Software Quality Journal (2024) 32:1239–1285
https://doi.org/10.1007/s11219-024-09687-z

RESEARCH

Integrated multi-view modeling for reliable machine learning-intensive software engineering

Jati H. Husen[1,2] · Hironori Washizaki[1] · Jomphon Runpakprakun[1] · Nobukazu Yoshioka[1] · Hnin Thandar Tun[1] · Yoshiaki Fukazawa[1] · Hironori Takeuchi[3]

This paper is an extension of a previously published conference paper Husen et al. (2023). This paper significantly expands the proposed framework in the form of an updated metamodel and process model, on top of new validations in the form of an extensive case study and a controlled experiment.

Accepted: 20 June 2024 / Published online: 3 July 2024

Abstract
Development of machine learning (ML) systems differs from traditional approaches. The probabilistic nature of ML leads to a more experimentative development approach, which often results in a disparity between the quality of ML models with other aspects such as business, safety, and the overall system architecture. Herein the Multi-view Modeling Framework for ML Systems (M^3S) is proposed as a solution to this problem. M^3S provides an analysis framework that integrates different views. It is supported by an integrated metamodel to ensure the connection and consistency between different models. To facilitate the experimentative nature of ML training, M^3S provides an integrated platform between the modeling environment and the ML training pipeline. M^3S is validated through a case study and a controlled experiment. M^3S shows promise, but future research needs to confirm its generality.

Keywords Machine learning · Model-based analysis · Multi-view · Integrated framework

1 Introduction

A machine learning (ML)-based system offers numerous benefits. For example, it provides software solutions to previously impossible functionalities, including autonomous driving, object recognition, and forecasting. Due to the criticality of the results, ML-based software must employ a reliable and trustworthy approach (Khomh et al., 2018; Ozkaya, 2020). However, achieving reliability and trustworthiness remains challenging due to the unique characteristics of the ML-based software components (Martínez-Fernández et al., 2022; Sculley et al., 2015).

Extended author information available on the last page of the article

Developments of ML components, which are more often referred to as ML models, are highly experimental and nondeterministic (Wan et al., 2021). Identifying the ML model version suitable for the predefined quality described in the performance metrics requires experimentation using different parameters and datasets (Vogelsang & Borg, 2019). Then continuous monitoring is necessary after deployment to detect drifts and keep the ML Model relevant to the ever-changing operation domain via retraining (Xiang et al., 2023; Lima et al., 2022). Additionally, the characteristics of ML drive solutions that are uncommon to traditional software engineering. Dataset manipulation such as augmentation, reduction, or rebalancing is necessary to provide quality inputs to the training and testing process (Rahman et al., 2023). Manipulation of the neural network architecture (e.g., deep neural network (DNN) repair) is another example of ML-specific solutions that replace traditional software debugging activities (Sotoudeh et al., 2021).

The decisions made during ML-specific development activities often occur in isolation from the requirements of the rest of the software systems (Nahar et al., 2023). Analysis of the components in ML models considers only the ML perspective and ignores other aspects crucial for reliable and successful software system development, such as business and safety requirements (Wolf & Paine, 2020; Pereira & Thomas, 2020). Consequently, the reliability of the overall ML-based system cannot be demonstrated. This type of problem is not new for traditional software. The model-based approach overcomes this problem by facilitating traceability and consistency between different aspects of the software system (Batot et al., 2021). The term multi-view emphasizes the usage of different models to capture various aspects of the system (Reineke et al., 2019).

Several multi-view approaches have been proposed to facilitate ML characteristics (Villamizar et al., 2022; Nalchigar et al., 2021). However, these approaches are prone to inconsistency and a lack of traceability between the analysis and implementation because the decision is implemented separately from the analysis. Some works have aimed to implement a model transformation approach that integrates analysis models into ML model training, but the capability to discuss the relationship between the model defining the ML training design and other aspects necessary for a system-level analysis is lacking (Moin et al., 2022; Koseler et al., 2019). A different approach that combines the benefits of multi-view modeling and integrated implementation is necessary to achieve effective ML system analysis.

