IML4E

Industrial Machine Learning for Enterprises

Deliverable D4.2

Initial MLOps methodology and the architecture of the IML4E framework

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

Work package:	WP4
Deliverable number:	D4.2
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	November 30th, 2022
Actual delivery date:	January 26th, 2023
Responsible partner:	SILO AI

Contributors

Editor(s)	Harry Souris (Silo AI)
Contributor(s)	Jürgen Großmann (Fraunhofer FOKUS), Johan Himberg (Reaktor)
Quality assuror(s)	Ihasalo Heikki (Granlund), Dorian Knoblauch (Fraunhofer Fokus)

Version history

Version	Date	Description
1.0	23-01-26	Final version for publication

Abstract

The purpose of this document is to introduce the first version of the IML4E MLOPs methodology that aims to offer governance and auditability to ML products. In addition, purpose of the document is to describe the MVP version of 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 initial version of the IML4E framework. IML4E framework describes methodologies that will govern the life cycle 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.

In regards to the tools introduced, this document describes the implemented Operation Support System (OSS) which is an experimentation (MLOps) platform where machine learning products can be developed and executed. The OSS is a collection of well-established open-source tools. The overall architecture that describes how the open-source tools are interfacing for creating the MLOps platform is also presented. To that architecture this document introduces the "governance" component that aims to be an enabler for the continuous monitoring and validation of ML systems.

In regards to the other parts of the IML4E framework, a first high-level description to the MLOps maturity assessment tables is introduced. Those tables will be an enabler for enterprises to define their MLOps capability roadmap. Lastly, the continuous audit framework for the ML products is introduced, a framework that will enable control mechanisms on the operation of ML models.

Table of contents

1 Introduction

1.1 Role of this Document

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

Part of the IML4E framework is the Operation Support System (OSS) (also referred as MLOps platform) that is the technical layer where ML products are executed. This document elaborates on the implemented OSS platform. The basic components of the platform are discussed and link to the source code is provided (Section 4). The platform is a collection of open-source tools and a small description of the role of each tool is given. Part of the work conducted was the implementation of the interfacing between the different open-source tools and small examples of executing ML pipelines on top of the OSS are provided in our code repository.

In addition, the IML4E maturity assessment is introduced. This assessment aims to become a practical tool to enterprises focusing on their ML/AI and data capabilities. Different stages (levels) of integration of technologies, methods and people are discussed when it comes to the data and ML utilization, aiming to help enterprises identify their level of automation and unification of their ML and data projects.

Lastly, the continuous audit mechanism for ML systems -which is also part of the IML4E framework- is also introduced. The continuous audit mechanism will become an enabler to define quality expectations and control metrics to the operations of ML products. In software systems audit mechanisms have proven effective and their adoption in ML operations can increase the level of trust in ML solutions

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

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

Table 1 – Contractual documents

2 IML4E High-level components

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.

2.1 IML4E Framework

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 technologies and experimentation environment" and a *methodology lay*er called "MLOps principles and methodology" as shown in Figure 1.

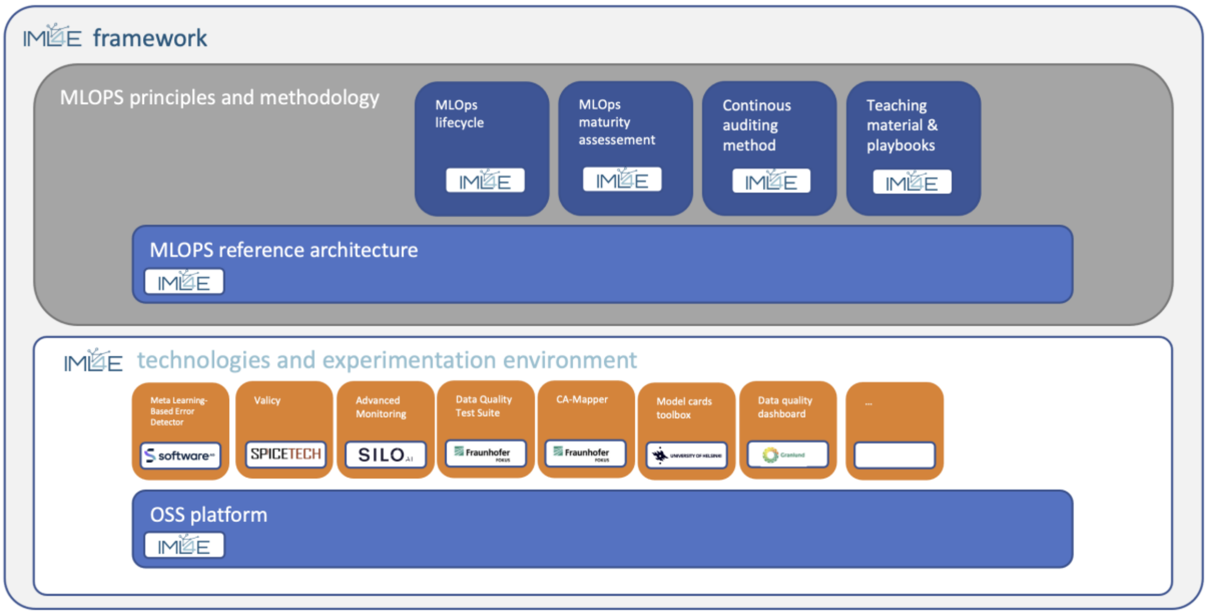

IML4E framework
MLOPS principles and methodology
MLOps lifecycle
MLOps maturity assessement
Continous auditing method
Teaching material & playbooks
MLOPS reference architecture
IML4E technologies and experimentation environment
Meta Learning-Based Error Detector
Valicy
Advanced Monitoring
Data Quality Test Suite
CA-Mapper
Model cards toolbox
Data quality dashboard
...
software AG
SPICETECH
SILO.AI
Fraunhofer FOKUS
University of Helsinki
Granlund
OSS platform

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. The individual technical solutions are outlined in the following section. 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. The IML4E reference architecture and the OSS platform are outlined in the Section 3. The requirements that the refence architecture is addressing are described in the first deliverable of work package 4 with a title "Requirements for the IML4E online experimentation and training platform" (IML4E 2022c)

2.2 IML4E Technologies

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 in Table 2. Details are available in (IML4E 2022a, IML4E 2022b).

Name	Provider(s)	Topic(s) Covered	WP
Meta learning-Based Error Detection	Software AG	Data Preparation Automation	WP2
Data quality dashboard for continuous monitoring of large data volumes	Granlund	Data Preparation Automation	WP2
Automatic removal of human errors from dataset with anomaly detection technique	Basware Oy	Data Preparation Automation	WP2
Privacy-friendly Image Preparation for AI Pipelines	Budapest University of Technology and Economics	Data Preparation Automation	WP2
Data Quality Test Suite	Fraunhofer Fokus	Data Quality, Automated Testing, CABC	WP2
Data Version Control	Silo AI	Data versioning	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
CABC-Mapper	Fraunhofer Fokus	Model training, MLOps lifecycle	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

Table 2 – IML4E Technologies Overview

2.3 The IML4E reference architecture

The IML4E reference architecture provides an outline of the basic building blocks and their interrelationship that are required to set up an MLOps infrastructure. Figure 2 shows an initial sketch of such an architecture. It is based on the terminology described by Google employees (Google, 2022a) in their white paper and an architecture extracted by Valohai (Valohai, 2022) to derive a so-called MLOps stack template. The template distinguishes data management components colored in green and pure operational components colored in blue.

In IML4E we will use this stack as a basis to describe a reference architecture based on it, which further specifies the meaning of the individual architecture elements and describes the capabilities to be expected from each element, so that this description can be used for tool selection. Currently, this reference architecture distinguishes the following 9 components.

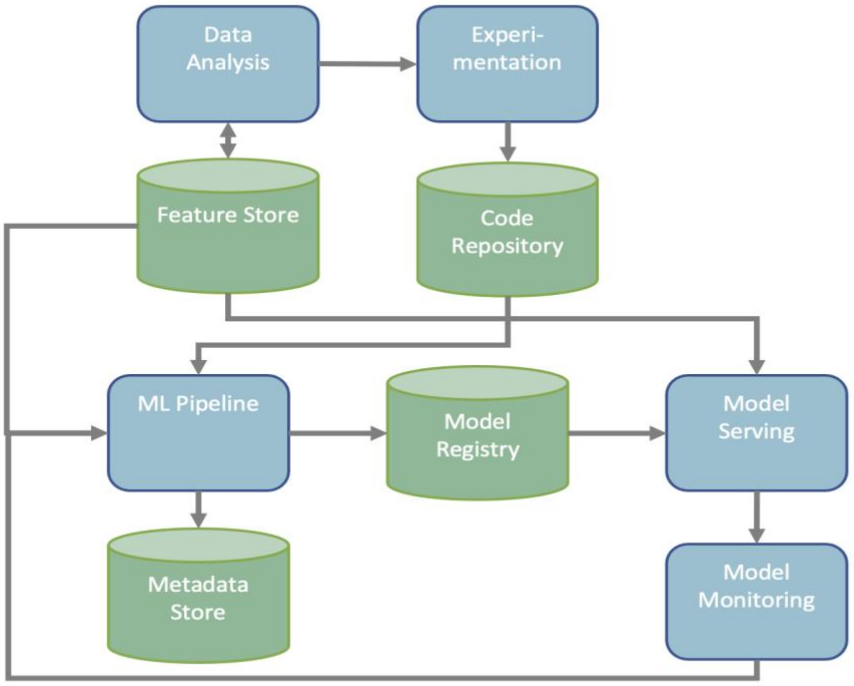


Figure 2 – MLOps reference architecture (Valohai, 2022)

- **Data Analysis Component:** Encompasses tools and infrastructures that are used to understand the available data an identify their potential for the business case.
- **Experimentation:** Encompasses tools and infrastructures that are used to in the initial experiments to optimize the overall training process by experimenting with new features, implementations, architectures, and hyperparameters.
- **Feature Store**: Is a centralized repository with standardized access of features for training and serving. It builds the bridge or interface between the data preparation process and often multiple training and inference processes in an organization.
- **Code Repository: Encompasses one or more repositories that are used to** manage coding artifacts including the model code, the pipeline code and the code that may be used in data preparation and pre-processing.
- **ML Pipeline:** Provides the infrastructure for the operationalization and automation of the training process. This includes automation in data splitting and batching, automation in hyper parameter evaluation and optimization as well as training operation, tracking and repetition.
- **Model Registry:** Is a data management infrastructure that allows model management for deployment. This includes storing and versioning of models and their documentation in a consistent and traceable manner.
- **Model Serving: Provides an** infrastructure to serve the model to dedicate operational environments. Dependent on the application requirements serving might target one or different operation environments ranging from micro service architectures in cloud environments to complex, distributes edge infrastructures.
- **Model Monitoring**: Includes a set of tools that provide feedback on model performance and fails in operation. It consists of elements that collect, stores, manages and visualizes the feedback and provides means to effectively organize the communication between the different stakeholders from business, operation and development.
- **Metadata Store:** Is a data store that allows to store relevant metadata on pipeline operation for the purpose on reproducibility and traceability in the training process.

