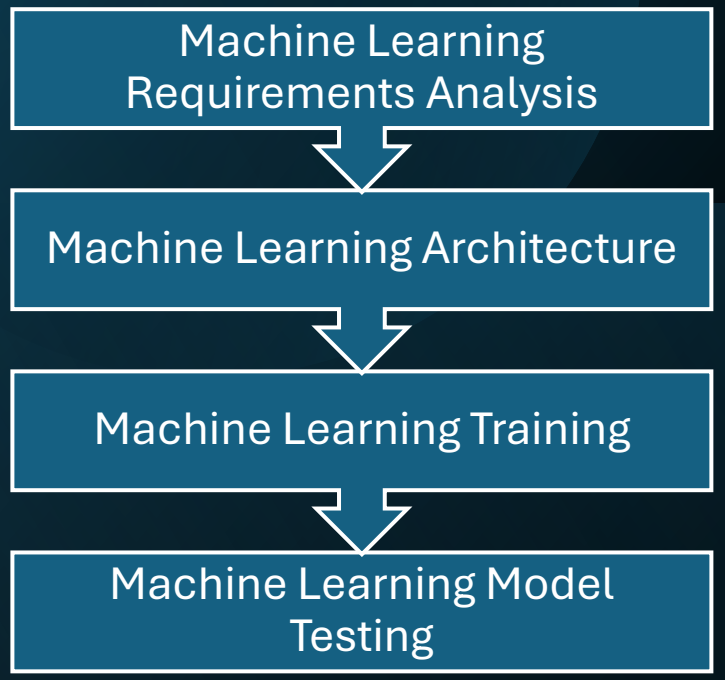
MLE.1-3 Machine Learning Engineering

Process Group

Overview of the ML Engineering Process Group



MLE focuses on integrating ML-specific processes into engineering systems for robust, explainable, and efficient ML models

1. Purpose of MLE Processes

- Refines ML-specific requirements, architecture, and training approaches.
- Ensures alignment with software and system engineering principles.
- Improves reliability, explainability, and scalability of ML systems.

2. Integration with ASPICE Framework

- Aligns ML development with ASPICE process requirements.
- Links ML processes to system and software engineering practices.
- Facilitates structured, traceable, and compliant ML engineering workflows.

3. Focus Areas of MLE

- MLE.1: Refines and structures ML requirements.
- MLE.2: Designs ML architectures for deployment and training.
- MLE.3: Develops, trains, and validates ML models.
- MLE.4: Tests compliance of ML models with defined requirements..

Automotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination

Software Engineering vs. Machine Learning Engineering

SW engineering focuses on software systems, while ML engineering specializes in building and deploying data-driven models

1. Scope of Work 🌐

- **SW Engineering:** Develops structured software systems, focusing on deterministic behavior.
- **ML Engineering:** Designs data-driven models with learning capabilities and adaptability.
- **Overlap:** ML engineering leverages software engineering principles for model deployment.