This paper extends our proposal of the Multi-view Modeling Framework for ML Systems (M^3S) as a model-based framework that facilitates consistent and comprehensive analysis of ML systems (Husen et al., 2023). M^3S analyzes the ML components and the overall system itself. The analysis includes integrating the modeling environment and the ML pipelines to facilitate the highly experimental characteristics of ML models, in which a series of training and evaluations are conducted with different solutions and configurations to identify an ML model version that satisfies all requirements. These integrations are based on a cohesive metamodel to ensure analysis consistency. The underlying approach of M^3S supports a reliable and comprehensive analysis of the ML system while ensuring tight synchronization with the implementation. This synchronization provides the base for a feedback loop between the analysis and the implementation at both the ML model and the overall system levels.

Previously, we proposed a version of M^3S and evaluated it with a limited case study (Husen et al., 2023). The early version features an analysis framework without any inclusion of integrated implementation of the decision documented in the models. This paper extends the previous work with an improved version of the framework. The enhanced framework includes the integration of the modeling environment with ML training pipelines along with an updated metamodel and process to reflect the improvement. Finally, a

more comprehensive case study and a controlled experiment validate the capabilities of the updated M^3S.

The contributions of this paper include:

- **Proposal of a multi-view modeling approach for an overall analysis of ML systems.** Because comprehensive analysis is necessary to achieve reliable ML systems as an overall solution and not simply an ML model, we validate the usefulness of the proposed approach.
- **Development of an integrated metamodel for ML system analysis.** Integrating different modeling approaches into a single entity is a limitation of the multi-view modeling approach. Herein we create an integrated model, which combines the unique characteristics of ML systems, and introduce an integrated metamodel that incorporates analysis approaches developed for ML systems (e.g., ML Canvas and the ML model pipeline) itself.
- **Integrated tool between the modeling environment and ML training pipeline.** ML model development is a highly experimentative approach. Separating analysis from the ML model training and testing causes drift. To enhance the consistency between these two sides with different natures, we developed a tool that integrates the modeling environment and the ML pipeline through the definition and execution of configurations in ML training approaches.

The rest of this paper is structured as follows. Section 2 provides a motivating example. Section 3 summarizes related works. Section 4 presents the M^3S and its implementation as an integrated tool. Sections 5 and 6 show a case study and experiment of M^3S, respectively. Section 7 discusses the benefits and limitations of M^3S identified during the evaluation. Finally, section 8 concludes this paper with the future direction of M^3S.

2 Motivating example

Figure 1 outlines the motivating example of our work. In a model-first ML system development, an ML model is developed using the standard experimental process where an experiment management tool executes several runs. This generates a set of versions for a multi-class classification model. The performance of each version is tested and is ready for use as the basis to decide the version to be deployed into the system.

On the other hand, the system where the ML model will work has its requirements. At the very least, a business-level decision will dictate the success criteria of the overall project. The problem happens in the link between the performance of the selected version of the ML model and requirements at a higher level. How to trace the order of the importance of the ML performance for each class into the business decisions? On top of that, do the business decisions align or contradict each other on the ML performance level? How to reconcile and maximize the achievement of the project goals in the case of conflicting decisions?

However, the system where the ML model will be deployed has its own requirements. At a minimum, a business-level decision will dictate the success criteria of the overall project. This typically creates issues in the link between the performance of the selected version of the ML model and the requirements at a higher level. Not only is tracing the order of the ML performance for each class into the business decisions crucial but the business decisions must align with the ML performance level. Hence, when the ML performance and business decisions conflict, they must be reconciled to achieve the project goals.

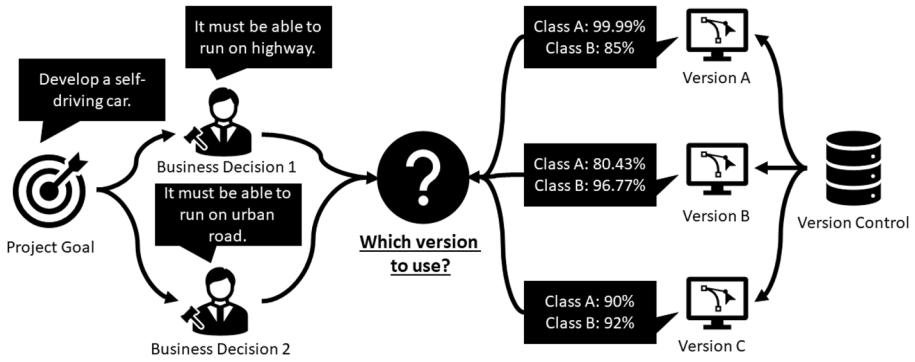


Fig. 1 Motivating example

In a hypothetical situation where training a new version of the ML model is available, another problem also emerges. The approach to train the version must satisfy the business decisions. Analysis based on the defined business decisions must be employed to select a suitable approach. Otherwise, experimentation on training the new version of the ML model may lead to unachievable goals. It should be noted that business decisions may state not only the functionality of the ML model but also other quality aspects such as safety and fairness. Hence, a trade-off may be necessary (Software engineering - systems and software quality requirements and evaluation (square) - quality model for ai systems, 2023).