To the above reference architecture IML4E aims to introduce a new component named as “model governance” component. Purpose of new component is to provide an interface to model owners to apply continuous validation and auditing on the development and operations of ML products. The interfaces of this component towards the basic components of the MLOPs reference architecture are going to be defined and described in the future iterations of the project. The new component is visualized with the red color in Figure 3.

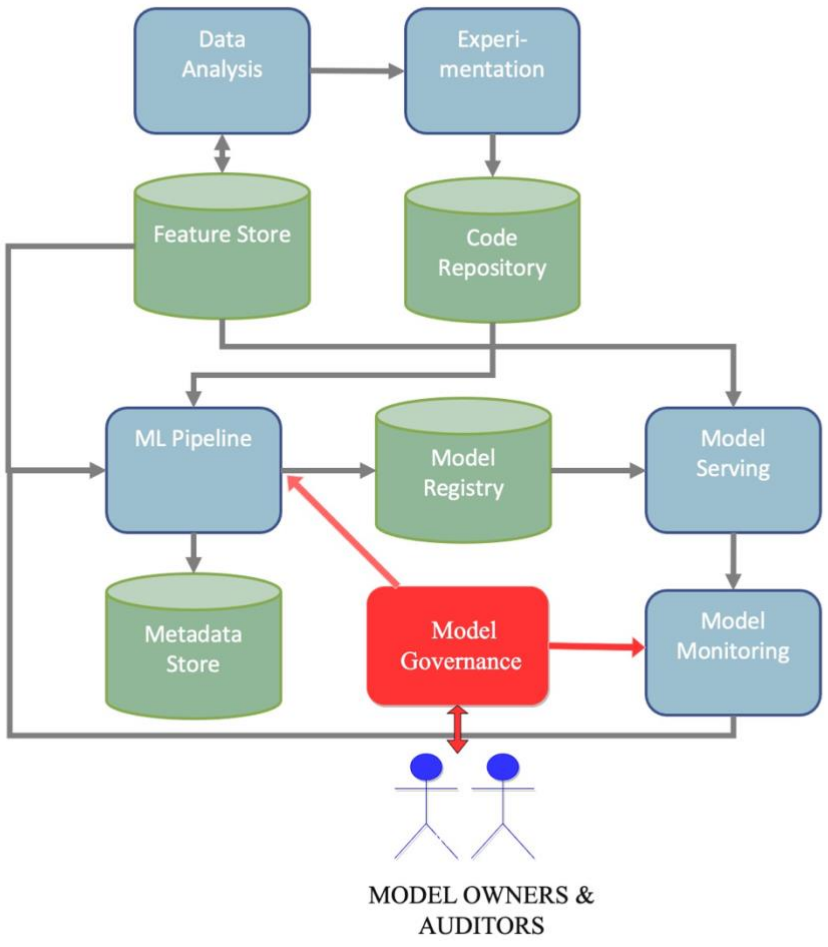


Figure 3 – IML4E proposed addition to the MLOps reference architecture.

3 IML4E experimentation and OSS platform

3.1 First's MVP Scope for the OSS platform

Scope of the first MVP of the IML4E OSS platform is to deliver software that enables the deployment of a generic MLOps reference architecture, on local machines and compute clusters. Outside of the scope of the MVP was the deployment and operation of such a platform for the IML4E partners and the coverage of the requirements related to data versioning (WP2). More information regarding the scope of the MVP of the OSS platform can be found in the first deliverable of Work Package 4. Future releases of the IML4E experimentation platform will include the governance component that needs to be analysed further during the progress of the project. Finally, future releases will describe how data versioning could be achieved utilizing open source technologies.

3.2 Requirements fulfilled in the first MVP

The first MVP of the OSS platform addresses the requirements of enabling:

- a scalable compute layer for the operation of the ML pipelines
- ML experiment comparison and tracking
- versioning capabilities to the executed ML pipelines
- automation and orchestration to the execution of ML workloads
- inference and monitoring services to the ML pipelines

The current OSS solution is not providing services for data versioning and advanced monitoring but expects those features to be met as the work package 2 progresses.

The Section 5 introduces the technologies that were used to address the basic requirements. In addition, it provides a link to a source code repository with the necessary instructions for deploying the OSS platform on local machines.

3.3 IML4E OSS technologies stack implemented

Following the directions of the first deliverable of Work Package 4, the proposed OSS platform that developed during the third quarter of 2022 includes the utilization of well-established open-source tools. Those tools can be divided into a core group of tools that are fundamental for the operations of the ML pipelines and into an additional group of tools that act as complements to the core tools.

The following Figure 4 maps core MLOps technologies to the reference architecture described in Section 3 of the document. Furthermore, Section 4.5 presents a more detailed view of the tools introduced and their purpose in the MLOPs platform.

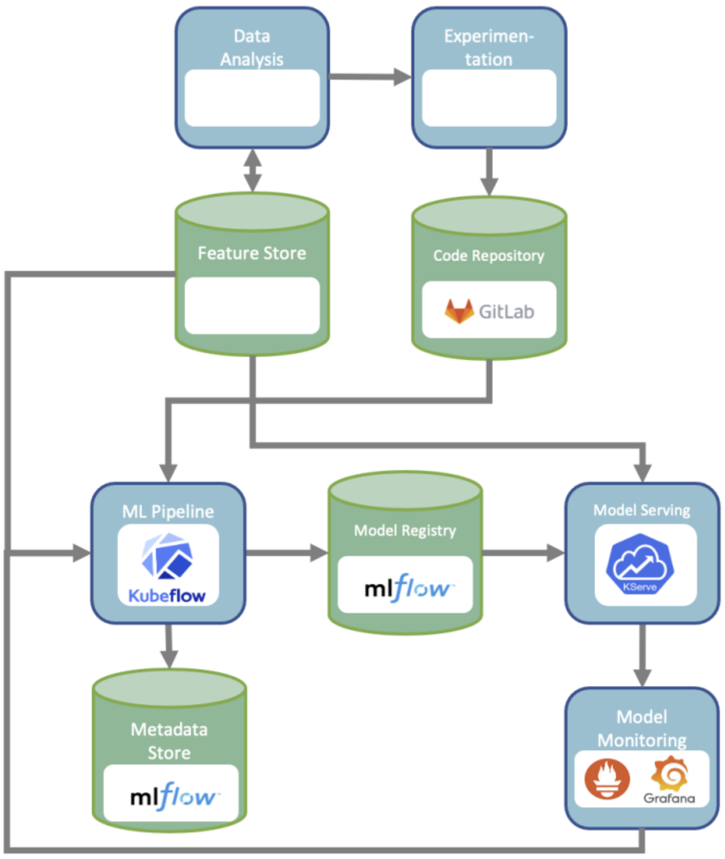


Figure 4 – MLOPs tools utilized for the establishment of the IML4E OSS platform.

In the above figure, the Feature store component remains empty since it was difficult to introduce one feature store technology that can generalize to all of the MLOps use cases while keeping the complexity of the solution low. IML4E partners are of the opinion that different data layer technologies can be utilized as feature stores. For example, for batch use cases that are running on top of tabular data Deltalake (Delta, 2023) technology can be used. For use cases that online inference is a requirement Feast tool (Feast, 2023) could be utilized or an inhouse implementation of feature store using Redis or other online databases could be developed. It is worth mentioning that in the area of feature stores, there are not a lot of open-source tools to be used and the tools available are in their starting phase with low utilization.

The Data analysis and experimentation components on the platform remain empty too. At the moment no particular technology has been introduced in the OSS platform for data analysis. Jupiter notebooks is a very popular technology used in the industry for this purpose. This tool may be introduced in the later releases of the MLOps experimentation platform if requested by the partners.

3.4 Architecture Diagram

A more detailed view of the tools introduced in the experimentation platform is presented and discussed in the following paragraphs. Besides the core tools introduced in the following Figure 4 it is presented and supporting technologies like NGINX and MINIO that are needed for the operations of the platform.

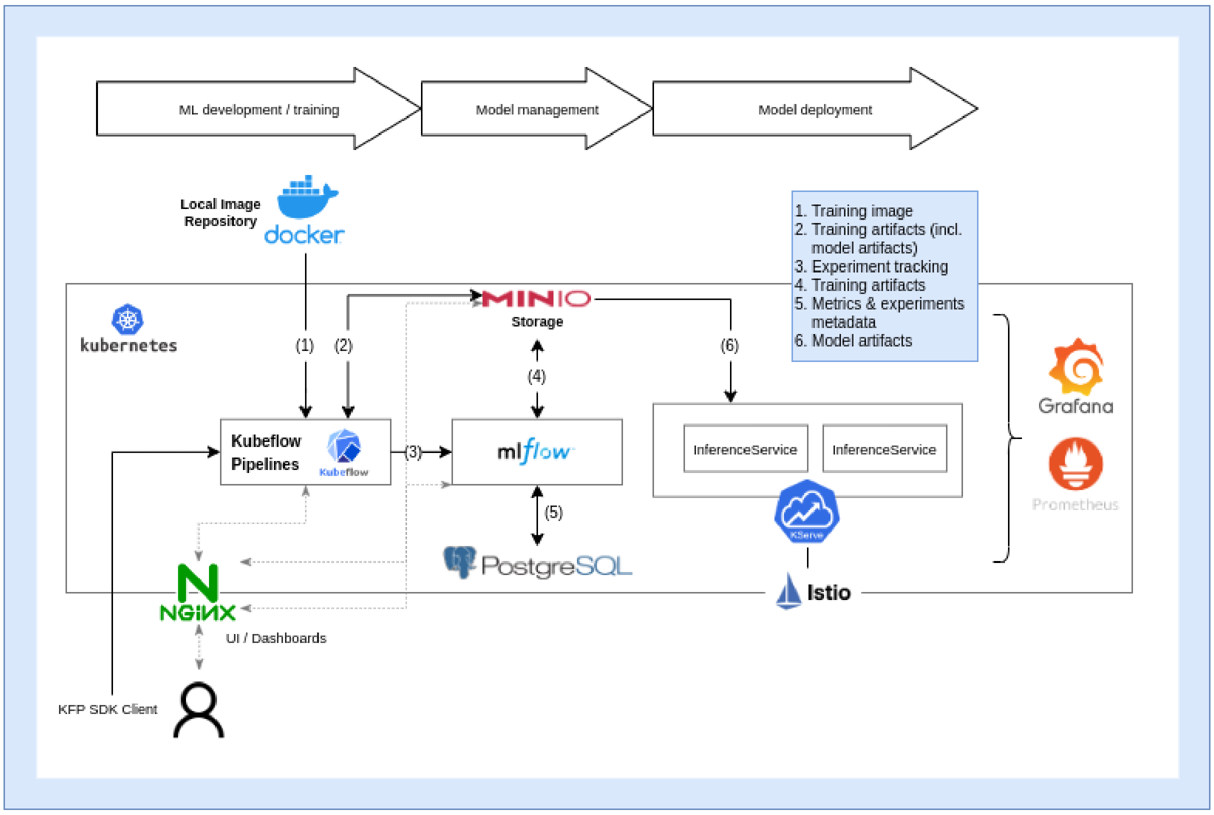


Figure 5 – A detailed view of the implemented IML4E experimentation platform

3.5 Core components

In our OSS platform the core components are

- Platform Architecture (compute layer): Kubernetes (OSS)
- Orchestrator: KubeFlow Pipelines
- Experiment Tracker: MLFlow
- Model registry: MLFLow
- Inference Engine: KServe

Kubernetes (K8s) is introduced for the automation of scaling, deployment, and management of containerized solutions. In addition to being de facto standard in this regard Kubernetes meets the requirements for having a scalable compute layer that can include CPU as well as GPU resources. Finally, Kubernetes can run on local machines as well as onin clusters of machines and it is supported by the majority of the cloud providers. (Google Cloud, Kubernetes, 2022)

Kubeflow is introduced for addressing the needs of “the machine learning developing / engineering layer”. Kubeflow pipelines enable the orchestration and automation of ML pipelines on top of Kubernetes. They help in managing dependencies between the different execution steps in an ML pipeline and they enable the execution of multiple ML pipelines in parallel. Kubeflow runs on top of Kubernetes and originated in Google. (Kubeflow pipelines, 2022)

MLFlow is introduced for the purposes of managing the needs of versioning and governance of the ML models. MLflow supports metadata tracking of the different artifacts generated during the execution of one ML pipeline. In addition, it supports comparison of different ML experiments, and it also supports a layer for governance regarding the state of the models [state can be "None’, ‘Archived’, ‘Staging’, or ’Production’]. MLFlow can run on top of Kubernetes and it was originated from Databricks company and now is part of the Linux Foundation (MLFlow 2022).

