) – Model robustness testing. – Bias and Fairness Assessment.
Evaluation concepts and criteria	– The evaluation concept and criteria are sufficient to ensure an adequate selection and evaluation of the candidate models.	– Review of evaluation concept and criteria.

5.3.2.4 Test items of the system development and integration

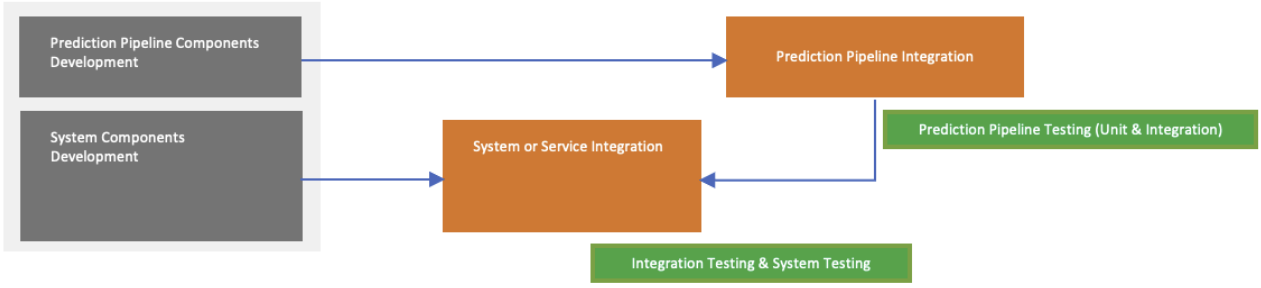

Figure 10 - Test activities in the phase of system development and integration.

In the system development and integration phase, the ML model is successively integrated into the software environment required for operation in production. As the first integration stage, we consider the integration of the model with software components that have a direct impact on the quality and performance of the model inference. This includes the integration of the model with the data sources for the inference (databases, user interfaces, sensors, etc), the data preprocessing components for the inference, and components that make it plausible or contextualize the result of the inference. We call the result of this integration the prediction pipeline. The model is then integrated with other system components until a complete system is available. The testing and quality assurance activities in this phase largely follow the established best practices of classical software testing.

Prediction Pipeline Testing (Unit & Integration): The prediction pipeline consists of the ML model, the software components that acquire, process, and feed data to the model, and the software components that directly interpret the model's prediction results. It can be assumed that especially the components of the prediction pipeline have a high degree of dependencies to each other. The test of these components takes place according to the strategies of the classical software testing by test of the individual components and the test of the integration as complete pipeline.

Integration Testing & System Testing: This activity aims to test all system components and its integration. Dependent on the definition of the system this varies from testing the prediction pipeline as mentioned above to arbitrary integrations of the prediction pipeline as part of a complex ML-based system (e.g. an automated car or train). Integration and system testing is carried out based on a given integration strategy based on best practices and approaches well known in software engineering.

Table 19 provides an overview on the major work products of the **system development and integration** phase, the related acceptance criteria and testing types and test objectives.

Table 19: Work products, acceptance criteria and test types of the system development and integration phase

Work product/test item	Acceptance criteria	Test type/test objective

Prediction pipeline	– ML model is correctly integrated in the prediction pipeline. – The prediction pipeline is correctly integrated with additional components e.g. safety mechanisms (safety cage, redundant models, plausibility checker etc.)	– Integration test (i.e., classical software testing).
ML-based system or component	– Prediction pipeline is integrated with the rest of the ML-based system. – Software-hardware embedding of the prediction pipeline and the ML-based system (model and data pre-processing or result preparation, GPU integration).	– Integration test (i.e., classical software testing). – System test (i.e., classical software testing). – Performance test for inference.
Acceptance testing	– The ML-based system complies with stakeholder requirements.	– Performance and stakeholder requirements testing. – Testing the compliance with given rules and regulations.

5.3.2.5 The test items of the operation and monitoring phase

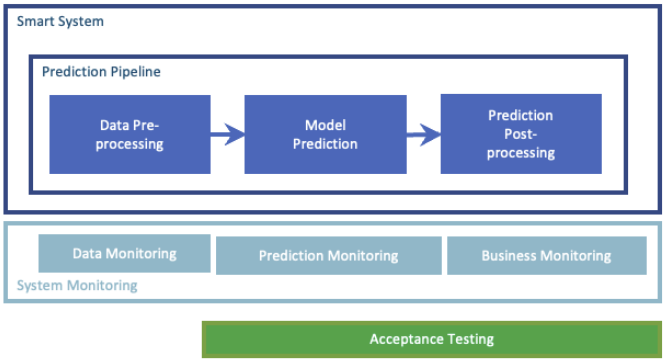

Figure 11 - Test activities in the phase of operation and monitoring.