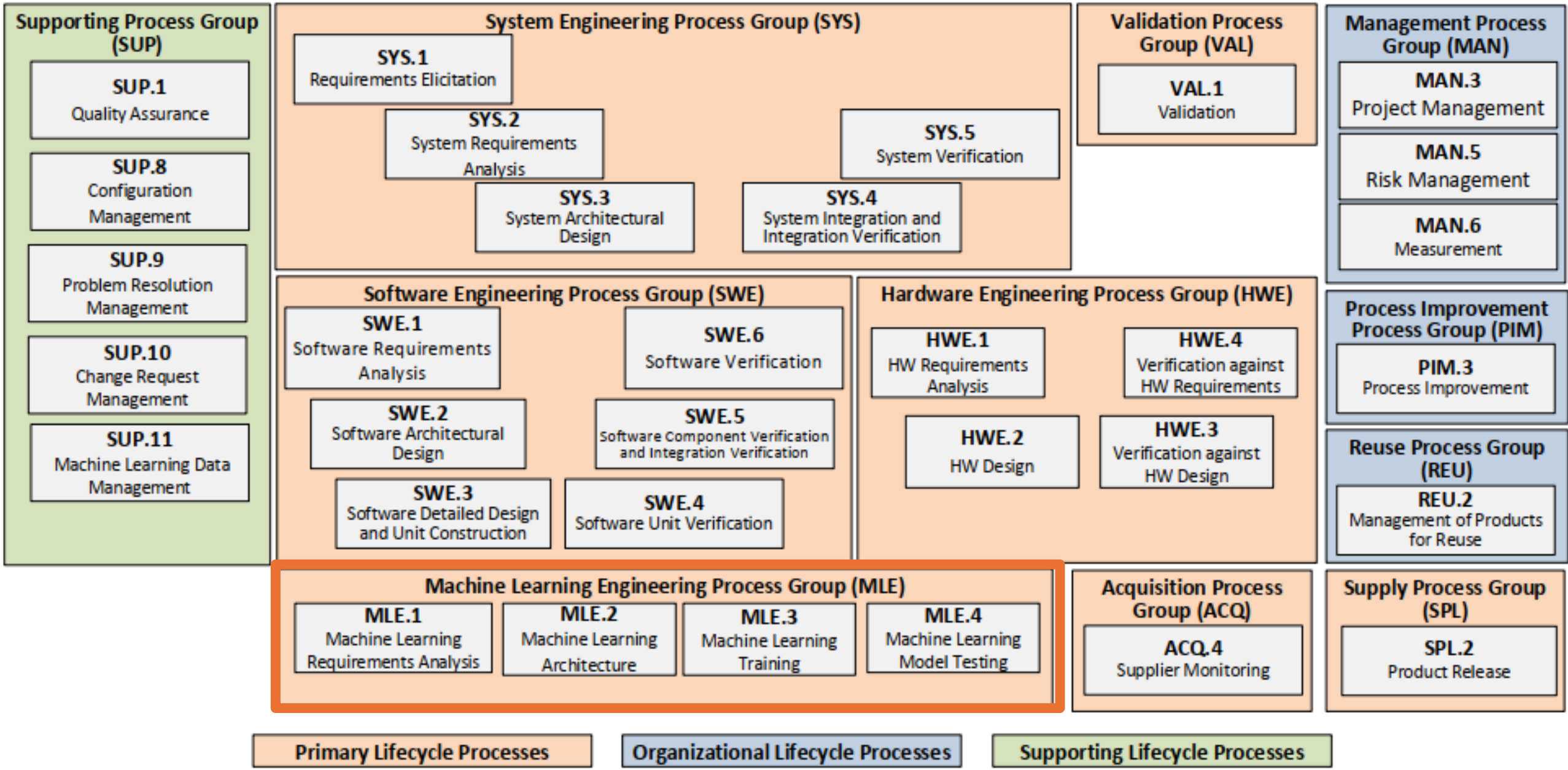
2. Requirements and Design ✍️

- **SW Engineering:** Defines functional and non-functional requirements for software.
- **ML Engineering:** Includes data requirements, model specifications, and performance metrics.
- **Collaboration:** ML engineering uses outputs from software engineering for integration.

3. Verification and Validation 🔄

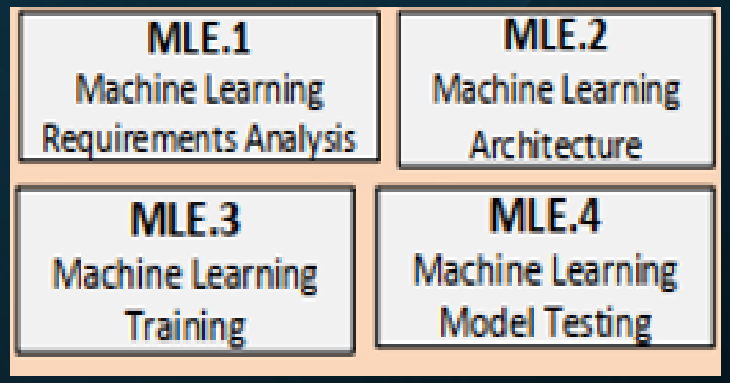
- **SW Engineering**: Verifies software systems against defined requirements.
- **ML Engineering:** Validates model accuracy, robustness, and performance in real-world scenarios.
- **Dependency:** ML validation includes software framework verification to ensure end-to-end reliability.

Automotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination



ASPICE – **A**utomotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination

Purpose and Scope of the SWE Process Group



MLE process group, emphasize its integration with the overall ASPICE lifecycle.

1. Purpose of the SWE Process Group

- Establishes a structured approach for ML development.
- Ensures traceability from ML requirements to implementation and evaluation.
- Aligns ML-specific processes with ASPICE standards.

2. Scope of the SWE Processes

- Includes ML requirements, architecture, training, and validation.
- Supports the integration of ML systems in real-world applications.
- Applies to ML-enabled safety-critical and high-performance systems.

3. Connection to ASPICE Framework

- Links ML development to traditional engineering practices.
- Ensures bidirectional traceability between ML models and system-level requirements.
- Promotes consistency and collaboration across engineering disciplines.

Software Requirements vs. Machine Learning Requirements

SW requirements define functional and non-functional needs, while ML requirements emphasize data, performance metrics, and learning objectives

1. Definition and Focus

- **Software Requirements:** Detail what the software must do and its constraints (functional and non-functional).
- **ML Requirements:** Include objectives for the model, data dependencies, and success metrics (e.g., accuracy, precision).
- **Overlap:** Both must align with higher-level system requirements.

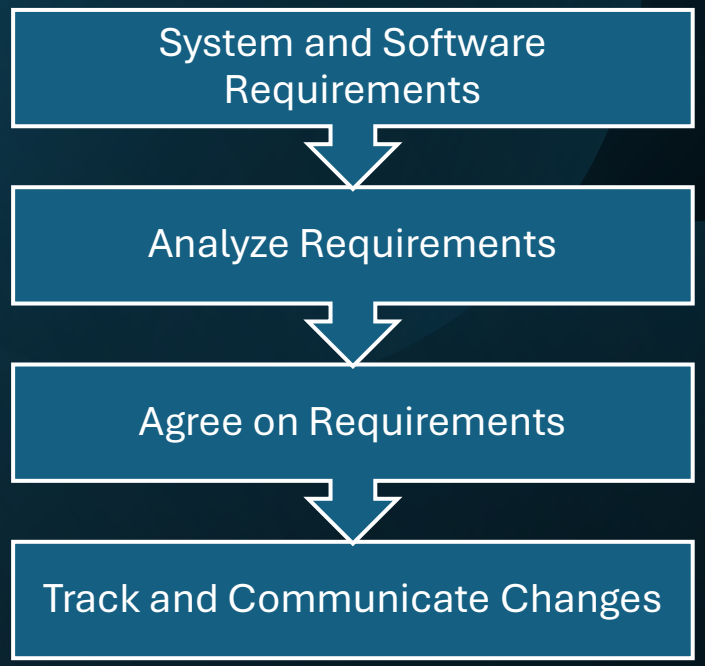
2. Specification Criteria

- **Software Requirements:** Derived from stakeholder needs, emphasizing deterministic behavior.
- **ML Requirements:** Specify datasets, features, and expected performance metrics (e.g., F1-score, recall).
- **Dependency:** ML requirements often originate from broader software requirements

3. Validation and Traceability

- **Software Requirements:** Validated through traditional testing methods against system expectations.
- **ML Requirements:** Validated using model evaluation metrics and test datasets.
- **Collaboration:** Traceability ensures alignment between ML outcomes and software objectives.

MLE.1 Machine Learning Requirements Analysis Overview



MLE.1 focuses on defining and analyzing ML requirements to ensure alignment with system and software objectives

1. Purpose of Machine Learning Requirements Analysis

- Identify ML-specific requirements, including data, model objectives, and performance metrics.
- Align ML requirements with higher-level system and software goals.
- Address unique ML challenges, such as data dependencies and evaluation criteria.