KServe is introduced for the deployment of ML models behind REST-APIs, and it also supports predictions for batch data. KServe runs on top of Kubernetes and it originated from the same project as Kubeflow pipelines. In addition, KServe integrates well with Grafana and Prometheus tools that we are using for our monitoring purposes (KServe 2022).

3.6 Additional Components

- Nginx (Ingress Controller)
- Docker registry
- Grafana & Prometheus
- Minio

Nginx is introduced with a purpose to control the ingress traffic to our OSS platform. With the help of Nginx we can also enforce user authentication on the services exposed outside our Kubernetes cluster.

Docker registry is introduced for hosting the container images that are going to be loaded to our Kubernetes cluster. Introducing a Docker registry in the solution enables better versioning of the software that runs in our platform.

Grafana & **Prometheus** are introduced for monitoring the execution of the deployed ML pipelines, for monitoring the "health" of the OSS platform and for monitoring ML models operations (logging inference inputs and outputs)

Minio is introduced for providing an object storage solution in the IML4E platform. Minio implements the S3 protocol for storing data and is used for storing model binary files versioned in MLFlow as long as datasets.

3.7 Accessing the OSS platform source code

The implemented OSS platform is accessible at the following GitLab repository: https://gitlab.fokus.fraunhofer.de/iml4e/iml4e_oss_exp_platform.[1]

3.8 Tutorials and educational material for the OSS platform

Tutorials for the OSS platform reference architecture can be found here:

https://gitlab.fokus.fraunhofer.de/iml4e/iml4e_oss_exp_platform/-/tree/main/tutorials[2]

In addition to the technical material, partners are building a case-based training that partly may utilise the OSS platform. The gist of the training will be to highlight the principals and methodology that the following chapter introduces. This type of training is ideal for intentionally mixed professional team that include a range of roles from conceptual design to data scientist and ML/data engineers.

1. Introduction to MLOps concepts and lifecycle
2. Workshop(s) where the team plans for realistic scenarios. Some of the scenarios includes implementation tasks whilst most of them are whiteboard / desk exercises. The questions that the team has to consider during analysing a case:
    a. Stakeholder analysis: who needs what, who has to do things?
    b. Should the solution be essentially technological or cultural?
    c. Is a general plan enough? Should you specify the particular tools?

[1] Currently the access to the code repository is restricted. In order to obtain access to the above link a request should be sent to the IML4E Management team. Please contact us at https://iml4e.org
[2] See above

   d. Usually solutions have up- and downsides for the various stakeholders. Analyse them. Assess if the solution would be realizable in the real world. If not, what would you deem the hardest obstacle?

3. An example of a study case for the workshops:

   "A four-person start up develops a machine vision quality assurance system for industry. There is one technical person working on the model and another one with the infra. The model maker gets into a serious accident and a new person - experienced in computer vision, but from the media field - takes over. What arrangements would make - or have made -

   - the new modeller to know the objectives,
   - the relevant data discoverable and understandable
   - There's a model deployed for production. How to keep it alive and make decisions on update and development need?"

4 Initial IML4E Principles and Methodology

MLOps is a combination of the disciplines of DevOps, Machine Learning and Data Science. It addresses: Interdisciplinary Teams, ML and Data Pipelines, Versioning of Models and Data, Model Validation, Data Validation, Monitoring and establishes a close link between the different disciplines and the development and operation of a model.

In this section, we describe principles and methods to help organizations implement and establish MLOps. We assume that the established methods from both software development and machine learning are not sufficient on their own. The reason for this is that figure there is a fundamental difference between ML and traditional software. ML is not just code, but code plus data. Moreover, ML is much more experimental than classical software engineering. In the following, we provide a short overview on existing processes and life cycle models in ML, we introduce the IML4E MLOps life cycle model, an initial IML4E maturity and improvement scheme and a method for continuous audit-based monitoring and certification.

4.1 Overview on existing processes and frameworks in ML

For ML to work for industrial applications, established tools, methods and processes are required. In the following section, we present a series of process models that are suitable for forming the basis for a reference process as a foundation for MLOps. (Amershi et al. 2019) summarizes the experiences of several Microsoft software development teams into a nine-step workflow for integrating machine learning into application and platform development (see Figure 6). The workflow includes the phases of model requests, data collection, data cleaning, data markup, feature engineering, model training, model evaluation, model deployment, and model observation.

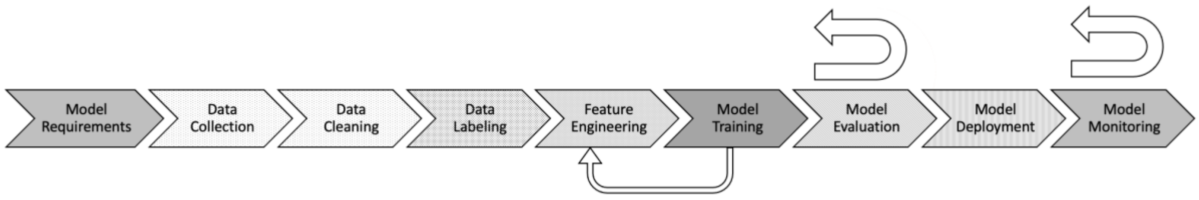


Figure 6 – Microsoft Workflow for ML

(Akkiraju 2017) describe a reinterpretation of the Software Capability Maturity Model (CMM) for the machine learning model development process. The paper presents a set of machine learning activities and best practices to help organizations reach a higher level of maturity in ML application development, regardless of their current starting point. The basis of the maturity model is a reference process model with 5 different sub-processes covering the distinct activities of data mining, data preparation, model building, testing and validation, and deployment. R. Akkiraju et al. distinguish in

1. data collection & preparation
2. feature preparation
3. model training
4. testing & benchmarking
5. deploy & learn

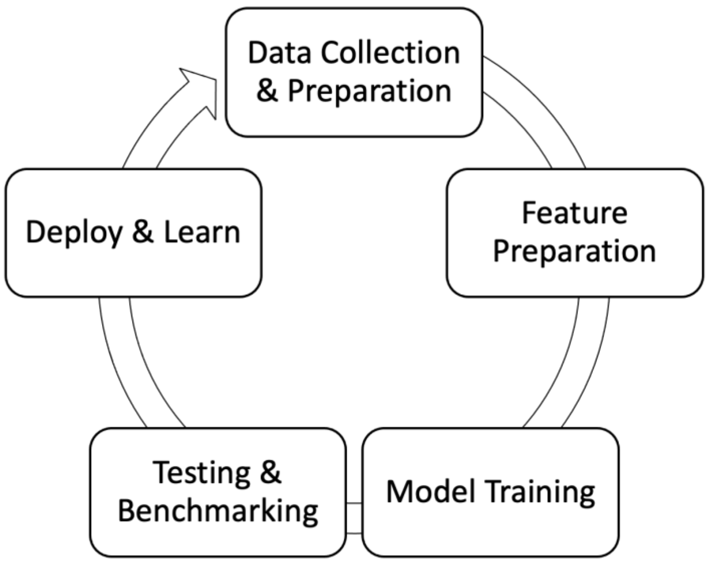


Figure 7 – The 5 different sub-processes covering the distinct phases of ML (Akkiraju 2017)

(Sothilingam et al. 2020) build on the work of R. Akkiraju et al. and provide a conceptual model to express the characteristics of the key processes in the ML workflow at an abstract level. The workflow starts with a definition of the business need and the question of what added value an ML application can bring to the enterprise. The development of the technical ML model starts with the preparation and cleansing of the data for the training process. Between training and validation there is an iterative cycle to achieve the desired quality for the ML model. Once the ML model is developed, it is deployed in a business application and tuned to the business case that was de-fined in the first step of the ML workflow. Once deployed, the ML model becomes accessible to the end user.

CRISP-DM (Cross Industry Standard Process for Data Mining) defines a process model for data mining and ML. This standard is used by ISACA as the basis for ML audits. CRISP-DM defines six phases that are iterated one or more times. These are:

1. Business understanding: in this phase the goals and requirements should be defined. The result is the formulation of the concrete task and the rough approach.
2. Data understanding: In this phase the available data are secured, and their quality is evaluated.
3. Data preparation: Construction of the final data set for modelling
4. Modelling: Methods suitable for the task are applied to the final data set.
5. Evaluation: Selection of the model that best fulfils the task. Careful comparison with the task.
6. Deployment: Integration of the model into the operational infrastructure of the customer/user.

Based on CRISP-ML, Studer et al. (Studer et al. 2020) propose CRISP-ML (Q), a process model for the development of machine learning applications extended by quality assurance activities. The process model defines tasks that span the entire life cycle of an ML application. For each task, a quality assurance methodology is presented that is based on practical experience as well as scientific literature and provides a solid foundation for holistic quality assurance. The process model distinguishes the phases Business and Data Understanding, Data Preparation, Modelling, Evaluation, Deployment, Monitoring and Maintenance. The authors propose to evaluate the models on the basis of six complementary properties. In addition to model performance (i.e., accuracy and generalization properties), non-functional measures such as robustness, explainability, scalability, hardware requirements, and model complexity must also be evaluated.

(Wood et al. 2019) describe aspects of a workflow for the development of an application with a deep neural network (DNN). They orient themselves on the phases of a classical software life cycle with the phases Define, Specify, Develop & Evaluate and Deploy & Monitor.