For the operation and monitoring phase, the model is executed in its operating environment. Testing and monitoring activities shall ensure that the model functions safely in the application context and is not outdated. Depending on the assessed risk of the ML-based system during runtime, it is necessary to implement the execution of online testing (monitoring) of the system in operation. These tests go hand in hand with dedicated security and monitoring components that are supposed to identify corner cases and potential distribution shifts. As part of the system testing also the effectiveness of the online testing (monitoring) measures shall be verified.

Acceptance Testing: Acceptance testing for ML-based systems refers to the process of evaluating a trained machine learning model's performance when integrated within its software environment. Acceptance testing aims for determining whether an ML-based application meets the desired criteria and requirements established by stakeholders.

Data Monitoring: By monitoring the incoming data, it is possible to identify anomalies in the data stream, shifts in the data distribution and to detect concept drift. This allows to initiate special treatment of outliers and other anomalies and to re-evaluate assumptions on the data, update the model if needed, or trigger alerts for manual intervention.

Prediction Monitoring: Prediction monitoring enables you to track the performance of the model over time, detect any degradation in its predictive capabilities, and identify when it may need retraining or recalibration. By

monitoring the technical model's performance, it can be ensured that the ML-based application remains effective and allows to initiate timely adjustments if necessary.

Business Monitoring: Business monitoring aims to assess how well the ML-based applications are aligned with the business objectives and compliance rules. By monitoring business based key performance indicators (KPIs), it is possible to track the model's performance in context of the associated business or application environment and allows to evaluate the economic impact and value generated by the ML-based applications. Moreover, it allows for proactive risk management, ensuring compliance with legal and ethical standards and maintaining trust among stakeholders and customers.

Table 20 provides an overview on the major work products relevant in the **operations** phase, the related acceptance criteria and testing types and test objectives.

Table 20: Work products, acceptance criteria and test types of the operation and monitoring phase

Work product/test item	Acceptance criteria	Test type/test objective
ML-based system or component	End user accepts the model in production.	User Acceptance testing (e.g. A/B testing)
ML-model	Model is free from drift.	Monitor data drift between the training and testing sets to ensure that the model is still accurate and reliable over time. Monitor inference skew and bias

5.4 The IML4E MLOPS CABC Methodology

Continuous Auditing Based Certification (CABC) is a cutting-edge methodology designed to ensure that dynamic systems, particularly those involved in MLOps, consistently adhere to a wide range of standards such as ISO, ETSI, and sector-specific regulations. Unlike traditional certification processes that operate on fixed intervals, CABC emphasizes real-time, continuous assessment and certification, ensuring that systems maintain compliance throughout their entire lifecycle.

5.4.1 CABC Methodology

5.4.1.1 Core Principles of CABC

CABC is grounded in three core principles that guide its operation and implementation:

Continuous Assessment: CABC continuously monitors compliance using real-time data, making it a vital approach for AI systems that undergo frequent updates and iterations. This uninterrupted assessment allows for immediate detection and correction of any deviations from compliance standards, thereby ensuring that systems remain aligned with the required quality and regulatory benchmarks at all times.

Stakeholder Trust: Transparency is key in CABC, with regular updates provided to stakeholders through detailed reports. This openness not only supports regulatory compliance but also enhances the system's credibility among users, regulatory bodies, and other stakeholders by keeping them informed of the system's compliance status.

Adaptability: Given the rapidly evolving nature of AI systems and the external regulatory landscape, CABC is designed to be flexible, allowing it to adapt to changes in internal systems, industry standards, and legislative requirements. This flexibility ensures that AI systems remain compliant even as they evolve and new regulations emerge.

5.4.1.2 Roles and Architecture in CABC

CABC involves several key roles and procedures that ensure its effective implementation:

Auditee: The organization responsible for defining the scope of CABC and implementing the necessary technical measures to provide evidence of compliance.

Auditing Party: This entity conducts the audit, often involving automated assessments to evaluate the system's adherence to defined metrics and requirements.

Certification Status Publishing Entity: This role involves communicating the current certification status of the AI system to stakeholders, ensuring transparency and trust in the certification process.