We aim to solve this problem using M^3S. The multi-view approach should bridge the deterministic side of higher-level requirements and the decisions to implement training strategies inside the black-box ML training activities. An integrated metamodel will guide which part of the decisions are connected and synchronized. Finally, if a change in higher-level requirements is necessary, the traceability of the framework should provide reliable information on the limitations of possible solutions from ML model training.

3 Related works

Several works have investigated reliable ML system development. Here, we classify them based on their similarity to M^3S. In terms of a model-based approach for analyzing ML models, early examples include Bishop's work in probabilistic graphical models (Bishop, 2013) and Infer.net (Minka et al., 2018). Moin et al. proposed the integration of Model-Driven Software Engineering and Automated ML (AutoML) with automated source code and ML model generation (Moin et al., 2022) as well as a Model-driven Engineering (MDE) approach for analytic and software modeling with a focus on ML mainly for the Internet of Things (IoT) domain with a prototype named ML-Quadrat (Moin et al., 2022). Kirchoff et al. conducted a comparative study using MontiAnna, a textual modeling framework, and ML-Quadrat to explain the potential of the MDE approach in the ML domain (Kirchhof et al., 2022). Koseler et al. defined a domain-specific modeling language for ML for a metamodel for ML systems (Koseler et al., 2019). Langford et al. proposed MoDALAS to facilitate model-driven runtime monitoring for learning-enabled components (Langford et al., 2021). M^3S differs from the other studies in this area as it tries to address not only single aspect using a single model but to integrate different aspects into a single workflow.

Some studies have connected ML performance to requirements on higher levels. Villamizar et al. proposed perspective-based ML task specification based on the classification of 45 ML concerns into five perspectives: objectives, user experience, infrastructure, model, and data (Villamizar et al., 2022). Takeuchi and Yamamoto devised an analysis method to construct a business-AI alignment model in ArchiMate (Takeuchi & Yamamoto, 2020). Chuprina et al. proposed an artefact-based requirements engineering approach, which divides the concerns into four layers: context, requirements, system, and data-centric (Chuprina et al., 2021). Nalchigar et al. proposed GR4ML, a conceptual modeling framework for ML that utilizes three perspectives: business, analytic design, and data preparation (Nalchigar et al., 2021). M^{3}S follows the same concept, but a different approach in the abstraction of the levels of requirements. M^{3}S also extends the concept into the ML training pipeline.

Idowu et al. proposed the Experiment Management Meta-Model (EMMM) as an integrated metamodel of a commonly used experiment management tool (Idowu et al., 2022) and taxonomy of those tools (Idowu et al., 2023). The difference between our metamodel and EMMM is the focus of the metamodel. EMMM is an experiment management tool, while our integrated metamodel aims to connect the elements of the analysis models to the artefacts inside the experiment management tool itself. At M^{3}S, the metamodel of the pipeline exists as a single part of the overall integrated metamodel.

The main difference between M^{3}S and the aforementioned studies is the approach utilized for connecting the analysis and implementation of the analysis inside the machine learning training. M^{3}S is not designed to generate the source code for model training. Instead, it connects the solutions, configurations, and results embedded in the ML pipeline with higher-level requirements to support project requirements achievement. Moreover, M^{3}S facilitates a feedback loop between the ML pipeline and the system modeling through the integrated environment for modeling and training, which is not well supported by traditional model-driven approaches of model-to-code generation.

4 Multi-view modeling framework for ML system

Figure 2 overviews M^{3}S, which aims to integrate and synchronize both sides. The key concept behind M^{3}S is an integrated, traceable process between system models and the ML pipeline. It involves a multi-peak process where the analysis and implementation are iteratively refined from a highly abstract and speculative state into a more concrete and proven one. An integrated metamodel guides the connection between system models, ML pipelines, and configurations, connecting both sides to ensure consistency and traceability. There are three use cases of M^{3}S: top-down, model-first, and parallel approaches. The use cases are based on which side is developed first. The use cases are based on common trajectories of ML system development (Nahar et al., 2022).