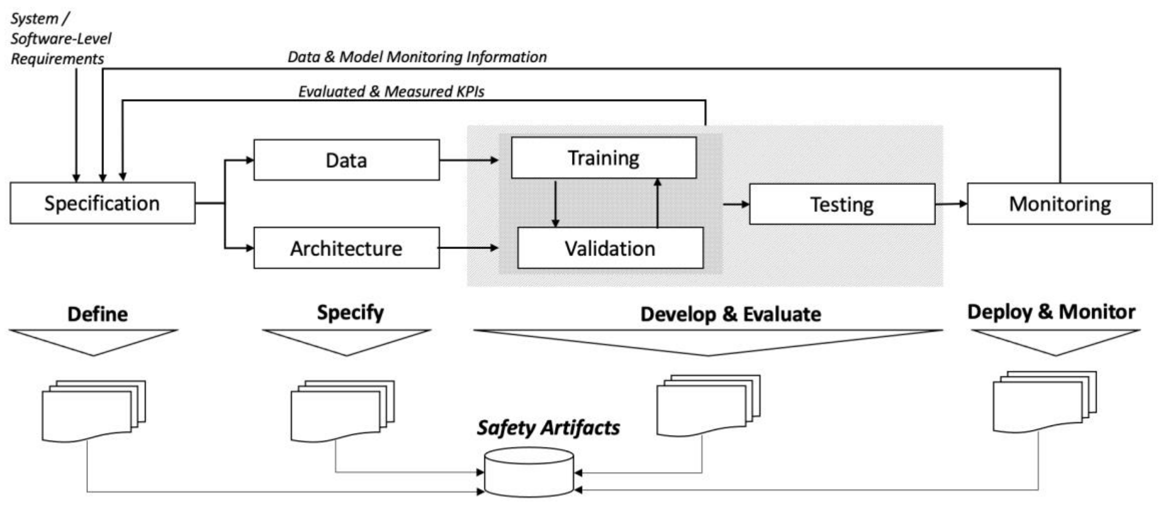


Figure 8 – Process model according to Wood et. al. (Wood et al. 2019)

In order to be able to compare the different process models for ML with each other, we have drawn up a comparison table (see Table 3) based on Studer et al. (Studer et al., 2020) that relates the individual phases of relevant process models to each other. We included CRISP-DM 2003 (Chapman et. al. 2018), CRISP-ML(Q) (Studer et al. 2020), Amershi (Amerhsi et. al. 2019), Safety First (Wood et al. 2019), Akkiraju (Akkiraju et al. 2017) in the comparison.

CRISP-DM	CRISP-ML(Q)	Amershi 2019	Akkiraju 2017	Safety First
Business Understanding	Business and Data Understanding	Model Requirements	Model Goal Setting and Offering Management:	Define
Data Understanding			Content Management Strategy	
		Data Collection	Data Collection	
Data Preparation	Data Preparation	Data Cleaning	Data Augmentation	Specify
		Data Labeling		
		Feature Engineering	Feature Engineering	Develop & Evaluate
Modeling	Modeling	Model Training	Modeling	
Evaluation	Evaluation	Model Evaluation	Testing & Benchmarking	
Deployment	Deployment	Model Deployment	Model Deployment	Deploy & Monitor
	Monitoring & Maintencance	Model Monitoring	AI Operations Management	

Table 3 – Comparison of process models from literature (extension of Studer et al. (Studer et al., 2020))

In all processes, there is a clear separation between the phases of *Problem Understanding*, *Data Preparation*, *Modeling*, *Deployment* & *Monitoring*. CRISP-DM lacks a monitoring and maintenance phase. This is primarily because CRISP-DM is strongly related to data mining and not to the operation of software. Furthermore, there are some differences in the extent to which explicit modeling of the business requirements is provided. Here, CRISP-DM and Akkiraju are very explicit. In addition, Akkiraju and Amerhsi further break down the process of data preparation and identify an explicit phase for feature engineering. Compared to the others, Wood et. al. use terminology that is strongly orientated on the development of safety-critical software. Here, for example, a clear delineation of the phase for modeling, which is integrated in their phase Develop & Evaluate, is also missing. However, in principle and except for Wood et. al. we were unable to identify any major differences in the rough structure of all the reference models considered.

4.2 The IML4E MLOps life cycle

MLOps is a DevOps that focuses on machine learning. Just like DevOps, MLOps is not just focused on operations and is a culture and set of practices for the entire team. Unlike traditional software systems, where the behaviour

of the system comes from the code, the behaviour of machine learning systems comes from the model, which is trained with data. Compared to DevOps, this brings additional challenges that have not been well researched or fully resolved.

While many DevOps practices from traditional software systems are applicable to ML systems, there are key differences. Amershi et al. (Amershi 2019) for example identify three fundamental differences in the application of software engineering to ML components compared to other application domains. These are:

- **Data:** The mining, management, and versioning of data needed for applications of ML is much more complex and difficult than other types of software engineering
- **Competency:** The adaptation and reuse of models requires very different skills than are typically found in software teams. These include mathematical, scientific skills and a solid know-how in statistics.
- **Complexity:** AI components are more difficult to manage as standalone modules than traditional software components. Models can be "entangled" in complex ways and exhibit non-monotonic error behavior, making their integration a major challenge.

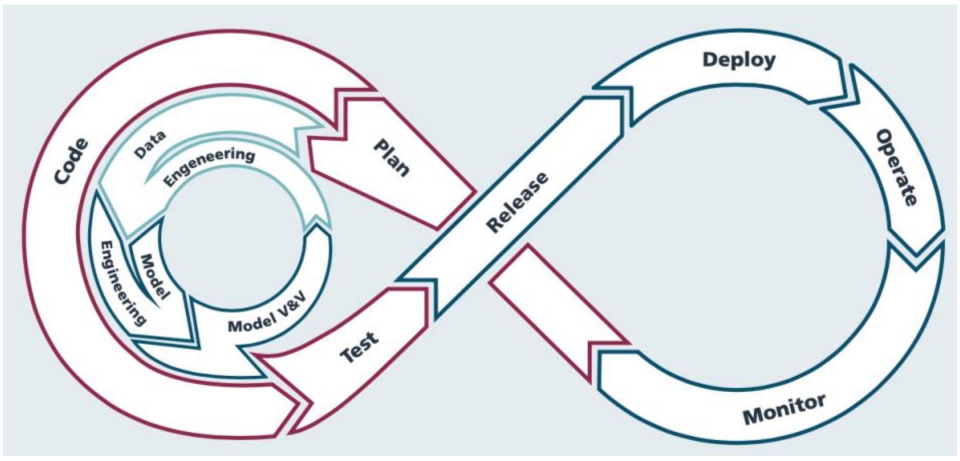


Figure 9 – IML4E MLOps life cycle

An MLOps cycle consists of the following phases as depicted in Figure 9.

- **Plan:** This phase serves to develop a basic understanding of the problem to be solved by ML as well as the data required for it. Developers decide which features are feasible with machine learning and which can be useful for a given existing or a new product. Importantly, it is at this stage that they decide what types of models are best suited for the given problem and what data is needed to successfully learn the desired features.

- **Code:** This phase aims to develop the source code that is required for setting up the model and related components. This includes, among other things, all components that form the data preparation and decision pipeline during deployment. The individual components, the software implementation of the model and the integration of all components into a functional unit must be realized and tested.
- **Data Engineering:** This phase aims to identify the right data sets in the right distributions, collect the data in such a way that the model output can be delivered as efficiently as possible, enrich the data through labelling, store the lineage of the data, verify the quality of the labelled and prepared data, establish specific metrics to measure the quality of the data, store and analyse the data.
- **Model Engineering:** This phase aims to parameterize and train model variants based on the available data and their labels by considering all related requirements, and to identify and select the most appropriate model. In practice, training runs 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. The best model(s) is/are selected and passed to the subsequent process for further V&V. For modelling, predefined ML frameworks are usually used to support the developers in creating the model code and realizing the algorithms.
- **Model V&V:** In this phase, the model is fully validated and verified. The aim is to assess the extent to which the model fully meets the functional and extra functional requirements given. In addition to accuracy and generalizability, scalability, complexity, robustness, fairness, and resource requirements in particular play a crucial role in the evaluation of a model. The model is checked against all relevant performance characteristics and KPIs. Finally, it is decided at this point whether the model(s) at hand are suitable for integration into the software stack or need to be improved via further iterations.
- **Test:** The selected model must be tested and evaluated in its deployment environment. For this purpose, the model must be integrated with the software and hardware of the target platform and systematically tested. The evaluation focuses on the safe functioning of the model in interaction with the necessary software and hardware as well as the interaction with additional safety components and functions, such as watchdogs for identifying dangerous situations and performing plausibility checks, redundancy for safeguarding the overall functionality, etc.
- **Release:** In this phase, the model is transfered in a real-world setting or integrated into the application/edge device it was developed for. Most organizations typically follow stringent procedures, or even provide standard templates for building a complete release package. Testing happens throughout this process, starting with testing all input CIs, to testing and rehearsing the services before they are deployed live.
- **Deploy:** In this phase, the release package is deployed to the live environment, beginning when change management authorizes the release package to be deployed to the target environments. Afterwards operators deploy the application to a production environment. They may also perform maintenance tasks on the application. There are different levels of deployment for models. In shadow deployment, the model results are not used directly. Deployment occurs in parallel with another model or the final decision is made by a human, regardless of what the model predicts. Usually this is done to determine how well the model is performing. In canary deployment, the model is presented with only a small portion of the data on which it is allowed to make decisions. Depending on how the model performs, the data is gradually increased, or the model is withdrawn, and adjustments are made. At Blue-Green deployment, the traffic is gradually changed from an old version (blue version) to the new version (green version). In this way, any kind of downtime is avoided and in case of errors, the application can be easily reverted to the previous stable version or the blue version.
- **Operate:** This phase is organizing the operation of the ML-based application. It aims to minimize or eliminate downtime for your end users through efficiently managing hardware and software changes.
- **Monitor:** Monitoring of the ML-based application and its application environment is the final phase in the MLOps cycle. It builds on the customer feedback by collecting data and providing analysis on customer behaviour, application performance, and defects. Regarding the ML-models, monitoring includes the predictive quality, the inference performance, the occurrence of special kinds of failure,

and certain kinds of drift. Especially in highly automated MLOps processes, this includes not only data about the application and the model, but also data about the entire infrastructure, i.e. the compute environment, the data and modelling pipelines, trying to identify bottlenecks in the process of continuous training and the provision of new models and application updates. The monitoring data is used to take automated decisions like e.g., retraining and is fed back to the product manager and the development team in a structured and preferably automated way.

4.3 The IML4E Maturity Assessment and Improvement Scheme

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, based on this, to make informed decisions about which improvement actions may be a useful addition in light of the current maturity and expansion goals. In the following, we present some existing schemes that can be used to determine the maturity of ML or MLOps processes. We then describe a first version of the IML4E Maturity Assessment schema.

4.3.1 Existing approaches to ML and MLOps maturity

While in the area of Big Data there exists a number of maturity schemes for Big Data related processes, there are only a limited number of maturity schemes for ML and MLOps. The work of R. Akkiraju et al (Akkiraju et al. 2017) addresses ML and considers several existing maturity management models for Big Data. These include the TDWI Big Data Maturity Model (Nott 2015), the IBM Big Data and Analytics Maturity Model (Schmarzo 2016), EMC's Big Data Business Model Maturity Index (Dhanuka 2017), the Hortonworks Big Data Maturity Model (Hortonworks 2016), Booz & Company's BDMM (El-Darwiche 2014), and Radcliffe's BDMM (Radcliffe 2015).

John et. al. (2021) have developed an MLOps framework that details the activities involved in the continuous development of machine learning models. The framework distinguishes 4 levels of maturity namely “Automated Data Collection”, “Automated Model Deployment”, “Semi-automated Model Monitoring”, and “Fully-automated Model Monitoring”. For each level, the authors have defined a set of preconditions that are derived from the state of the art. They validated the framework in three embedded systems case companies.