5.4.1.3 Modes of CABC

CABC can be implemented in three modes to accommodate different organizational needs:

Self-Assessment Mode: Organizations conduct internal audits, continuously evaluating compliance with legislative requirements and quality goals. This mode is particularly useful for early detection and correction of potential compliance issues.

Third-Party Audit Mode: An external accredited body performs the audit, providing an impartial assessment of the organization's adherence to the set standards, enhancing the credibility of the audit findings.

Certification Mode: This mode involves a comprehensive evaluation by recognized certification bodies, assisting organizations in obtaining formal certifications that validate their adherence to established standards and regulatory requirements.

5.4.2 Operationalization

5.4.2.1 Steps for Operationalization

The operationalization phase in CABC involves transforming high-level Certification Specifications into actionable steps and machine-readable metrics. This process begins by identifying *Quality Dimensions* relevant to the system, such as fairness, transparency, reliability, privacy, and security. Each dimension is then segmented into smaller, manageable components, ensuring the assessment framework remains adaptable to technological and business needs.

1. **Identify Quality Dimensions**: Defining quality dimensions is crucial for structuring the assessment of different aspects of the AI system, such as fairness, transparency, and security. These dimensions are broken down into specific, actionable components.
2. **Risk Identification**: Risks associated with each quality dimension are identified and analyzed. This step is essential for developing targeted mitigation strategies that are traceable back to their respective risks.
3. **Define Requirements and Metrics**: Requirements derived from the identified risks must be clear, measurable, and relevant. For each requirement, specific metrics are established to quantify compliance. These metrics are continuously assessed at frequencies appropriate to the requirements.
4. **Implement Measurements**: A framework for continuous, automated assessment is established, enabling the system to measure compliance efficiently. The measurements are aligned with the defined metrics and systematically collected and stored for auditing purposes.

5.4.2.2 Operationalization Documentation

Documentation is critical in informing stakeholders about operational decisions and configurations. It functions as a configuration file for automated assessments and supports nested operationalization for systems involving multiple vendors. This documentation ensures transparency, traceability, and configurability, allowing stakeholders to review and confirm that the operationalization of conformity specifications meets their quality standards.

The operational documentation is structured hierarchically as follows:

1. **Conformity Specification**: A high-level document that specifies required types of conformity.
2. **Dimensions**: These are subcomponents of the Conformity Specification that detail quality dimensions.

3. **Requirements**: These are specific criteria derived from quality dimensions and associated risks.
4. **Metrics**: These are standards established to measure compliance with the requirements.
5. **Measurements**: Measurements need to be designed as a flexible and extensible framework for measuring various types of data and comparing these measurements against specified targets. This system accommodates different data types (numeric, ordinal, boolean, string) and supports multiple comparison strategies, enabling a unified and versatile approach to handling diverse measurement needs. By defining a generic measurement type and implementing specific comparison strategies, the framework ensures that measurement values can be accurately assessed against target values regardless of their underlying data type.

Attributes of Operationalization Documentation

The following tables detail the attributes of each hierarchical element in the operationalization documentation:

Table 21: Attributes of Conformity Specification

Attribute Name	Data Type	Description
Name	String	Name of the Conformity Specification
Comment	String	Provides explanations or additional information
ConfidentialityFlag	Boolean	Indicates if the information is confidential
Assessor	String	Entity responsible for assessing the specification
AssessmentInterface	String	Where to fetch the assessment result if Assessor is not None

Table 22: Attributes of Dimensions

Attribute Name	Data Type	Description
Name	String	Name of the Dimension
Comment	String	Descriptive text regarding the dimension
ConfidentialityFlag	Boolean	Flag to mark if dimension details are confidential
Assessor	String	Entity responsible for assessing the dimension
AssessmentInterface	String	Where to fetch the assessment result if Assessor is not None

Table 23: Attributes of Requirements

Attribute Name	Data Type	Description
Name	String	Name of the Requirement
Comment	String	Explanation or commentary on the requirement
ConfidentialityFlag	Boolean	Indicates confidentiality of the requirement details

Attribute Name	Data Type	Description
Frequency	String	Specifies the frequency of requirement evaluation
Assessor	String	Entity responsible for assessing the requirement
AssessmentInterface	String	Where to fetch the assessment result if Assessor is not None