2. Process Scope

- Includes functional and non-functional requirements specific to ML models.
- Emphasizes data requirements, preprocessing needs, and expected model behavior.
- Focuses on traceability between ML requirements and system/software requirements.

3. Expected Outcomes

- Well-defined ML requirements, ready for architecture and development phases.
- Clear traceability to system-level requirements and operational goals.
- Identification of validation criteria to measure ML model success.

Process ID
MLE.1
Process name
Machine Learning Requirements Analysis
Process purpose
The purpose is to refine the machine learning-related software requirements into a set of ML requirements.
Process outcomes
1) the ML requirements, including ML data requirements, are identified and specified based on the software requirements and the components of the software architecture 2) ML requirements are structured and prioritized 3) ML requirements are analyzed for correctness and verifiability 4) the impact of ML requirements on the ML operating environment is analyzed 5) consistency and bidirectional traceability are established between ML requirements and software requirements, and between ML requirements and software architecture 6) the ML requirements are agreed and communicated to all affected parties

MLE.1 – Machine Learning Requirements Analysis	Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5	Outcome 6
Output Information Items						
17-00 Requirement	X	X				
17-54 Requirement attribute		X	X			
13-52 Communication evidence						X
13-51 Consistency evidence					X	
15-51 Analysis results			X	X		
Base Practices						
BP1: Specify ML requirements	X					
BP2: Structure ML requirements		X				
BP3: Analyze ML requirements			X			
BP4: Analyze the impact on the ML operating environment				X		
BP5: Ensure consistency and establish bidirectional traceability					X	
BP6: Communicate agreed ML requirements						X

Base practices

MLE.1.BP1: Specify ML requirements. Use the software requirements and the software architecture to identify and specify functional and non-functional ML requirements, as well as ML data requirements specifying data characteristics and their expected distributions.

Note 1: Non-functional requirements may include relevant characteristics of the ODD and KPIs as robustness, performance and level of trustworthiness.

Note 2: The ML data requirements are input for SUP.11 Machine Learning Data Management but also for other MLE processes.

Note 3: In case of ML development only, stakeholder requirements represent the software requirements.

MLE.1.BP2: Structure ML requirements. Structure and prioritize the ML requirements.

Note 4: Examples for structuring criteria can be grouping (e.g. by functionality) or variants identification.

Note 5: Prioritization can be done according to project or stakeholder needs via e.g. definition of release scopes. Refer to SPL.2.BP1.

MLE.1.BP3: Analyze ML requirements. Analyze the specified ML requirements including their interdependencies to ensure correctness, technical feasibility, and ability for machine learning model testing, and to support project management regarding project estimates.

Note 6: See MAN.3.BP3 for project feasibility and MAN.3.BP5 for project estimates.

MLE.1.BP4: Analyze the impact on the ML operating environment. Analyze the impact that the ML requirements will have on interfaces of software components and the ML operating environment.

Note 7: The ML operating environment is defined as the infrastructure and information which both the trained ML model and the deployed ML model need for execution.

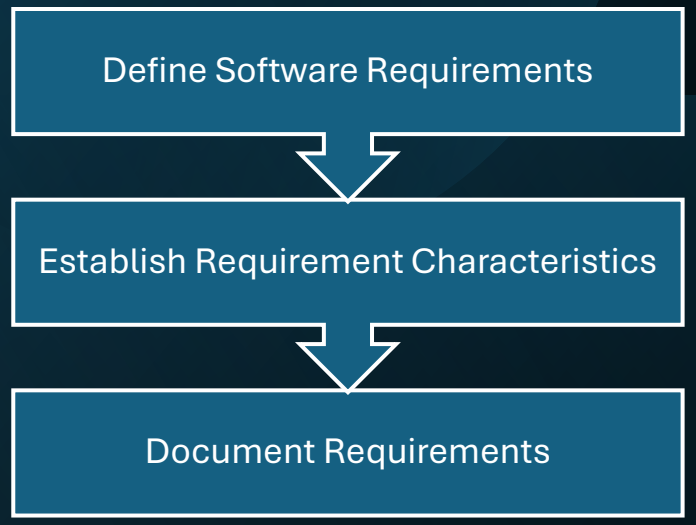
MLE.1.BP5: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between ML requirements and software requirements and between ML requirements and the software architecture.

Note 8: Bidirectional traceability supports consistency, facilitates impact analyses of change requests, and verification coverage demonstration.

Note 9: Redundant traceability is not intended, but at least one out of the given traceability paths.

MLE.1.BP6: Communicate agreed ML requirements. Communicate the agreed ML requirements, and the results of the impact analysis on the ML operating environment to all affected parties.

BP1: Specify ML Requirements



System requirements and architecture serve as inputs for defining actionable software requirements

> **MLE.1.BP1: Specify ML requirements.** Use the software requirements and the software architecture to identify and specify functional and non-functional ML requirements, as well as ML data requirements specifying data characteristics and their expected distributions.
>
> *Note 1: Non-functional requirements may include relevant characteristics of the ODD and KPIs as robustness, performance and level of trustworthiness.*
>
> *Note 2: The ML data requirements are input for SUP.11 Machine Learning Data Management but also for other MLE processes.*
>
> *Note 3: In case of ML development only, stakeholder requirements represent the software requirements.*

1. Define Functional Requirements

- Specify model objectives (e.g., classification, regression, clustering).
- Outline data preprocessing steps required for model training.
- Identify expected model outputs and behaviors under different scenarios.

2. Specify Non-Functional Requirements

- Include performance metrics (e.g., accuracy, precision, recall).
- Define constraints such as latency, scalability, and resource usage.
- Address robustness to noisy or incomplete data.

3. Document Data Requirements

- Detail input datasets, including sources and formats.
- Specify data quality and preprocessing requirements.
- Identify any dependencies on external data sources or pipelines.