Google has introduced a three-level maturity scheme in their basic white paper on MLOps (Google 2022). Google distinguishes “Manual process”, “ML-pipeline automation”, and “CI/CD pipeline automation”. While a manual process is characterized by script driven and interactive processes, a missing connection between ML and operations as well as no CI/CD and a lack of active performance monitoring, ML-pipeline automation is characterized by continuous training (CT), continuous delivery (CD) of models and pipeline deployment, i.e., the deployment of an infrastructure that continuously serves the active models with training updates. CI/CD pipeline automation in addition provides a rapid and reliable update of the pipelines in production by means of an automated CI/CD system.

In addition, Microsoft has introduced a 5-level maturity scheme for MLOps (Microsoft 2022), which realizes a more detailed analysis and improvement path. In addition to the 5 maturity levels, which range from “No MLOps” to “Full MLOps Automated Operations”, a distinction is made between "People", "Model Creation", "Model Release" and "Application Integration" as relevant topics or perspectives of the assessment.

In a literature review Sadiq et. al. (2021) identified 15 studies that deals with the topic of ML or AI maturity. In addition to the approaches listed above, the authors examined 14 approaches that are not listed above. Most of them have been developed in the last 3 years. While 10 of them are created with a certain domain in mind, only 5 are applicable in a general manner. The study reveals that the maturity models have different designs regarding the number of levels, descriptors, and elements.

For example, the study shows that there is differentiation in the number of maturity levels. The most frequently chosen differentiation consists of five levels. This appears to the authors to be suitable because it prevents too much dispersion of details and thus makes the evaluation process less complicated.

The most critical dimensions identified from these 13 studies are Data, Analytics, Technology and Tools, Intelligent Automation, Governance, People, and Organization. None of the studies under review took care to measure all critical dimensions.

4.3.2 Levels and Dimensions

The IML4E Maturity Assessment is mainly envisioned to support the integration of technologies, methods, and processes in the IML4E framework and to measure the progress that is made in the IML4E case studies as described in the Deliverables D1.1 (IML4E 2022a) and D1.2 (IML4E 2022b).

It is built upon the assessment scheme introduced by Microsoft (Microsoft 2022). We think this makes sense, because the Microsoft scheme with its 5 levels has introduced a relatively fine-grained evaluation and improvement scheme, in which the transitions from one level to the next can be made continuously and without big steps. These are depicted in Figure 10 and are listed below.

- Level 4: Full MLOps Automated Operations
- Level 3: Automated Model Deployment
- Level 2: Automated Training
- Level 1: DevOps but no MLOps
- Level 0: No MLOps

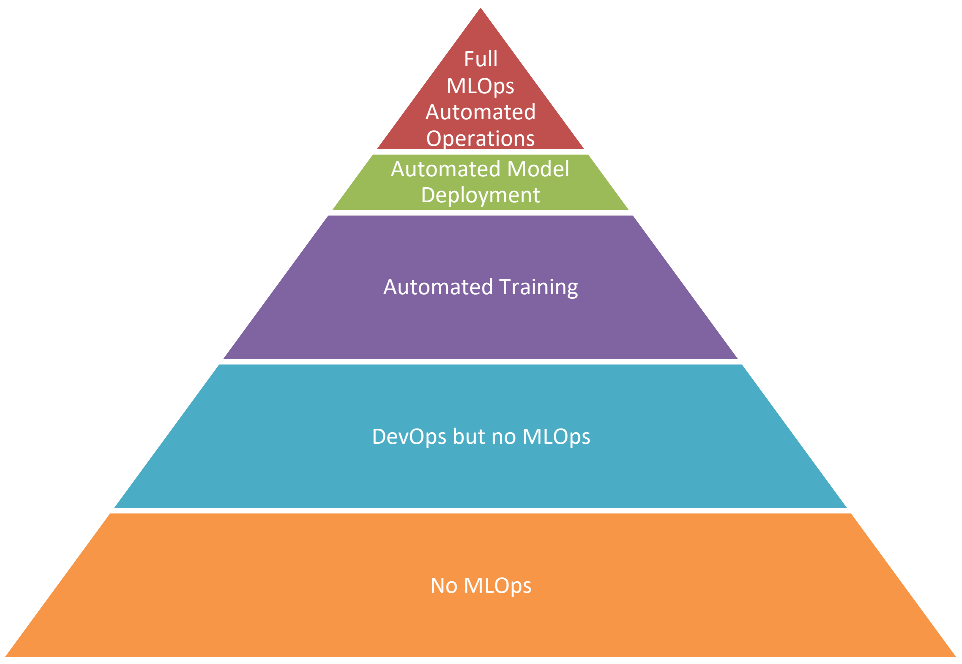


Figure 10 – Maturity levels of the IML4E Maturity Scheme (adopted from (Microsoft 2022))

In addition to the 4 dimensions defined by Microsoft, we use a more fine-grained structure that is meant to cover aspects that we consider relevant in the IML4E project and for a mature MLOps processes. These are depicted in Figure 11 and listed below.

- People
- Data Preparation
- Model Creation
- Model Release & Application Integration
- Visualization, Monitoring & Documentation
- Testing and Validation

- Tool Support and Tool Integration

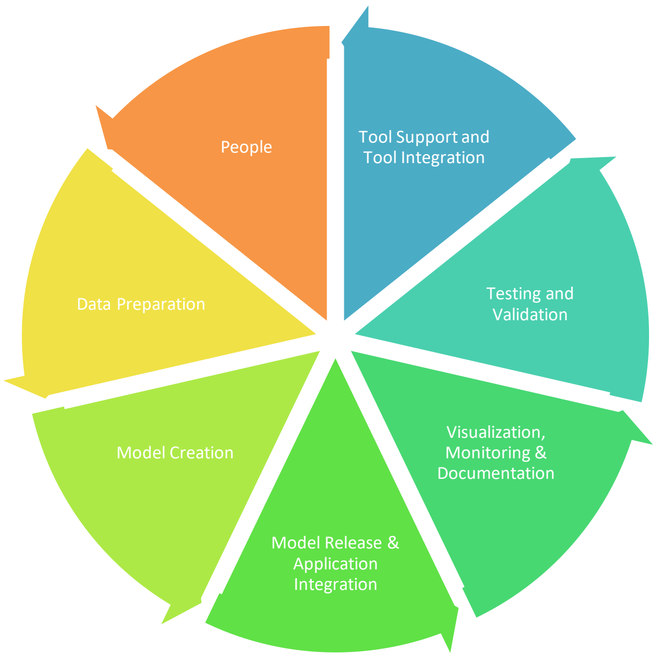

People
Tool Support and Tool Integration
Testing and Validation
Visualization, Monitoring & Documentation
Model Release & Application Integration
Model Creation
Data Preparation

Figure 11 – Assessment dimensions in the IML4E Maturity Scheme

The individual level definitions are defined in the following sections. They are partially taken from (Microsoft 2022) and partially defined in the IML4E project. Level definition that are taken from (Microsoft 2022) are denoted in grey and have origin "MS". The level definition that are defined in the IML4E project are denoted in black and have origin "IML4E".

4.3.2.1 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 scientist), dev ops teams (e.g., software engineers, operations) and the business teams (e.g., business architects). The following table describes the various assessment statements this aspect.

People					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	Origin[3,4]
	Data scientists: siloed, not in regular communications with the larger team		Data scientists: Working directly with data engineers to convert experimentation code into	Data scientists: Working directly with data engineers to convert experimentation code into	MS

[3] All level definitions that are marked with origin "MS" are adopted from (Microsoft 2022). Copyright is subject to the rules and regulation that yield for (Microsoft 2022).
[4] All level definitions that are marked with origin "IML4E" are introduced by the IML4E project and are licensed according to CC BY 4.0

			repeatable scripts/jobs	repeatable scripts/jobs. Working with software engineers to identify markers for data engineers	
	Data engineers (if exists): siloed, not in regular communication with the larger team	Data engineers: Working with data scientists		Data engineers: Working with data scientists and software engineers to manage inputs/outputs	MS
		Software engineers: siloed, receive model remotely from the other team members	Software engineers: Working with data engineers to automate model integration into application code.	Software engineers: Working with data engineers to automate model integration into application code. Implementing post-deployment metrics gathering	MS
Dev teams and data teams are not in regular communication with operations and business team		Dev teams and data teams are in direct communication with the operations team to commonly address issues during operations.	DevOps teams and data teams are in direct communication and resolve issues observed during operation,	MLOps teams and business teams are in direct communication to directly react on business related fails.	IML4E

Table 4 – Level definitions for the aspect "People"

4.3.2.2 Data Preparation

During data 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 this aspect.

Data Preparation					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
Data gathered manually				Data pipeline gathers data automatically	MS

Data are prepared for individual models/solutions		Data are prepared/reused for multiple models/solutions		Data are systematically managed for reuse (e.g., by dedicated feature stores)	IML4E
		Data, models and other artifacts are version controlled in a consistent manner	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.	IML4E

Table 5 – Level definitions for the aspect "Data Preparation"

4.3.2.3 Model Creation

Model Creation aims at providing the best modelling solution for a given tasks. 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 are usually provided support in creating the model code and realizing the algorithms. The following table describes the various assessment statements this aspect.

Model Creation					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
Compute is likely not managed	Compute is or isn't managed			Compute managed	MS
	Experiments aren't predictably tracked			Experiment results tracked	MS
	End 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 triggered automatically based on production metrics	MS
	Manual hyperparameter and architecture determination	Automated hyperparameter and architecture optimization for a given training pipeline	Automated hyperparameter adjustment for each model iteration.	Full automated hyperparameter and architecture determination for each model iteration.	IML4E

Table 6 – Level definitions for the aspect "Model Creation"

4.3.2.4 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 this aspect.

Model Release & Application Integration					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
	Manual process	Manual release		Automatic Release	MS
	Scoring script may be manually created well after experiments, likely version controlled			Scoring Script is version controlled with tests	MS
Release handled by data scientist or data engineer alone	Is handed off to software engineers	Release managed by Software engineering team	Release managed by continuous delivery (CI/CD) pipeline	Release managed by continuous integration and CI/CD pipeline	MS
		Basic integration tests exist for the model		Unit and Integration tests for each model release	IML4E
Release is heavily reliant on data scientist expertise to implement		Release is heavily reliant on data scientist expertise to implement model		Release is less reliant on data scientist expertise to implement model	MS
Manual releases each time				Releases automated	IML4E
		Application code has unit tests		Application code has unit/integration tests	IML4E

Table 7 – Level definitions for the aspect "Model Release & Application Integration"

4.3.2.5 Visualization, Monitoring & Documentation

Monitoring, visualization, and documentation ensure that the performance of a model can be continuously verified and proven. Any model that is transferred to operational use starts to lose performance. Such performance loss must be detected, documented, and communicated. In addition, there must be a way to continuously report to stakeholders in the organization on whether and how the machine learning solution provided actually 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 this aspect.

Visualization, Monitoring & Documentation					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
No visualization	Individual data visualization techniques to support data exploration, verification, and validation.		Integrated data visualization dashboards to systematically guide data exploration, verification, and validation.	Comprehensive visualization dashboards to monitor and control technical and business related KPIs by business experts.	IML4E
No documentation	Manual documentation creation of some aspects of the model	Manual documentation creation of data and model	Semi-automatic documentation	Fully automated documentation generation	IML4E
Individual monitoring of some technical KPIs		Integrated 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.	IML4E

Table 8 – Level definitions for the aspect "Visualization, Monitoring & Documentation"

4.3.2.6 Testing and Validation

Testing and Validation ensures that the model meets its specified KPI. Testing and validation activities span over the whole life cycle of ML-based software. It starts with validating the requirements, the data and the model itself. But it also includes debugging, testing and validation activities at different levels of integration up to the deployed 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. The following table describes the various assessment statements this aspect.