Table 24: Attributes of Metrics

Attribute Name	Data Type	Description
Name	String	Name of the Metric
Comment	String	Provides details about the purpose and use of the metric
ConfidentialityFlag	Boolean	Indicates if the metric includes confidential data
Assessor	String	Entity responsible for assessing the metric
AssessmentInterface	String	Where to fetch the assessment result if Assessor is not None

Table 25: Attributes of Measurements

Attribute Name	Data Type	Description
Name	String	Name of the Measurement
Comment	String	Additional information on how measurements are conducted
ConfidentialityFlag	Boolean	Specifies if measurement details are to be kept confidential
Assessor	String	Entity responsible for performing the measurement
AssessmentInterface	String	Describes how to retrieve the value
ComparisonStrategy	String	The strategy used for comparing values
DataType	Enum	The type of data being measured
TargetValue	Generic	The target value to compare against

In all cases, a “None” value in the “Assessor” attribute would mean no external assessor, while the “AssessmentInterface” is optional. In Table 25, the “AssessmentInterface” can be a REST endpoint, CLI command, database query, or file location. Similarly, the “ComparisonStrategy” could be “LessThan,” “Contains,” “GreaterThan,” or “EqualTo.” “DataType,” on the other hand, could be numeric, ordinal, boolean, or string, while the “TargetValue” depends on the measurement type.

This hierarchical and comprehensive approach to documentation and operationalization ensures that CABC can effectively and continuously evaluate AI systems against relevant standards and guidelines. The result is machine-readable documentation that not only facilitates automated assessments but also provides transparency for

stakeholders, allowing them to verify that the operationalization aligns with their quality requirements and expectations.

5.4.2.3 Measurement Framework

The measurement framework within CABC is designed to handle a wide array of data types, including numeric, ordinal, boolean, and string values. Each data type is paired with appropriate comparison strategies to evaluate measurement values against target values effectively. These comparison strategies include methods such as LessThan, GreaterThan, EqualTo, Contains, NotEqualTo, and InRange, providing a flexible and robust system for comparing data. The framework's design ensures accurate assessments regardless of the data type by implementing a generic measurement type that applies the relevant comparison strategy.

Core Components of the Measurement Framework

The measurement framework is structured around several key components, each of which plays a vital role in ensuring that the assessment process is both flexible and precise:

1. **Measurement Interface**: This interface serves as the foundational layer of the framework, defining the properties and methods necessary for any measurement. It includes the value being measured, the strategy used to compare this value to a target, and the interface through which the measurement value is retrieved.
2. **Comparison Strategy**: A crucial component of the framework, the comparison strategy interface defines how different data types are compared against their respective target values. The strategies ensure that measurements are evaluated consistently and accurately, regardless of the underlying data type.
3. **Assessment Interface**: This interface specifies the method through which measurement values are obtained, ensuring that the data used for comparisons is collected in a standardized and reliable manner. This could involve retrieving data from various sources, such as REST endpoints, database queries, or file systems.

Expanded Functionality of the Measurement Framework

To accommodate the diverse requirements of AI systems and the varying contexts in which they operate, the measurement framework has been designed to be highly extensible. It supports the addition of new data types and comparison strategies as needed, ensuring that the framework can evolve alongside the systems it evaluates. For instance:

1. **Custom Data Types**: If a system introduces a new type of data that is not covered by the standard types (numeric, ordinal, boolean, string), the framework can be extended to include this new data type. This ensures that all relevant aspects of the system's performance and compliance can be accurately measured.
2. **Advanced Comparison Strategies**: In addition to the basic comparison strategies like LessThan, GreaterThan, and EqualTo, the framework allows for the implementation of more complex strategies such as pattern matching, range checks with tolerance levels, and custom algorithms tailored to specific compliance requirements.

Integration with Other Systems

The measurement framework is designed to be interoperable with other systems and tools used within the CABC methodology. This includes integration with logging systems, data storage solutions, and real-time monitoring tools. By facilitating seamless data flow between these systems, the framework ensures that measurements are always based on the most current and accurate data available. This interoperability also supports the automated generation of compliance reports, making it easier for stakeholders to understand the system's current status and any areas that may require attention.