BP2: Structure ML Requirements

BP2 focuses on structuring ML requirements to ensure alignment, prioritization, and traceability

MLE.1.BP2: Structure ML requirements. Structure and prioritize the ML requirements.

Note 4: Examples for structuring criteria can be grouping (e.g. by functionality) or variants identification.

Note 5: Prioritization can be done according to project or stakeholder needs via e.g. definition of release scopes. Refer to SPL.2.BP1.

1. Group Requirements by Categories

- Separate functional, non-functional, and data requirements.
- Identify dependencies between different requirement types.
- Ensure that categories reflect the scope of the ML solution.

2. Prioritize Requirements Based on Impact

- Rank requirements based on criticality and feasibility.
- Focus on high-impact requirements like essential performance metrics.
- Use prioritization techniques such as MoSCoW (Must Have, Should Have, Could Have, Won't Have).

3. Align Requirements with System Goals

- Ensure traceability to higher-level system and software requirements.
- Verify that structured requirements align with stakeholder expectations.
- Address alignment gaps early in the process.

BP3: Analyze ML Requirements

Ensures ML requirements are analyzed for feasibility, consistency, and alignment with project goals

MLE.1.BP3: Analyze ML requirements. Analyze the specified ML requirements including their interdependencies to ensure correctness, technical feasibility, and ability for machine learning model testing, and to support project management regarding project estimates.

Note 6: See MAN.3.BP3 for project feasibility and MAN.3.BP5 for project estimates.

1. Evaluate Feasibility

- Assess whether the requirements are achievable with available data and resources.
- Identify constraints such as computational power and dataset size.
- Address potential risks related to data quality or availability.

2. Verify Requirement Consistency

- Ensure there are no conflicting or redundant requirements.
- Check alignment between functional and non-functional requirements.
- Resolve inconsistencies to avoid design bottlenecks.

3. Check Requirement Completeness

- Confirm that all necessary requirements are captured.
- Include edge cases, error handling, and fallback mechanisms.
- Validate against stakeholder and system-level expectations.

BP4: Analyze the Impact on the Operating Environment

Evaluates the interaction of ML requirements with the operating environment to ensure compatibility and performance

MLE.1.BP4: Analyze the impact on the ML operating environment. Analyze the impact that the ML requirements will have on interfaces of software components and the ML operating environment.

Note 7: The ML operating environment is defined as the infrastructure and information which both the trained ML model and the deployed ML model need for execution.

1. Evaluate Deployment Constraints 🛠️

- Assess hardware and software requirements for deploying the ML model.
- Identify limitations in storage, computational power, and network connectivity.
- Ensure compatibility with existing infrastructure.

2. Consider Environmental Factors 🌐

- Address data security, privacy, and compliance requirements.
- Analyze the operational conditions, such as latency and real-time performance needs.
- Account for external factors like data sources and variability.

3. Simulate Operational Scenarios ✅

- Test how the model performs in its intended deployment environment.
- Identify bottlenecks or risks in real-world conditions.
- Validate the feasibility of meeting performance and reliability goals.

BP5: Ensure Consistency and Traceability

Ensures ML requirements are consistent and traceable to higher-level and derived artifacts

MLE.1.BP5: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between ML requirements and software requirements and between ML requirements and the software architecture.

Note 8: Bidirectional traceability supports consistency, facilitates impact analyses of change requests, and verification coverage demonstration.

Note 9: Redundant traceability is not intended, but at least one out of the given traceability paths.

1. Establish Bidirectional Traceability

- Link ML requirements to system and software requirements.
- Create traceability to architectural elements and validation criteria.
- Use tools like Jira or DOORS to document and manage traceability.

2. Verify Requirement Consistency

- Ensure alignment across functional, non-functional, and data requirements.
- Cross-check ML requirements with system-level goals and constraints.
- Regularly review and update requirements to reflect changes.

3. Document Traceability Relationships

- Clearly define traceability links for all requirements.
- Ensure documentation supports collaboration and transparency.
- Use visual representations (e.g., traceability matrices) for clarity.

BP6: Communicate Agreed ML Requirements

Ensures ML requirements are clearly communicated to stakeholders for alignment and approval

> **MLE.1.BP6: Communicate agreed ML requirements.** Communicate the agreed ML requirements, and the results of the impact analysis on the ML operating environment to all affected parties.

1. Prepare Comprehensive Documentation

- Summarize functional, non-functional, and data requirements.
- Include traceability relationships and validation criteria.
- Use clear, concise language tailored to the audience.

2. Engage Stakeholders for Feedback

- Present requirements in meetings or reviews.
- Address stakeholder questions and incorporate feedback.
- Ensure all parties understand the implications of the requirements.

3. Use Collaborative Tools

- Share documentation through platforms like Confluence or SharePoint.
- Use collaborative features for annotations and comments.
- Ensure version control for all requirement artifacts..

Challenges in MLE.1 - ML Requirements Analysis

ML requirements analysis faces unique challenges in defining, aligning, and validating data-driven needs

1. Defining Clear and Feasible Requirements 🛠️

- **Challenge:** Difficulty in articulating ML-specific functional and non-functional requirements.
- **Impact:** Ambiguity may lead to misaligned development efforts.
- **Solution:** Use structured requirement elicitation techniques and iterative validation.

2. Managing Data Dependencies 🌐

- **Challenge:** Reliance on diverse, dynamic, or incomplete datasets.
- **Impact:** Inconsistent data can compromise model training and performance.
- **Solution:** Define robust data requirements and preprocessing standards early.

3. Ensuring Stakeholder Alignment ✅

- **Challenge:** Differences in understanding of ML capabilities and limitations.
- **Impact:** Miscommunication can delay approval and increase rework.
- **Solution:** Regular stakeholder engagement and use of collaborative tools for clarity.

MLE.2 – Machine Learning Architecture Overview

MLE.2 focuses on designing scalable, explainable, and deployment-ready ML architectures

1. Purpose of Machine Learning Architecture

- Develop a structured architecture for ML models and workflows.
- Ensure scalability, explainability, and integration readiness.
- Align architecture with ML requirements and system goals.