Testing and Validation					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
Compliance policies and related quality attributes are checked manually		Relevant compliance policies and related quality attributes are checked based on automated tools integrated in the CI/CD pipeline		Continuous Audit of compliance policies and related quality attributes over the whole MLOps life cycle.	IML4E

Sporadic and manual 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.	IML4E
	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 examined further in a predefined process	Continuous analysis of data sets, i.e., examining gaps in data, missing values, existing trends, and so forth.	IML4E
	Data unit test are done sporadically.		Data unit test are done based on tools and automatically triggered for new data sets.	Data unit testing is fully automated and synchronized with other MLOps activities like data preparation, model training and operations.	IML4E

Table 9 – Level definitions for the aspect "Testing and Validation"

4.3.2.7 Tool Support and Tool Integration

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 this aspect.

Tool Support and Tool Integration					
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Origin*[3,4]
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	IML4E
Third party tools are not integrated				Full integration support for third party tools.	IML4E

Table 10 – Level definitions for the aspect "Tool Support and Tool Integration"

4.3.3 Planned Method of Application in IML4E

One of the purposes of the IML4E assessment scheme is to allow us to measure the progress of the project within the case studies. For this purpose, we have prepared an Excel template that can be used to assess the maturity of a case study at key project milestones. These would be project milestones MS2, MS4, and MS6. While the project milestone MS2 describes the point in time when the case studies are defined but not yet supported by contributions from the IML4E project, the milestones MS4 and MS6 represent reference points for the integration and evaluation of the IML4E techniques along the case studies. A representation of the milestones can be found in Figure 12.

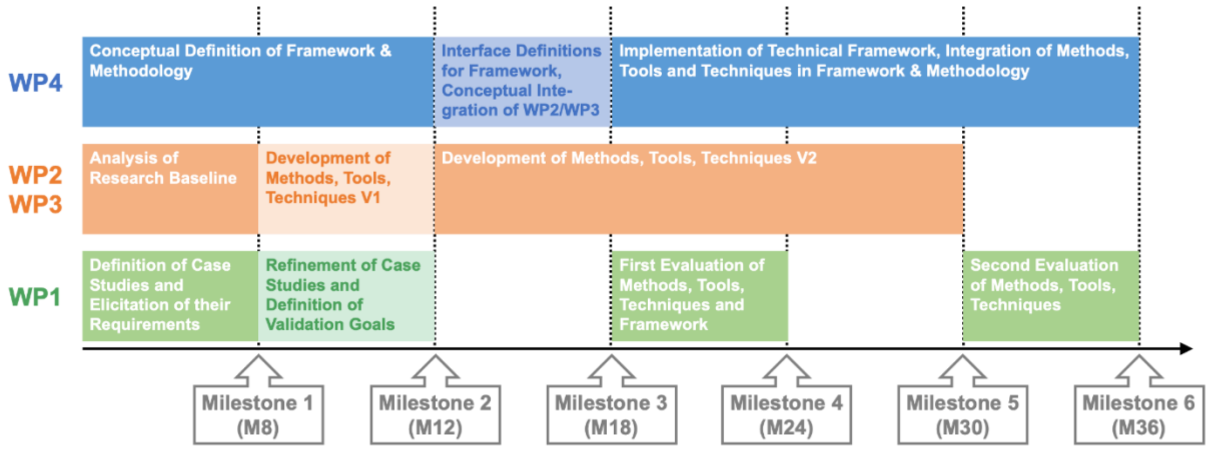
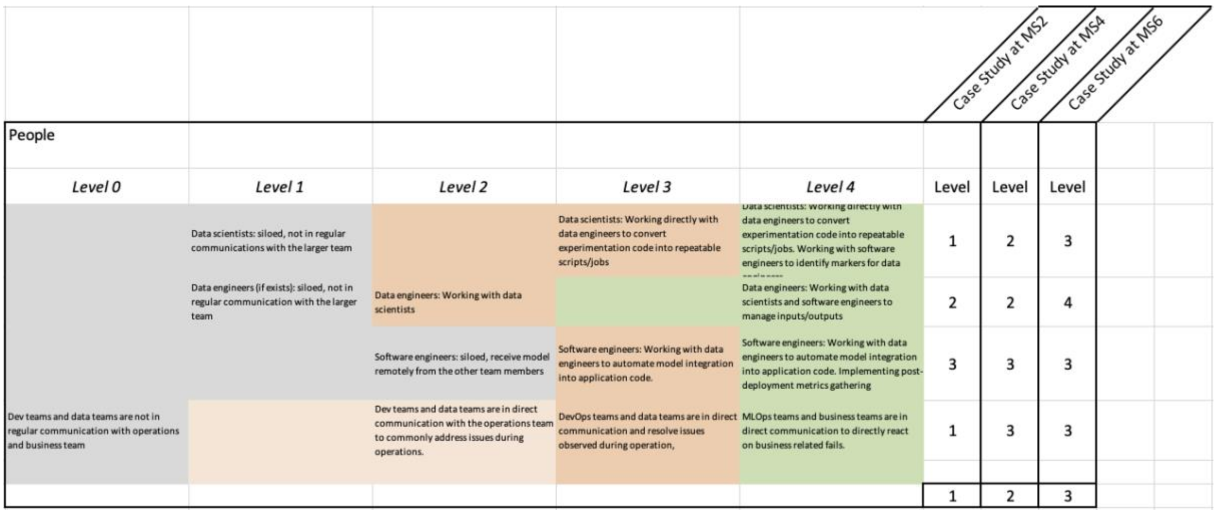

WP4
Conceptual Definition of Framework & Methodology
Interface Definitions for Framework, Conceptual Inte-gration of WP2/WP3
Implementation of Technical Framework, Integration of Methods, Tools and Techniques in Framework & Methodology
WP2
WP3
Analysis of Research Baseline
Development of Methods, Tools, Techniques V1
Development of Methods, Tools, Techniques V2
WP1
Definition of Case Studies and Elicitation of their Requirements
Refinement of Case Studies and Definition of Validation Goals
First Evaluation of Methods, Tools, Techniques and Framework
Second Evaluation of Methods, Tools, Techniques
Milestone 1 (M8)
Milestone 2 (M12)
Milestone 3 (M18)
Milestone 4 (M24)
Milestone 5 (M30)
Milestone 6 (M36)

Figure 12 – Milestones in the IML4E project

With the help of the Excel template, a simple evaluation of each case study can be made. In the template, there is a table for each perspective, as shown in Figure 13. The table defines a set of factors that we consider relevant for each perspective. Each of these factors is assigned a value according to the descriptions defined in the table.

The resulting overall score for a perspective is defined by the minimum of all values.



People					Case Study at MS2	Case Study at MS4	Case Study at MS6
Level 0	*Level 1*	*Level 2*	*Level 3*	*Level 4*	Level	Level	Level
	Data scientists: siloed, not in regular communications with the larger team		Data scientists: Working directly with data engineers to convert experimentation code into repeatable scripts/jobs	Data scientists: Working directly with data engineers to convert experimentation code into repeatable scripts/jobs. Working with software engineers to identify markers for data	1	2	3
	Data engineers (if exists): siloed, not in regular communication with the larger team	Data engineers: Working with data scientists		Data engineers: Working with data scientists and software engineers to manage inputs/outputs	2	2	4
		Software engineers: siloed, receive model remotely from the other team members	Software engineers: Working with data engineers to automate model integration into application code.	Software engineers: Working with data engineers to automate model integration into application code. Implementing post-deployment metrics gathering	3	3	3
Dev teams and data teams are not in regular communication with operations and business team		Dev teams and data teams are in direct communication with the operations team to commonly address issues during operations.	DevOps teams and data teams are in direct communication and resolve issues observed during operation,	MLOps teams and business teams are in direct communication to directly react on business related fails.	1	3	3
					1	2	3

Figure 13 – Excel template for assessing individual IML4E case studies

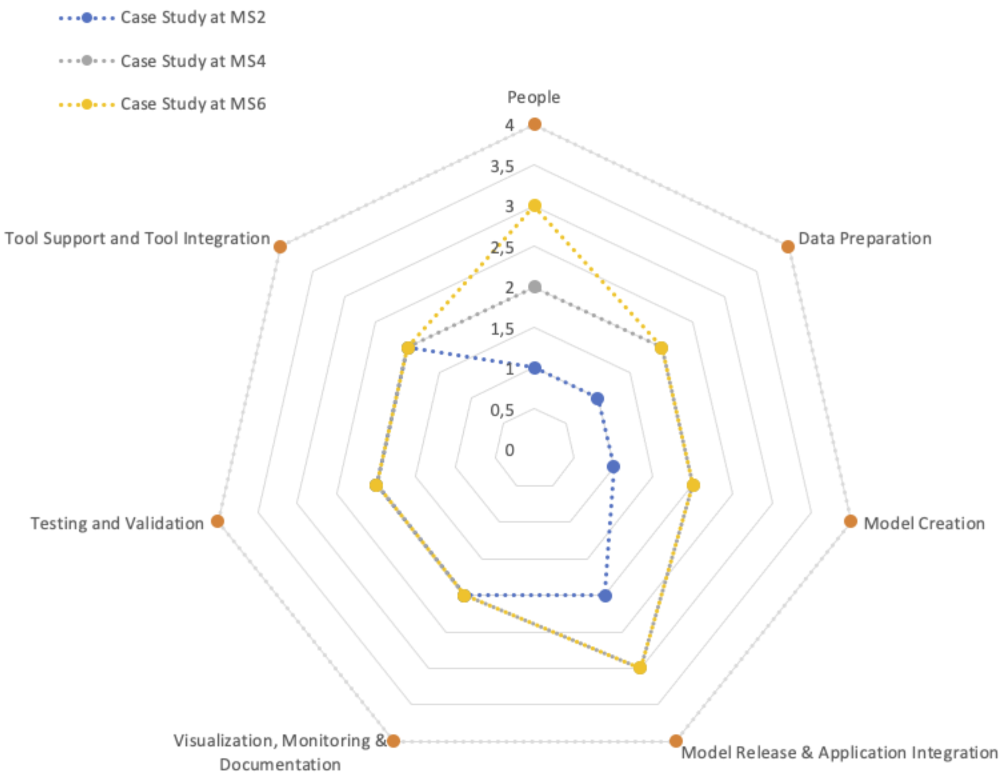


Figure 14 – Results overview for an individual case study

Figure 14 shows a potential result of the evaluation of a case study after 3 evaluation rounds. For each perspective, the increase in maturity can be easily and clearly read.

4.4 The Continuous Audit-based Quality Assurance Methodology

4.4.1 Idea

In the area of quality management, audits have proven to be an effective measure for objectively evaluating the maturity or quality of a software artifact during the process. This procedure can also be usefully transferred to the development of ML models.

The audit is a check to determine whether guidelines, requirements or processes in a company meet the required quality specifications and standards. This can be either self-created or nationally or internationally specified specifications and standards. According to DIN EN ISO 9000, an audit is a "systematic and objective investigation to determine the degree of fulfillment of agreed requirements".