Pseudocode for the Measurement Framework

The pseudocode below represents the core structure of the measurement framework. It focuses on the interfaces and the function that compares measurement values to target values, illustrating how different components of the measurement system interact within the framework.

```
// Interface for all measurements
interface MeasurementInterface:
    properties:
        Value  // The value of the measurement
        ComparisonStrategy  // Strategy used for comparison
        AssessmentInterface  // Method to retrieve the value
    methods:
        MeetsTarget(targetValue)  // Returns true if the measurement meets
                                  // the target based on the strategy

// Interface for comparison strategies
interface ComparisonStrategy:
    methods:
        Compare(value, targetValue)  // Defines the comparison logic

// Generic method to compare measurement values
function CompareMeasurement(measurement, targetValue):
    return measurement.MeetsTarget(targetValue)
```

The pseudocode begins by defining a `MeasurementInterface` that includes properties for the measurement value, the comparison strategy, and the assessment interface. It also defines a method, `MeetsTarget(targetValue)`, which checks if the measurement meets the specified target value using the comparison strategy. Next, the `ComparisonStrategy` interface is defined, which includes the method `Compare(value, targetValue)` to encapsulate the logic for comparing the measurement value to the target value. Finally, the function `CompareMeasurement(measurement, targetValue)` is provided to facilitate the comparison process. This function invokes the `MeetsTarget` method of the measurement object, which uses the appropriate comparison strategy to determine if the measurement value satisfies the target criteria.

Scalability and Performance Considerations

As AI systems grow in complexity and scale, the measurement framework must remain efficient and capable of handling large volumes of data without compromising performance. To address this, the framework is optimized for scalability, ensuring that it can process numerous measurements in parallel and integrate seamlessly with high-performance data processing pipelines. This includes:

1. **Parallel Processing**: The framework supports parallel execution of measurement evaluations, allowing it to handle multiple assessments simultaneously. This is particularly important for large-scale AI systems where compliance checks must be performed across numerous components in real-time.
2. **Caching and Optimization**: To minimize the overhead associated with repetitive calculations, the framework includes mechanisms for caching results of frequently accessed measurements. This reduces the computational load and improves the overall efficiency of the assessment process.

Continuous Improvement and Adaptation

The measurement framework is not static; it is designed to evolve alongside the AI systems it assesses. As new challenges and compliance requirements emerge, the framework can be updated with additional comparison strategies, data types, and assessment methods. This continuous improvement ensures that the framework remains relevant and effective in an ever-changing technological landscape.

5.4.3 Continuous Assessment

5.4.3.1 Continuous Assessment and Reporting

CABC is structured to deliver ongoing evaluations of AI systems, ensuring they meet conformity specifications, quality objectives, and effectively manage risks. The continuous assessment process is meticulously designed to ensure that every aspect of the AI system's operation is consistently evaluated against the relevant standards and guidelines.

The assessment process in CABC is deeply integrated with the *Operationalization Documentation*, which acts as a configuration tool. This documentation outlines the dimensions, requirements, metrics, and measurements necessary for continuous assessment, facilitating the automatic interpretation of audit criteria and corresponding measurements. The frequency of assessments varies based on the specific requirements of each operational component and is defined within the configuration file.

The assessment process is triggered under several conditions, most notably whenever a new model is deployed. During the operational phase, assessments are conducted at predefined intervals or in response to specific events, ensuring that the system continuously adheres to the required standards.