2. Process Scope

- Covers data pipelines, model components, and deployment strategies.
- Focuses on defining interfaces, resources, and hyperparameter management.
- Integrates ML components with broader system and software architecture.

3. Expected Outcomes

- Documented ML architecture supporting training, deployment, and validation.
- Defined interfaces and resource objectives for ML components.
- Traceability established between architecture and ML requirements.

Software Architecture vs. ML Architecture

Software architecture structures software components and interactions, while ML architecture focuses on data pipelines, models, and training workflows

1. Scope and Objectives

- **Software Architecture:** Defines software modules, interfaces, and their interactions.
- **ML Architecture:** Designs data pipelines, feature extraction, model components, and training workflows.
- **Overlap:** Both aim to ensure scalability, reliability, and alignment with requirements.

2. Design Elements

- **Software Architecture:** Emphasizes modularity, system-wide interaction, and interface design.
- **ML Architecture:** Includes hyperparameter tuning, training infrastructure, and deployment strategies.
- **Dependency:** ML architecture operates within the broader software architecture.

3. Validation and Traceability

- **Software Architecture:** Validated via functional testing and integration reviews.
- **ML Architecture:** Evaluated using model performance metrics, operational testing, and scalability checks.
- **Collaboration:** Ensures ML components align with the software structure and system-level goals.

Process ID
MLE.2
Process name
Machine Learning Architecture
Process purpose
The purpose is to establish an ML architecture supporting training and deployment, consistent with the ML requirements, and to evaluate the ML architecture against defined criteria.
Process outcomes
1) an ML architecture is developed 2) hyperparameter ranges and initial values are determined as a basis for the training 3) evaluation of ML architectural elements is conducted 4) interfaces of the ML architectural elements are defined 5) resource consumption objectives for the ML architectural elements are defined 6) consistency and bidirectional traceability are established between the ML architectural elements and the ML requirements 7) the ML architecture is agreed and communicated to all affected parties

MLE.2 – Machine Learning Architecture	Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5	Outcome 6	Outcome 7
Output Information Items							
04-51 ML architecture	X	X	X	X	X		
13-52 Communication evidence							X
13-51 Consistency evidence						X	
01-54 Hyperparameter	X	X					
15-51 Analysis results	X		X				
Base Practices							
BP1: Develop ML architecture	X						
BP2: Determine hyperparameter ranges and initial values.		X					
BP3: Evaluate ML architectural elements			X				
BP4: Define interfaces of the ML architectural elements				X			
BP5: Define resource consumption objectives for the ML architectural elements					X		
BP6: Ensure consistency and establish bidirectional traceability						X	
BP7: Communicate agreed ML architecture							X

Base practices

MLE.2.BP1: Develop ML architecture. Develop and document the ML architecture that specifies ML architectural elements including details of the ML model, pre- and postprocessing, and hyperparameters which are required to create, train, test, and deploy the ML model.

Note 1: Necessary details of the ML model may include layers, activation functions, and backpropagation. The level of detail of the ML model may not need to cover aspects like single neurons.

Note 2: The details of the ML model may differ between the ML model used during training and the deployed ML model.

MLE.2.BP2: Determine hyperparameter ranges and initial values. Determine and document the hyperparameter ranges and the initial values as a basis for the training.

MLE.2.BP3: Evaluate ML architectural elements. Define evaluation criteria for the ML architectural elements. Evaluate ML architectural elements according to the defined criteria.

Note 3: Trustworthiness and explainability might be criteria for the evaluation of the ML architectural elements.

MLE.2.BP4: Define interfaces of the ML architectural elements. Determine and document the internal and external interfaces of each ML architectural element including its interfaces to related software components.

MLE.2.BP5: Define resource consumption objectives for the ML architectural elements. Determine and document the resource consumption objectives for all relevant ML architectural elements during training and deployment.

MLE.2.BP6: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the ML architectural elements and the ML requirements.

Note 4: Bidirectional traceability supports consistency, and facilitates impact analyses of change requests, and verification coverage demonstration.

Note 5: The bidirectional traceability should be established on a reasonable level of abstraction to the ML architectural elements.

MLE.2.BP7: Communicate agreed ML architecture. Inform all affected parties about the agreed ML architecture including the details of the ML model and the initial hyperparameter values.

BP1: Develop Machine Learning Architecture

Creating a robust ML architecture that ensures scalability, performance, and integration readiness

Base practices

MLE.2.BP1: Develop ML architecture. Develop and document the ML architecture that specifies ML architectural elements including details of the ML model, pre- and postprocessing, and hyperparameters which are required to create, train, test, and deploy the ML model.

Note 1: Necessary details of the ML model may include layers, activation functions, and backpropagation. The level of detail of the ML model may not need to cover aspects like single neurons.

Note 2: The details of the ML model may differ between the ML model used during training and the deployed ML model.

1. Define Static and Dynamic Aspects

- Identify key components such as data preprocessing, model training, and validation modules.
- Design workflows for data flow, model updates, and deployment pipelines.
- Document architectural components and their relationships.

2. Support Scalability and Performance

- Optimize architecture for varying data volumes and computational loads.
- Incorporate techniques for distributed processing and resource allocation.
- Plan for iterative improvements and model retraining cycles.

3. Align with System and Software Architecture

- Ensure compatibility with system-level architecture and operational goals.
- Define interfaces for data exchange between ML components and other system modules.
- Establish traceability links to ML requirements and system architecture.

BP2: Determine Hyperparameter Ranges and Initial Values

Selecting and defining hyperparameter ranges to optimize model training and performance

MLE.2.BP2: Determine hyperparameter ranges and initial values. Determine and document the hyperparameter ranges and the initial values as a basis for the training.

1. Identify Critical Hyperparameters

- Define key parameters such as learning rate, batch size, and regularization terms.
- Prioritize hyperparameters that significantly impact model performance.
- Document the purpose and impact of each parameter.