In quality management, there are various forms of audit, each of which can be used to identify potential for improvement and assess the effectiveness of measures. One criterion by which audits can be differentiated is the party performing the audit:

- **Internal audit (first party).** Only the company itself is involved in an internal audit. It is usually carried out by a trained employee outside the team entrusted with development. Internal audits are primarily used for self-assessment and promote communication about how to make processes in the company even more efficient.
- **Supplier audit (second party).** Two parties are involved in a supplier audit: a supplier and a customer or alternatively a party commissioned by a customer. In some industries, such as the automotive industry, it is common for customers to audit their suppliers prior to collaboration in order to check their suitability for collaboration.
- **External audit (third party).** External audits are carried out by independent certification bodies, so that a third, external party is involved. Based on samples, the auditor assesses the extent to which standards such as ISO 9001 for quality management are met. The aim is to obtain official certification and thus prove to customers, for example, that systems have been developed in accordance with international standards.

An audit requires an appropriately qualified auditor. Depending on the form of the audit, this can be an employee of a company or an external auditor. During the audit, the auditor examines various questions according to a predefined guideline, for which he is provided with the necessary data and information by the auditee. The audit concludes with documented results that can be built upon further. For example, the results of an audit can form the basis for internal company changes or product approval by an external certification body.

In a classic audit procedure, the auditor asks questions to various people from the group of auditees who can provide information on the respective topic. This is a very time-consuming manual process in which usable results are only available after a considerable amount of time, depending on the complexity of the question. Even with long release cycles, writing the necessary reports requires considerable manual effort and ties up resources. In a more agile process like MLOps - where changes can be implemented on a weekly or daily basis, possibly even independently of user intervention - effective auditing must be based on different principles.

The MLOps process optimization approach aims at an efficient, reliable and high-quality design of the development, deployment, management and monitoring of machine learning models in short development cycles. Methods such as Continuous Integration, Continuous Delivery and Continuous Deployment are used. MLOps relies on automated processes to ensure continuous testing, monitoring and validation of machine learning models.

With the **Continuous Audit-based Quality Assurance** approach described below, the MLOps portfolio is extended by a process support that supports the continuous audit-based assessment of MLOps artifacts (ML models) regarding development status and quality. The manual and time-consuming effort for audits is thus significantly minimized. Like classical software components, the development process and development status of ML models are primarily evaluated based on automatically determined key figures/metrics. This makes the collection of information required for the audit independent of manual collection, which can be carried out additionally if necessary. Instead, the metrics are automatically collected in the MLOPs process steps and made available for independent evaluation via a dedicated interface (Audit API) to any interested auditor. The (anonymized) tapping of the key figures via the Audit API supports, on the one hand, the independent evaluation of these key figures by the auditor. On the other hand, different audit scenarios can be supported, enabling a hierarchically staggered approach, starting with the simple compilation of metrics for reports up to a dedicated certification process for ML models.

4.4.2 Measurement

In software development, software metrics are classically used to evaluate the developed software artifacts. A software metric, or metric for short, is a (usually mathematical) function that maps a property of software into a numerical value, also called a measure. Hereby formal comparison and evaluation possibilities are created.

In the form of a numerical value, the metric serves as a measure of a property, a quality characteristic, of software. Simple metrics often represent only a simple functional relationship, more complex metrics, on the other hand, also attempt to assess more abstract properties, such as the maintainability, extensibility, and comprehensibility of the source code. With a suitable number of different metrics, it can be judged, how complex (i.e. personnel and cost-intensive) the maintenance, advancement and subsequent tests of the software become. Metrics cannot evaluate a correct implementation of the functions, but they can help to determine, which expenditure will prepare the production of the software approximately and how many errors will occur. If metrics are used regularly during the long-term further development of a software, negative trends, i.e. deviations from the quality target, can be detected and corrected at an early stage.

Metrics can also be used in an analogous way to assess the development of ML models, although different metrics are used here than for classical software artifacts. Accuracy is probably the most widely known and most easily understood metric for ML models. It compares the number of correct classifications to all classifications to be made. However, factors surrounding the actual ML model, such as the quality of the underlying training data, also play a decisive role in the evaluation of an ML model.

However, the number and possible criteria for evaluating an ML model are currently still quite confusing, as suitable metrics have yet to emerge. In part, existing standards can be used to evaluate the data quality of test data, for example, but in part, adequate metrics must first be developed depending on the respective use case, e.g., when "soft" quality criteria such as the robustness of an ML model are to be evaluated. For example, the evaluation of an ML model depends on whether it is unsupervised or supervised learning, regression or classification in the case of supervised learning, what the underlying use case is, and so on.

The metrics required for the evaluation of ML models can be collected automatically to a large extent (similar to classic software metrics). For this purpose, the existing development tools usually already provide the necessary raw data. These can be found in the form of entries in logs, meta logs and protocols and can be read out relatively easily and accurately with suitable analysis programs. In addition, automatically executable active tests can be used to collect further key figures that are useful for assessing the maturity of ML models. Corresponding tools are developed within the IML4E project and made available for the platform (CABC mapper).

The recording of the key figures is carried out exclusively by the auditees, which includes the entirety of those entrusted with the development of the ML models (developers, testers, project managers, etc.). It is their responsibility to integrate the tools available for this purpose into the development process or, if necessary, to design and use additional tools for the determination of key figures. The key figures recorded by the auditee are continuously forwarded in their chronological order via a dedicated interface (Audit API) and thus made available to interested auditors for various evaluation scenarios.

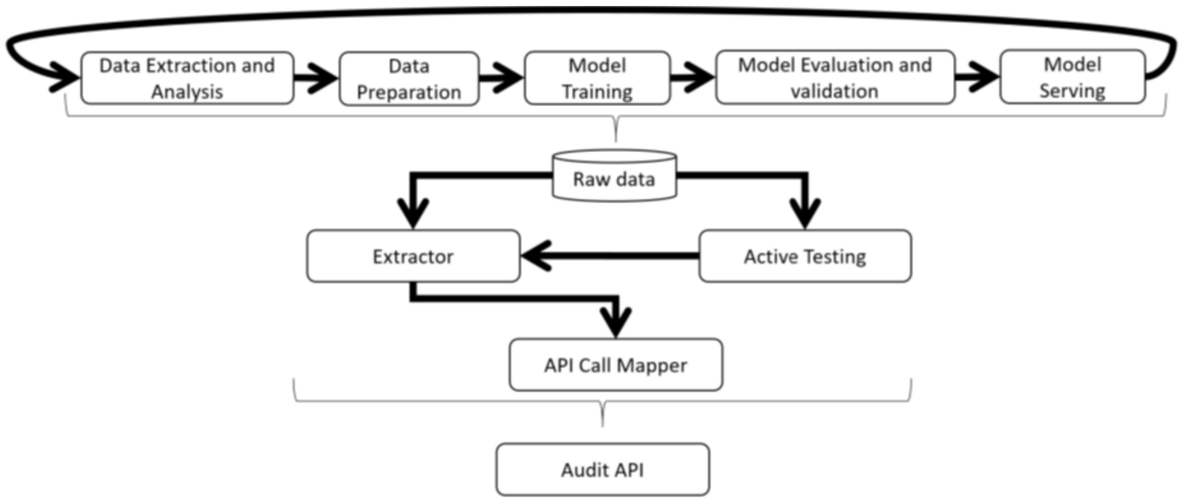


Figure 15 – Mapping of raw data to the Audit API

4.4.3 Evaluation/Audit

A metric from the development phase of ML models alone is not a quality criterion. The key figures determined in this way are initially only measured figures in themselves and require an evaluation by an qualified auditor to enable a quality assessment of the ML models. The interface thus enables a neutral and independent evaluation of the key figures by independent counterparts, which is of crucial importance for trustworthy quality control. Based on a sufficient representation of the measured values, the measured quality characteristics can be combined via an audit to form a quality assessment of the fulfillment of the requirements for an ML model. The strict separation of measured value collection and evaluation makes it easier to comply with the following properties for an objective quality assessment:

- Objectivity: no subjective influences of the measurer
- Reliability: same results when repeated
- Normalization: measurement result scale and comparability scale
- Comparability: measure can be put into relation with other measures
- Usefulness: measurable fulfillment of practical needs
- Validity: inference from measurable quantities to other characteristics
- Economy: minimal costs

The latter point (economy) also includes the question of how much effort one is willing to invest in continuous quality assessment. MLOps represents a development process in an infinite loop, in which the accompanying quality assessment must also be carried out continuously and is therefore subject to the tight time constraints of the development process. The cascaded approach for an audit-based quality assessment described below is intended to take this circumstance into account. It enables quality management to be built up in stages and either successively expanded or, depending on the respective application, limited to the appropriate effort or operational possibilities on a project-by-project basis.

4.4.3.1 Documentation

The lowest expansion stage is intended to support only sporadic, event-driven audits. The effort for the auditor or the auditing process should remain low. For this purpose, the key figures determined by the CABC Mapper tool and transferred via the interface are first stored (documented) in a repository in a version- and audit-proof manner, which can be done automatically as far as possible. In case of an audit, the historical sequence of the key figures can be viewed manually via standard queries. If necessary, the automatic generation of documents is supported, each of which contains an excerpt from the data stock of the repository. The documents can be generated on an ad hoc or regular basis, but a check based on the documents is only carried out irregularly.

The documentation for a software product consists of different parts, which are aligned to different target groups. In addition to documents that are aimed at the end user (user documentation), there are other documents that primarily focus on the development process. This also includes the group of so-called verification documents, which are intended to document the proper development of a software product (design decisions, quality control, etc.) for third parties (e.g., in the context of a legal review). The key figures collected via the interface can be used as the data basis for such a verification. Their version-safe storage or their processing into legally usable verification documents can be automated via the interface.

4.4.3.2 Reporting

Companies are having tremendous success implementing rapidly changing product requirements by adopting Agile methodologies (such as MLOps), which provide an iterative approach to the design and development of software and ML models. The Agile approach considers the constant changes that occur in technology development and allows teams to break up the lengthy requirements, development, and testing phases into smaller segments so they can ultimately deliver working software faster and more frequently. In the process, there is an ever-increasing flood of data and information accompanying and describing the development process that micro- and macro-management want to account for. Today, more than ever, companies need effective reporting tools to effectively manage the ever-increasing flood of data and transform it into understandable and meaningful information. The resulting insights are often used by developers as well as managers and executives as a basis for important operational and strategic decisions.

In the next expansion stage, the key figures continuously collected by the CABC Mapper tool will be used to control the development process. For this purpose, the key figures for ML development provided via the interface are displayed in a visually structured manner, e.g. in a dashboard, close to the time of their collection, and can thus be made available to all employees and decision-makers in an easily understandable and timely manner (daily report). This requires more effort for the meaningful design of the reports. However, the reports can then be generated or updated automatically. However, this should pay off, because important information no longer remains hidden in vast amounts of log data but can now be considered in every decision-making process.