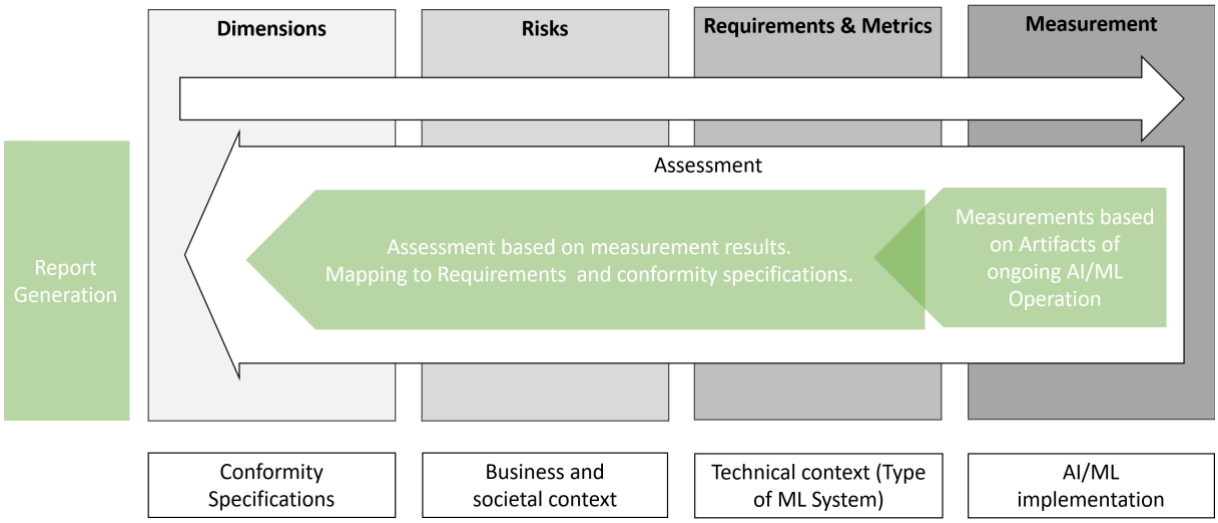

Figure 12 - Assessment Process

Assessment Process

The CABC assessment process involves several key steps, as illustrated in the figure (Figure 12). These steps ensure that the assessment is comprehensive, and that all relevant data is captured and evaluated systematically:

1. **Artifacts Production and Usage**:
   Artifacts such as log files, model weights, and data samples are generated or utilized at different stages of the machine learning lifecycle. These artifacts serve as the primary inputs for the measurement process. Some artifacts provide direct measurement results through parsing, while others may require more complex test suites to accurately extract relevant data. This step corresponds to the "Measurements based on Artifacts of ongoing AI/ML Operation" in Figure 12.
2. **Measurement Using Artifacts as Inputs**:
   Measurements are conducted by analyzing the artifacts produced during the system's operation. The results of these measurements are then prepared for transmission to the auditing entity. This step supports both real-time and scheduled evaluations, triggered either by specific cycles such as model deployment or by the intervals specified for individual quality requirements. This aligns with the "Measurement" phase in Figure 12.
3. **Mapping Measurement Results**:
   The measurement data is automatically evaluated using the Operationalization Documentation as a

configuration tool. This evaluation process involves checking the measurement results against predefined metric thresholds that reflect the system's quality objectives. The automated nature of this evaluation ensures that the process is not only rapid but also consistently aligned with the quality goals outlined in the Conformity Specifications. This step is depicted in the "Mapping to Requirements and Conformity Specifications" in Figure 12.

4. **Report Generation**:
   After mapping the measurement results, a detailed report is generated. This report provides comprehensive information on how well the AI system meets the specified quality objectives by correlating each measurement result with the corresponding parts of the Conformity Specifications. The report is tailored to the needs of different stakeholders, offering varying levels of detail—from high-level overviews for executives to in-depth technical reports for engineers and auditors. This final step corresponds to the "Report Generation" phase in Figure 12.

Persistence of Assessment Results

The results of each CABC assessment are stored persistently, ensuring that a comprehensive record is maintained for future reference, quality tracking, and for demonstrating compliance during audits. The scope of this persistence includes:

1. **Measurement Results**: All metrics and their corresponding values are systematically stored. This data includes critical indicators such as system performance, fairness measures, and other parameters essential for evaluating the system's alignment with the specified standards.
2. **Evaluation Outcomes**: The results of comparisons between metrics and predefined quality goals are archived, providing a detailed record of the system's compliance status over time.
3. **Assessment Reports**: The final reports generated after each assessment are saved, ensuring that stakeholders have access to a transparent and comprehensive record of the system's compliance status.

5.4.3.2 Updating Conformity Status

CABC emphasizes the importance of regularly updating the conformity status of AI-enabled systems. This process involves continuous assessment of the AI system's compliance with set standards and quality goals, as well as regular updates to these assessments based on new data, evolving risks, and changes in regulatory requirements.

The conformity status is updated based on the results of quality assessments, which include evaluating measurement results from artifacts such as log files and model weights against predefined values. These predefined values are informed by expert knowledge and risk assessments. If the measurement results match the predefined values, a conformity status is issued, confirming that the system is compliant. If the results do not match, the conformity status is either revoked or adjusted to reflect the need for improvement, highlighting areas that require further attention and better risk management.

Regular updates to quality requirements are also crucial, as changes in these requirements can significantly impact the assessment outcomes. This adaptability to technological changes ensures that the AI system remains compliant in a dynamic regulatory environment.

5.4.3.3 Risk Mitigation and Traceability

Effective risk management is integral to the continuous assessment process in CABC. By identifying and addressing risks associated with each quality dimension, CABC enables the development of targeted mitigation strategies that are traceable back to their respective risks. This traceability ensures that any changes in risk levels or the emergence of new risks are promptly identified and managed, maintaining the system's alignment with its quality objectives.