2. Determine Feasible Ranges

- Establish ranges based on domain knowledge and preliminary experiments.
- Use techniques like grid search, random search, or Bayesian optimization to refine values.
- Ensure ranges are adaptable to different datasets and training conditions.

3. Document and Align with Requirements

- Ensure dynamic behavior supports system use cases and software requirements.
- Validate that interactions handle both typical and edge cases.
- Update models as new scenarios or requirements emerge.

BP3: Evaluate ML Architectural Elements

Ensures that ML architectural elements meet performance, robustness, and scalability criteria

MLE.2.BP3: Evaluate ML architectural elements. Define evaluation criteria for the ML architectural elements. Evaluate ML architectural elements according to the defined criteria.

Note 3: Trustworthiness and explainability might be criteria for the evaluation of the ML architectural elements.

1. Assess Feasibility and Performance

- Evaluate the practicality of architectural components for the defined requirements.
- Test for expected performance under realistic workloads.
- Identify bottlenecks or limitations in data flow and computational resources.

2. Ensure Robustness and Fault Tolerance

- Validate the system's ability to handle noisy or incomplete data.
- Include redundancy in critical components to minimize risks.
- Test error-handling mechanisms for various failure scenarios.

3. Analyze Scalability and Flexibility

- Verify the architecture’s ability to scale with increasing data volumes and user demands.
- Ensure flexibility for future updates, including retraining and new data sources.
- Use simulations to predict behavior under peak load conditions.

BP4: Define Interfaces of ML Architectural Elements

Specify interfaces for seamless interaction between ML components and other system elements

> **MLE.2.BP4: Define interfaces of the ML architectural elements.** Determine and document the internal and external interfaces of each ML architectural element including its interfaces to related software components.

1. Identify Interface Requirements

- Define inputs and outputs for each architectural component (e.g., data preprocessing, model training).
- Specify data formats, communication protocols, and performance expectations.
- Ensure compatibility with system-level interfaces and external data sources.

2. Design for Modularity and Scalability

- Create interfaces that allow easy integration of new components or data pipelines.
- Ensure modularity to facilitate updates and iterative improvements.
- Validate interfaces for performance under varying workloads.

3. Document and Test Interfaces

- Record detailed interface specifications for all components.
- Test interfaces in simulated and real-world scenarios to validate compatibility.
- Use tools like API testing frameworks to ensure reliability.

BP5: Define Resource Consumption Objectives

Ensure that resource consumption is planned and optimized for ML components during training and deployment

MLE.2.BP5: Define resource consumption objectives for the ML architectural elements.
Determine and document the resource consumption objectives for all relevant ML architectural elements during training and deployment.

1. Identify Resource Requirements

- Specify computational needs for training (e.g., GPU, CPU utilization).
- Define memory requirements for data processing and model storage.
- Include bandwidth and network latency considerations for deployment.

2. Optimize Resource Allocation

- Plan for efficient utilization of hardware and software resources.
- Use profiling tools to identify resource bottlenecks during execution.
- Adjust resource objectives based on iterative testing and performance metrics.

3. Document Resource Objectives

- Record maximum and minimum resource thresholds for ML components.
- Ensure alignment with system-level resource constraints and operational goals.
- Include scalability provisions to adapt to increased data or user demands.

BP6: Ensure Consistency and Traceability

Ensure alignment between ML architecture and requirements, establishing traceability for seamless integration

MLE.2.BP6: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the ML architectural elements and the ML requirements.

Note 4: Bidirectional traceability supports consistency, and facilitates impact analyses of change requests, and verification coverage demonstration.

Note 5: The bidirectional traceability should be established on a reasonable level of abstraction to the ML architectural elements.

1. Establish Traceability Links

- Map ML architectural elements to specific ML requirements.
- Trace dependencies to system and software architecture.
- Ensure bidirectional links to track changes in either requirements or architecture.

2. Validate Architectural Consistency

- Cross-check architectural elements for compliance with defined requirements.
- Ensure compatibility between different ML components and their interfaces.
- Address inconsistencies before moving to implementation.

3. Document Traceability Relationships

- Use traceability matrices to link requirements, architecture, and system goals.
- Regularly update traceability documentation to reflect changes.
- Leverage tools like Jira, DOORS, or Confluence to manage traceability.

BP7: Communicate Agreed ML Architecture

Ensures the ML architecture is clearly documented and shared for stakeholder alignment

MLE.2.BP7: Communicate agreed ML architecture. Inform all affected parties about the agreed ML architecture including the details of the ML model and the initial hyperparameter values.

1. Prepare Comprehensive Documentation

- Include all architectural elements, interfaces, and resource objectives.
- Highlight alignment with ML requirements and traceability relationships.
- Use visual representations like diagrams and flowcharts for clarity.

2. Engage Stakeholders for Review

- Present the architecture in stakeholder meetings or workshops.
- Address feedback to ensure all concerns and expectations are met.
- Facilitate discussions to clarify technical and operational aspects.

3. Leverage Collaborative Tools

- Share architecture documents via platforms like Confluence or SharePoint.
- Enable annotations and discussions for iterative improvements.
- Ensure version control to manage updates and changes.

Challenges in MLE.2 - ML Architecture Design

ML architecture design faces unique challenges in scalability, integration, and performance optimization

1. Balancing Complexity and Scalability

- **Challenge:** Designing architectures that handle growing data and workload complexity.
- **Impact:** Over-engineered designs can hinder performance; under-engineered designs may fail under load.
- **Solution:** Iteratively design and test scalability with modular components.

2. Ensuring Seamless Integration

- **Challenge:** Aligning ML architecture with system and software architecture.
- **Impact:** Poor integration leads to inefficiencies and deployment bottlenecks.
- **Solution:** Define clear interfaces and traceability between architecture levels.