4.4.3.3 Quality control

In the third expansion stage, the company's internal quality management is to be actively supported. Active quality control must not start with the output of a process. To be able to identify aberrations at an early stage, regular "section controls" in the form of "quality gates" are necessary in the process flow. Quality gates are points during a development project at which a decision is made on the release of the next project step based on clearly defined quality criteria. The decision should be based on comprehensible, measurable criteria. Regarding the development of ML models, the key figures collected via the interface can be used as the data basis for a decision. The basis for the decision will usually be a manual check by an internal but independent quality assurance team. This team can base its review on the reports from the previous expansion stage. Since these are only informative, additional documentation mechanisms must be introduced or used for the execution and documentation of the release decision for the quality gates. Based on a formalized evaluation of the key figures (e.g., by checking for threshold value violations), it may also be possible to introduce a release decision that can be automated.

4.4.3.4 Self-assessment

In the fourth stage of expansion, the quality characteristics of an ML model recorded during internal quality control are also made public as proof of quality to third parties (customers/users) after internal quality control has been completed (self-assessment). The product owner of the ML model is the auditee in this case, the audit is performed externally by the customer/user based on the key figures provided by the manufacturer. The key figures collected via the interface are summarized in a quality record describing the ML model, which is made publicly available (product sheet, online product presentation). Customers and users can use this proof to check whether the ML model meets their own quality requirements. In addition to the CABC mapper for collecting key figures, this also requires a platform that allows customers to check ML models largely independently and at their own discretion (audit configurator).

4.4.3.5 Certification

In safety-critical applications, the establishment of a formal certification process for ML models may be relevant (fifth expansion step). The added value here lies in the independent testing of the ML models by a third party that has no direct connection with either the development or the application of the product.

Certification processes are usually time-consuming and thus lengthy. The agile MLOps process, on the other hand, is oriented towards short development times in an infinite cycle, in which ML models are subjected to continuous expansion/improvement. Our aim is to establish a continuous certification framework for MLOps sharing aspects like evaluating the artifacts along the pipeline but extending in areas like actually mapping the artifacts to quality measurement inputs or taking high level quality standards like ISO 25012 as the baseline. We have developed and evaluated the approach of Continuous Audit-Based Certification (CABC) for security certification of cloud services in previous works and extend this approach hereby to MLOps.

Certification is establishing trust via a third party that verifies a grade of quality. Establishing trust in a traditional certification is achieved by introducing trustworthy third parties, usually organizations and humans. In comparison to a traditional point-in-time certification, we try to achieve the same grade of trustworthiness by mainly automated technical means. A certification is usually used to demonstrate compliance to customers or authorities. A self-proclamation from an auditee (like in the self-assessment scenario) does not generate the same amount of trust due to a conflict of interest as a third-party audit. For this reason, certification schemes often demand two or even three parties.

- A Certification body defines the rules for the certification process. It lays out the criteria under which an audit is conducted and defines the form of the audit report. It is the certification body that according to the audit report issues or suspends a certification.
- An auditing party conducts the audit under the rules of the certification body. It will verify the scope provided by the auditee for its suitability and its adherence to given requirements. It will verify the initial setup of the continuous auditing and facilitate the automated measurements and assessments at operation. The auditing party provides the means to receive the evidence from the auditee.
- The auditee is the owner of the ML-System. For establishing CABC the auditee needs to implement the technicalities in the MLOps process which means selecting the proper measurements for the evaluation of the quality attributes and making the actual technical implantation part of the MLOps process.

From a high-level view, CABC can be separated into a “preparation phase” and the “operation phase”:

- The preparation phase is an initial manual setup where the proper operationalization of the selected set of quality requirements takes place. Key actions in this phase are the definition of the scope, the identification of the measurements associated with each quality attribute, the determination of the frequencies at which each quality goal should be checked, as well as the implementation of the mapping of evidence and quality measurement input.
- In the operation phase the artifacts that are mapped to measurement inputs are being transmitted from the auditees’ premise to the auditor via the Audit API. As part of the assessment the auditor evaluates the received evidence and matches the result against the predefined values which reflects the quality goals. Based on the assessment a report gets generated mentioning each measurement result and the final verdict on the fulfillment of the quality goals. This report then gets transmitted to and evaluated by the certification body. Based on the evaluation of the report the registry for the specific continuous certification gets updated with either the new certificate or the revocation of the old one.

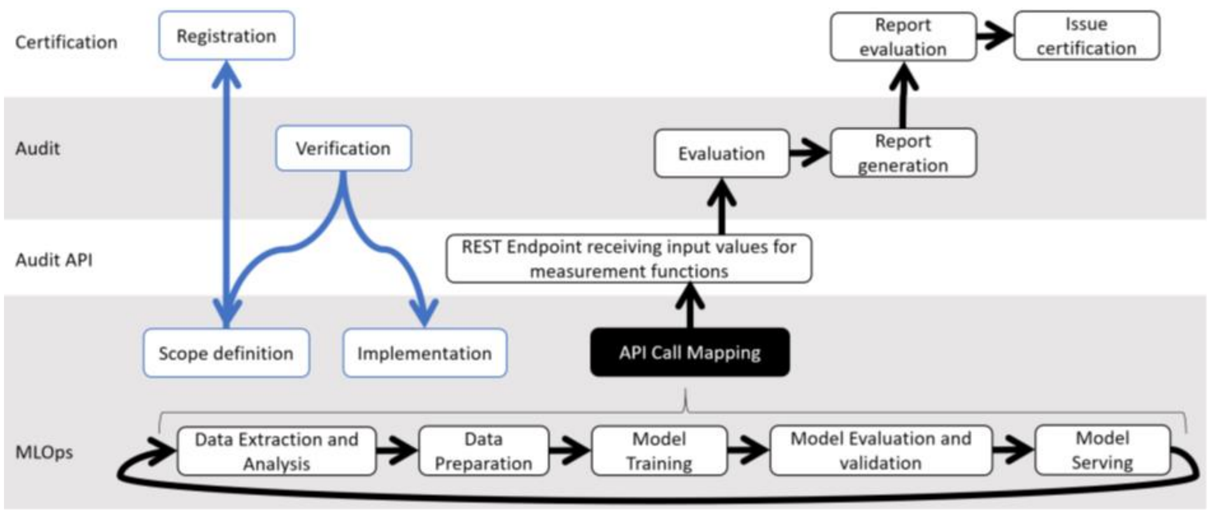


Figure 16 – Hierarchically staggered approach of the Continuous Audit-based Quality Assurance Methodology for certification

5 Future Steps

For the next iterations of the work package 4 partners aim to focus more on the elaboration and evolution of the introduced IML4E MLOPs framework. A good source for expanding the capabilities of the framework will be the learnings provided by implementing and operating the IML4E technologies and use cases. Thus, a final version of the IML4E framework is expected to be delivered close to the end of the IML4E project when IML4E technologies are going to be finalized and utilized.

In regards to the next steps of WP4 partners aim to also evaluate the introduced experimentation platform, provide a more detailed educational material and also deliver a reference use case that demonstrates the utilization of the MLOps platform and the applicability of the IML4E framework.

In summary the next steps are the:

a. elaboration and expansion of the introduced IML4E MLOps methodologies and framework

b. Evaluation of the experimentation MLOPs platform

c. delivery of use case-based training material

d. implementation of a reference use case that demonstrates the integration of the MLOps platform and IML4E's MLOps methodologies.

Trying to highlight the future capabilities of the IML4E framework it is envisioned that part of the framework offering will be the description of the different steps and actions that must be taken in the different phases of the ML products. The IML4E framework will act as gatekeeper for the trusted execution of ML products and will provide an interface for the model and business owners to define expectation and quality thresholds on the ML pipelines. Through the IML4E framework and in particular through the continuous validation and monitoring mechanisms (that are part of the framework) the ML owners are going to be in position to adjust their control metrics in their productionized ML systems. The above is one feature of the IML4E framework, another interesting feature of the framework that partners aim to expand would be the MLOps maturity assessment that will help enterprises to build ML/AI capabilities roadmaps according to their needs.

6 Conclusions and Summary

Within the scope of the second deliverable of WP4 the partners delivered a first MVP (minimum viable product) version of the ML experimentation platform. This platform serves as an enabler for the operations of the IML4E technologies presented in Section 3.

In addition, a deliverable of the second iteration of the WP4 was the introduction of the initial ideas and the basis of the IML4E MLOps framework. The introduced framework aims to govern the end-to-end Lifecyle of machine learning products.

Finally, deliverables of the second iteration of the work package 4 can be also considered the expansion of the MLOps reference architecture (Section 3) and the MLOps maturity level matrixes presented (Section 7). The maturity level matrixes can serve as a compass for road mapping the ML/AI and Data infrastructure initiatives within enterprises.

Concluding IML4E partners aim to evaluate and evolve the deliverables of WP4 by the adaptation of the IML4E use cases and technologies to the experimentation platform and the introduced framework.

References

Akkiraju et al. (2018). Characterizing machine learning process: A maturity framework, arXiv:1811.04871 [cs], Nov. 2018.

Sothilingam, Yu, and Senderovich (2020). Towards Higher Maturity for Machine Learning: A Conceptual Modelling Approach, IJ, Bd. 5, Nr. 1, S. 80–97, Jan. 2020, doi: 10.33137/ijournal.v5i1.33476.

Santhanam, Farchi, and Pankratius (2019). Engineering Reliable Deep Learning Systems, arXiv:1910.12582 [cs], Oct. 2019.

Wood, M. et al. (2019). Safety First for Automated Driving. Abrufbar unter https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf.

Amershi, Saleema, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann (2019). Software Engineering for Machine Learning: A Case Study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 291–300. Montreal, QC, Canada: IEEE. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR), Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) and Rüdiger Wirth (DaimlerChrysler) (2018). CRISP-DM 1.0 - Step-by-step data mining guide. https://www.the-modeling-agency.com/crisp-dm.pdf.

Studer, Thanh, Drescher, Hanuschkin, Winkler, Peters, and Mueller (2020). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. arXiv:2003.05155 [cs, stat], March. http://arxiv.org/abs/2003.05155.

Nott, C. A (2015). Maturity Model for Big Data and Analytics. IBM. https://www.ibm.com/developerworks/community/blogs/bigdataanalyt ics/entry/A_maturity_model_for_big_data_and_analytics?lang=en_u s

Schmarzo, B. (2016). Big Data Business Model Maturity Index Guide. Dell EMC. https://infocus.dellemc.com/william_schmarzo/big- data-business-model-maturity-index-guide/.

Dhanuka, V. (2017). Hortonworks Big Data Maturity Model. Hortonworks. http://hortonworks.com/wp-content/uploads/2016/04/Hortonworks-Big-Data-Maturity- Assessment.pdf.

El-Darwiche, B., Koch, V., Meer, D., Shehadi, R. T., & Tohme, W. (2014). Big data maturity: An action plan for policymakers and executives. The global information technology report, 43, 51.

Radcliffe, J. (2015). Leverage a Big Data Maturity Model to Build Your Big Data Roadmap. Radcliffe Advisory Services Ltd. http://www.radcliffeadvisory.com/research/download.php?file=RAS_ BD_MatMod.pdf

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

IML4E-D2.3-V1.0 (2022a). First version of tools for data collection, processing, and valorisation, https://itea4.org/project/iml4e.html, December 2022

IML4E-D3.3-V1.0 (2022b). First version of tools for advanced model engineering, https://itea4.org/project/iml4e.html, December 2022

IML4E-D4.1-V1.0 (2022c). Requirements for the IML4E online experimentation and training platform, https://itea4.org/project/iml4e.html, June 2022