The continuous nature of CABC's assessment process allows for real-time adjustments to risk management strategies, ensuring that the AI system remains resilient against both anticipated and unforeseen risks.

5.4.3.4 Metrics and Measurements

Metrics are essential for quantifying compliance with the requirements derived from identified risks. In CABC, specific metrics are established for each requirement, and these metrics are continuously assessed using a flexible and extensible measurement framework. This framework, as detailed in the Measurement Framework section, supports various data types (numeric, ordinal, boolean, string) and multiple comparison strategies (e.g., LessThan, GreaterThan, EqualTo) to ensure accurate and consistent assessments across different scenarios.

The ongoing evaluation of these metrics provides a clear and measurable understanding of the AI system's compliance status, enabling timely interventions when necessary. The results of these evaluations are systematically documented, ensuring that all stakeholders have access to up-to-date information about the system's performance and compliance with established standards.

5.4.4 Certification

Certification in Continuous Auditing Based Certification (CABC) is a dynamic and ongoing process designed to ensure that AI systems consistently meet established standards and guidelines. This section delves into the foundational aspects of certification within CABC, including the creation and application of certification specifications, as well as the communication of certification status to stakeholders.

5.4.4.1 Certification Specifications

Certification specifications form the cornerstone of the CABC process, providing a structured set of requirements that AI systems must adhere to. These specifications can be derived from a broad array of sources, such as international standards (e.g., ISO/IEC), national regulations, industry-specific guidelines, and internal organizational policies. In some cases, these standards may also incorporate ethical guidelines or sector-specific regulations, such as those in healthcare, finance, or defence.

Given the diverse origins of these specifications, they often vary in their level of abstraction. Some might be high-level principles requiring detailed interpretation, while others could be specific, actionable requirements. During the operationalization phase, these specifications are meticulously broken down into actionable steps, metrics, and measurable criteria that the AI system must meet to be certified. This breakdown ensures that the certification process is both rigorous and adaptable to the evolving nature of AI systems and the environments in which they operate.

Furthermore, the certification specifications are designed to be continuously auditable, allowing for real-time assessments of compliance. This continuous auditing capability is essential in dynamic fields like machine learning operations (MLOps), where changes and updates to the system occur frequently. The specifications must, therefore, be both comprehensive and flexible, capable of guiding the system through various stages of development and deployment while ensuring ongoing compliance with the relevant standards.

5.4.4.2 Publishing Certification Status

The publication of an AI system's certification status is a critical aspect of the CABC framework, serving as a communication channel between the certification body and stakeholders. After an AI system undergoes the certification process and is found to be in compliance with the established specifications, its certification status is formally documented and disseminated to relevant parties. This includes regulatory bodies, customers, partners, and other stakeholders who rely on the certified status to gauge the system's adherence to quality and safety standards.

The certification status is not static; it is regularly updated to reflect any changes in the system's compliance. For instance, if a system undergoes significant updates or modifications, a re-assessment may be required to ensure continued compliance. This dynamic updating process is crucial in maintaining the credibility and reliability of the certification. The frequency and nature of these updates depend on the specific operational context of the AI system and the associated risks.

In addition to providing transparency, publishing the certification status enhances trust in the AI system. Stakeholders can have confidence that the system operates within the bounds of the agreed-upon standards and

that any deviations are promptly addressed and communicated. This ongoing publication also supports accountability, as it ensures that any lapses in compliance are made visible and can be acted upon swiftly.

6 Conclusions and Summary

Within this document, the IML4E project provides a comprehensive overview of the IML4E framework, which is designed to support the lifecycle management of machine learning (ML) products within enterprises. The IML4E framework integrates methodologies, tools, and technologies that facilitate the governance, auditability, and operationalization of ML models. The document details the IML4E framework's three key layers:

- the methodology layer, which provides foundational support for MLOps implementation;
- the technology layer, which offers technical solutions for specific MLOps challenges;
- and the platform layer, which includes the IML4E OSS platform, an open-source platform that serves as a reference architecture for MLOps.

The IML4E framework and its components aims to streamline the development, deployment, monitoring, and maintenance of ML models by introducing best practices and tools that enhance collaboration between data scientists, engineers, and other stakeholders. Additionally, the document addresses AI ethics in MLOps, emphasizing the need for ethical considerations throughout the ML lifecycle. The IML4E MLOps testing methodology is also outlined, providing a structured approach to ensure the quality and reliability of ML-based systems.