3. Optimizing Resource Consumption

- **Challenge:** Managing computational, memory, and energy requirements effectively.
- **Impact:** Resource constraints can limit model performance or delay deployment.
- **Solution:** Use profiling tools and iterative testing to optimize resource allocation.

MLE.3 - Machine Learning Training Overview

MLE.3 focuses on training, validating, and optimizing ML models to meet defined requirements

1. Purpose of Machine Learning Training

- Develop ML models using defined requirements and architecture.
- Ensure models meet performance, reliability, and operational goals.
- Address challenges in data quality, training efficiency, and evaluation metrics.

2. Process Scope

- Includes data preparation, training, and validation workflows.
- Covers optimization of hyperparameters and model evaluation criteria.
- Integrates iterative improvement cycles based on validation results.

3. Expected Outcomes

- Trained ML models ready for deployment or further refinement.
- Documented training results, including performance metrics and validation reports.
- Established traceability between trained models, requirements, and architecture

Detailed Design and Unit Construction vs. Machine Learning Training

Detailed design produces software units, while ML training develops and optimizes models from data

1. Scope of Activities

- **Detailed Design:** Specifies the static and dynamic aspects of software components.
- **ML Training:** Focuses on creating, refining, and optimizing data-driven models.
- **Overlap:** Both translate design elements into functional outcomes.

2. Implementation Process

- **Detailed Design:** Involves coding, structuring, and testing software units.
- **ML Training:** Includes preprocessing data, running training algorithms, and tuning hyperparameters.
- **Dependency:** ML training requires a functioning infrastructure built from detailed designs.

3. Validation and Optimization

- **Detailed Design:** Validated through unit tests and integration testing.
- **ML Training:** Validated using metrics like accuracy, recall, and F1-score.
- **Collaboration:** Ensures trained models align with functional and non-functional requirements.

Process ID
MLE.3
Process name
Machine Learning Training
Process purpose
The purpose is to optimize the ML model to meet the defined ML requirements.
Process outcomes
1) an ML training and validation approach is specified 2) the data set for ML training and ML validation is created 3) the ML model, including hyperparameter values, is optimized to meet the defined ML requirements 4) consistency and bidirectional traceability are established between the ML training and validation data set and the ML data requirements 5) results of optimization are summarized, and the trained ML model is agreed and communicated to all affected parties

MLE.3 – Machine Learning Training	Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5
Output Information Items					
08-65 ML training and validation approach	X				
03-51 ML data set		X			
01-53 Trained ML model			X		
01-54 Hyperparameter			X		
13-51 Consistency evidence				X	
13-52 Communication evidence					X
Base Practices					
BP1: Specify ML training and validation approach	X				
BP2: Create ML training and validation data set		X			
BP3: Create and optimize ML model			X		
BP4: Ensure consistency and establish bidirectional traceability				X	
BP5: Summarize and communicate agreed trained ML model					X

Base practices

MLE.3.BP1: Specify ML training and validation approach. Specify an approach which supports the training and validation of the ML model to meet the defined ML requirements. The ML training and validation approach includes

- entry and exit criteria of the training and validation
- approaches for hyperparameter tuning / optimization
- approach for data set creation and modification
- training and validation environment

Note 1: The ML training and validation approach may include random dropout and other robustification methods.

Note 2: ML validation is the optimization of the hyperparameters during Machine Learning Training (MLE.3). The term "validation" has a different meaning than VAL.1.

Note 3: The training environment should reflect the environment of the deployed model.

MLE.3.BP2: Create ML training and validation data set. Select data from the ML data collection provided by SUP.11 and assign them to the data set for training and validation of the ML model according to the specified ML training and validation approach.

Note 4: The ML training and validation data set may include corner cases, unexpected cases, and normal cases depending on the ML requirements.

Note 5: A separated data set for training and validation might not be required in some cases (e.g., k-fold cross validation, no optimization of hyperparameters).

MLE.3.BP3: Create and optimize ML model. Create the ML model according to the ML architecture and train it, using the identified ML training and validation data set according to the ML training and validation approach to meet the defined ML requirements, and training and validation exit criteria.

MLE.3.BP4: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the ML training and validation data set and the ML data requirements.

Note 6: Bidirectional traceability supports consistency and facilitates impact analyses of change requests.

MLE.3.BP5: Summarize and communicate agreed trained ML model. Summarize the results of the optimization and inform all affected parties about the agreed trained ML model.

BP1: Specify ML Training and Validation Approach

Ensures that ML training and validation are systematically planned to achieve performance objectives

MLE.3.BP1: Specify ML training and validation approach. Specify an approach which supports the training and validation of the ML model to meet the defined ML requirements. The ML training and validation approach includes

- entry and exit criteria of the training and validation
- approaches for hyperparameter tuning / optimization
- approach for data set creation and modification
- training and validation environment

Note 1: The ML training and validation approach may include random dropout and other robustification methods.

Note 2: ML validation is the optimization of the hyperparameters during Machine Learning Training (MLE.3). The term "validation" has a different meaning than VAL.1.

Note 3: The training environment should reflect the environment of the deployed model.

1. Define Training Objectives

- Specify desired outcomes such as accuracy, precision, and recall.
- Include constraints like training time and computational resource limits.
- Align objectives with ML requirements and system-level goals.

2. Plan Validation Criteria

- Define metrics for assessing model performance (e.g., F1-score, ROC-AUC).
- Use validation datasets to test model generalization and robustness.
- Ensure criteria cover edge cases and diverse operating scenarios.

3. Select Training and Validation Methods

- Choose appropriate algorithms for training based on data type and objectives.
- Specify cross-validation techniques or test/train splits for evaluation.
- Document methods to ensure reproducibility and consistency.

BP2: Create ML Training and Validation Dataset

Create high-quality training and validation datasets to support ML model development

MLE.3.BP2: Create ML training and validation data set. Select data from the ML data collection provided by SUP.11 and assign them to the data set for training and validation of the ML model according to the specified ML training and validation approach.

Note 4: The ML training and validation data set may include corner cases, unexpected cases, and normal cases depending on the ML requirements.