The first is a top-down approach. This is like a traditional software process where analytical models are developed prior to the execution of ML pipelines. The results of ML testing are then used as the basis to refine the decisions before another ML pipeline is executed. This use case fits a product-first approach to ML-system development.

The second one is the model-first approach. In this case, early versions of the ML model are developed before the analysis starts. The analysis is done in a reverse engineering manner. First, the current capability of the ML model is connected to the higher requirements. Second,

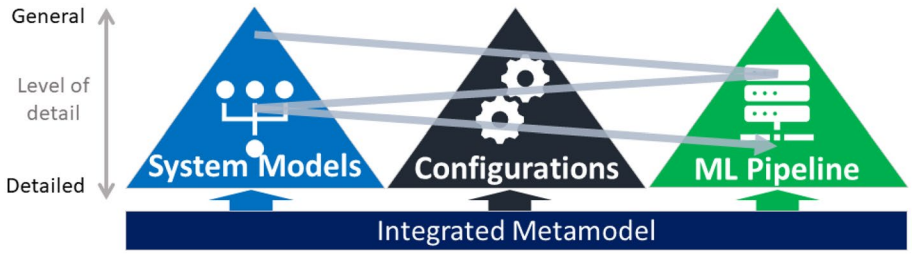


Fig. 2 Overview of M^{3}S

whether the model satisfies the goal is determined. Third, the requirements are retrained or negotiated based on the findings of the analysis. This approach is appropriate when the ML system process is based on exploring what kind of ML task the existing data can produce.

The last one is the parallel or hybrid approach. This use case starts by defining the highest-level requirements followed by small experimentation using anything the team can think of. In this case, both the analysis and experimentation sides must communicate frequently to close the gaps as the analysis and ML performance become clearer over time. This is the best usage of M^{3}S. However, this use case is also the most challenging to implement correctly.

4.1 Multi-view modeling process

M^{3}S is comprised of six views covering different aspects of a reliable ML system. Table 1 summarizes the views and the model responsible for each view. The views are selected based on the identification part ISO/IEC Guide 51, which specifies analysis steps of the safety aspects for standards (Safety aspects - guidelines for their inclusion in standards, 2014). ISO/IEC Guide 51 covers high-level use cases, functionalities, and failure analysis of the system itself. ISO/IEC Guide 51 is selected with the goal of aligning the framework with safety aspects by following the required iterative process for risk assessment. This allows M^{3}S to be utilized for safety-critical ML systems where highly reliable analysis and argumentation of the development of ML model and overall system are required. We transformed the steps into aspects that must be covered and subsequently assigned a model for each step. Figure 3 summarizes the transformation and model assignment.

Table 1 Views of M^{3}S

No	View	Covered Aspect	Responsible Model
1	Value	Business and project requirements	AI Project Canvas
2	ML Task	ML task requirements and higher-level performance requirements	ML Canvas
3	Architecture	Architectural design and integration requirements	Architectural Diagram (SysML)
4	Goal	Detailed ML performance requirements	KAOS Goal Model
5	Safety	Safety requirements	STAMP/STPA
6	Argumentation	Solution argumentation	Safety Case