Overall, this deliverable serves as a final specification of the IML4E MLOps framework, offering a detailed guide for enterprises looking to implement robust and scalable ML operations. The framework not only supports the technical aspects of MLOps but also promotes continuous assessment and improvement, ensuring that ML models remain effective and compliant with organizational and regulatory requirements.

References

Google (2022). MLOps: Continuous delivery and automation pipelines in machine learning, https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Microsoft (2022). Machine learning operations maturity model. [Online Available] https://learn.microsoft.com/en-us/azure/architecture/example-scenario/mlops/mlops-maturity-model

Delta (2023), Build Lakehouses with DElta Lake, https://delta.io/

Feast (2023), Open Source Feature Store for production ML, https://feast.dev/

Valohai (2022), The MLOps Stack, https://valohai.com/blog/the-mlops-stack

Sadiq, R.; Safie, N.; Abd Rahman, A. & Goudarzi, S . (2021). Artificial intelligence maturity model: a systematic literature review Peerj computer science, 7, 1-27

John, M. M.; Olsson, H. H. & Bosch, J. (2021) Towards mlops: a framework and maturity model 2021 47th euromicro conference on software engineering and advanced applications (seaa), IEEE, 1-8

Zhang, J. M., Harman, M., Ma, L. & Liu, Y (2019). Machine Learning Testing: Survey, Landscapes and Horizons. arXiv:1906.10742.

IML4E project: IML4E-D4.1-V1.0-Requirements for the IML4E online experimentation and training platform

IML4E project: IML4E-D4.2-V1.0-Initial version of the IML4E framework

IML4E project: IML4E-D2.2-V1.0-First version of methods and techniques for data collection, processing, and valorisation

IML4E project: IML4E-D2.3-V1.0-First version of tools for data collection, processing, and valorisation

IML4E project: IML4E-D2.4-V1.0-Second version of methods and techniques for data collection, processing, and valorisation

IML4E project: IML4E-D2.5-V1.0-Second version of methods and techniques for advanced model engineering

IML4E project: IML4E-D3.2-V1.0-First version of methods and techniques for advanced model engineering,

IML4E project: IML4E-D3.3-V1.0-First version of tools for advanced model engineering

IML4E project: IML4E-D3.4-V1.0-Second version of methods and techniques for advanced model engineering

IML4E project: IML4E-D3.5-V1.0-Second version of tools for advanced model engineering

Annex A: Mapping of the IML4E methods, tools and techniques to MLOps capabilities

	Mosquito data cleaner	Privacy-friendly Image Preparation	Data testing tool suite	MLOps and monitoring platform for high number of ML models	SAGED Error Detector	VALICY	Autonomously Adaptive Pipeline	Calibrated confidence estimator	Cost efficient ML	Inference Scaling	Model cards toolkit	Validation of pose estimation models	Pipeline Probe	Stevedore wrapper	Maturity Assessment	Continuous Auditing Based Certification for MLOps	OSS Platform	IML4E Framework	Data Quality Dashboard
	Basware	BUTE	FhG	Granlund	SAG	Spicetech	UH	UH	UH	UH	UH	Vitarex	FhG	Reaktor	FhG	FhG	Silo	FhG	SAG
Model Development and Lifecycle Management																			
Experiment Tracking				x			x				x	x					F		
Model and Artifact tracking				x			x				x	x					F		

Model performance validation				x			x					x						
Model deployment flexibility							x			x							X	
Model training automation				x			x										X	
Monitoring and Operations																		
Performance Monitoring:				x			x	X					X			X	X	
Logging and Auditing:				x			x				x	x	X			X	X	
Business Monitoring & Feedback:							x	x	x		x		X			X		
Data Engineering																		
Data Ingestion	x													(x)				X
Data Version Control												x						X

Data Preparation	x			x	x							x						X
Feature Provision and Reuse																		X
Data Validation & Quality Monitoring	x		x	x											x			X
Scalability and Infrastructure Management																		
CI/CD Pipelines:				x								x		x				
Containerization and Virtualization:				x			x			x				x			x	
Model Serving Flexibility							x			x				x				

Distributed Training				x			x										x		
Collaboration & Guidance																			
Collaboration Support				x									x	x	x				