Note 5: A separated data set for training and validation might not be required in some cases (e.g., k-fold cross validation, no optimization of hyperparameters).

1. Select Data Sources

- Identify relevant datasets from the ML data collection.
- Ensure inclusion of corner cases, unexpected cases, and normal examples.
- Verify data relevance to the ML objectives.

2. Assign and Split Data

- Divide data into training and validation subsets.
- Apply methodologies such as k-fold cross-validation if applicable.
- Maintain consistency in dataset usage across iterations.

3. Prepare and Document Data

- Apply preprocessing techniques (e.g., normalization, cleaning).
- Document data characteristics and preparation steps.
- Ensure transparency and reproducibility of data processing.

BP3: Create and Optimize ML Model

Create and refine the ML model to align with defined objectives

MLE.3.BP3: Create and optimize ML model. Create the ML model according to the ML architecture and train it, using the identified ML training and validation data set according to the ML training and validation approach to meet the defined ML requirements, and training and validation exit criteria.

1. Model Creation

- Develop the ML model based on the architectural design.
- Implement core algorithms and define model parameters.
- Ensure initial functionality aligns with requirements.

2. Hyperparameter Tuning

- Adjust hyperparameters (e.g., learning rate, batch size).
- Use optimization techniques such as grid search or random search.
- Balance between performance and resource efficiency.

3. Training Iterations

- Conduct multiple training iterations to optimize model performance.
- Monitor loss, accuracy, and other performance metrics.
- Implement early stopping to avoid overfitting.

BP4: Ensure Consistency and Traceability

Consistency and traceability links between ML requirements and training-validation processes

MLE.3.BP4: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the ML training and validation data set and the ML data requirements.

Note 6: Bidirectional traceability supports consistency and facilitates impact analyses of change requests.

1. Traceability Mapping

- Define links between ML data requirements and training objectives.
- Create traceability matrices for input and output data flows.
- Support auditability and impact analysis for changes.

2. Consistency Checks

- Verify that training data matches requirements (e.g., data quality, distribution).
- Ensure validation processes align with ML model objectives.
- Address inconsistencies during iterative model refinement.

3. Verification of Training-Validation Coverage

- Ensure adequate data coverage during training and validation phases.
- Validate that all identified ML requirements are addressed.
- Document verification results for compliance purposes.

BP5: Summarize and Communicate Agreed Trained ML Model

Summarizing ML model results and ensuring stakeholder alignment

MLE.3.BP5: Summarize and communicate agreed trained ML model. Summarize the results of the optimization and inform all affected parties about the agreed trained ML model.

1. Summarization of Training Outcomes

- Compile training and validation results, including accuracy metrics.
- Highlight key findings from optimization efforts.
- Ensure transparency in model performance documentation.

2. Stakeholder Communication

- Present results in an understandable format for all parties.
- Share insights on model limitations and areas for future improvement.
- Align stakeholders on model readiness for deployment.

3. Model Agreement and Finalization

- Confirm model's compliance with ML requirements.
- Obtain stakeholder sign-off for deployment or further iteration.
- Archive results for traceability and future reference.

Challenges in MLE.3 - Machine Learning Training

Software detailed design often faces challenges in alignment, modularity, and maintainability

1. Data Quality and Quantity Issues

- Insufficient data for training and validation.
- Presence of noisy or biased data affecting model performance.
- Challenges in creating diverse datasets for corner cases.

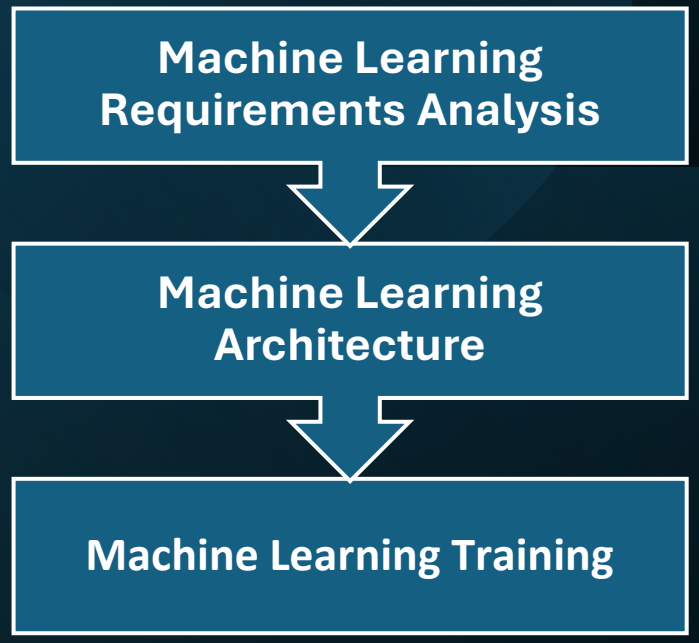
2. Hyperparameter Optimization Complexity

- Determining optimal values for hyperparameters.
- Managing computational resources during optimization.
- Balancing overfitting and underfitting in model training.

3. Stakeholder Collaboration Gaps

- Misalignment on model performance expectations.
- Communication barriers in explaining technical results.
- Challenges in obtaining consensus for model approval.

Summary and Q&A



Software engineering processes ensure structured design, implementation, and traceability for successful project outcomes.

1. MLE.1 - Machine Learning Requirements Analysis

- Focused on deriving and structuring ML-specific requirements.
- Emphasized consistency with software architecture and traceability.
- Addressed challenges in aligning ML data requirements with system constraints.

2. MLE.2 - Machine Learning Architecture

- Defined architectures supporting training and deployment.
- Established traceability and hyperparameter design alignment.
- Managed challenges in trustworthiness, explainability, and scalability.

3. MLE.3 - Machine Learning Training

- Optimized ML models for performance and compliance.
- Ensured alignment between training data, validation processes, and ML requirements.
- Overcame challenges in data variety, robustness, and hyperparameter tuning.