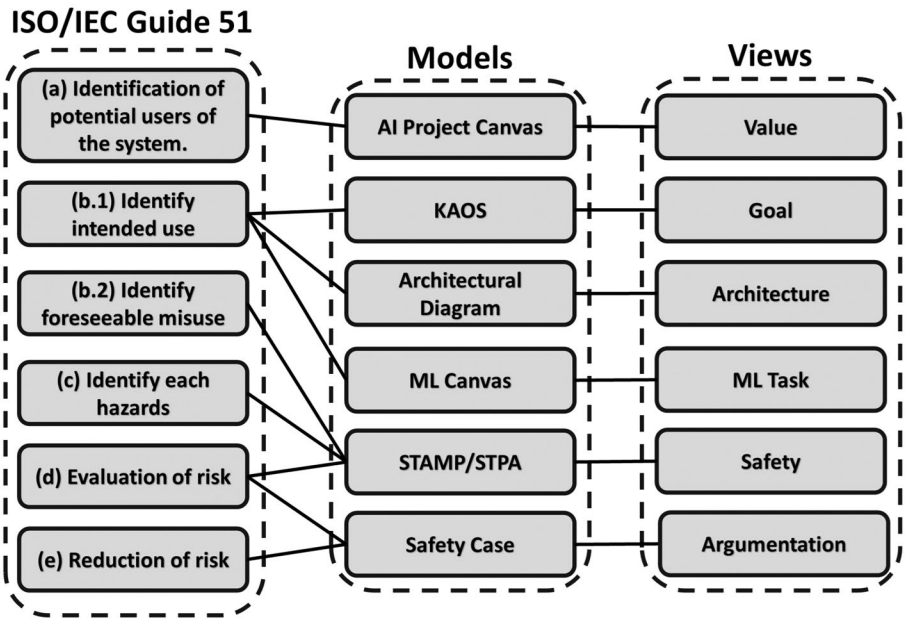


Fig. 3 Mapping of models into views and ISO/IEC Guide 51

Following the steps of ISO/IEC Guide 51, the modeling process becomes systematic steps of modeling activities. Back-and-forth adjustments between models may be necessary to correct and align the information between them. The decisions captured in the models are then implemented into a model training pipeline. In addition, a V-shaped process can be realized by connecting a validation process to each step of the modeling activities (Fig. 4). By utilizing a V-shaped process, the correctness of the decisions made during the analysis can be evaluated and traced for improvement and revision during the development process. Ultimately, a monitoring phase to detect performance degradation of the ML model from drift can be traced back to the main goal of the ML system.

The details of the process are as follows:

1. **Begin with the "Value" view**. This view involves developing an AI Project Canvas to capture the project-level requirements (Thiée, 2021). This step defines the top business-level requirements, which include value propositions, potential users and other stakeholders, related aspects of the systems, and financial aspects of the project.
2. **Identify the common aspects of the ML Task on the "ML Task" view using ML Canvas** (Dorard, 2015). ML Canvas incorporates the information of AI Project Canvas (e.g., the value proposition, output, and data) into the requirements necessary for building an ML pipeline. ML Canvas defines the data collection and processing activities, the desired capability of the ML model, and the need for continuous monitoring.
3. **Develop the "Architecture" view by an architecture diagram made in SysML.** This view overviews the workflow to integrate the ML models and traditional software components. Communications between the components, including sensors, controllers, and user interfaces, are defined here. The information on integration in AI Project Canvas acts as the baseline for developing this model.

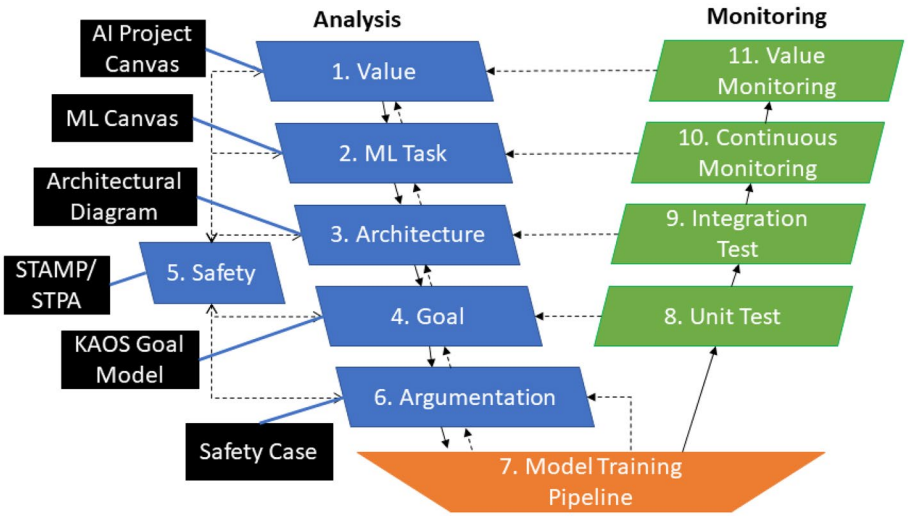


Fig. 4 Modeling Process of M^{3}S

4. **Model the "Goal" view using the KAOS Goal Model.** This view defines the details of the ML task and the expected performance defined in ML Canvas. It also assigns the expected performance to each ML component described in the architectural diagram (Matulevičius & Heymans, 2007). The model decomposes the task and required performance into more detailed specifications. In the leaf nodes of the KAOS diagram, details of the desired performance must be defined in a measurable form. A formal specification of the ML performance can also be used to give an unambiguous specification (Letier, 2001).
5. **Employ STAMP/STPA acts as the model for the "Safety" view in M^{3}S** (Leveson, 2012). The model uses the specified architecture in the architecture diagram to model interactions and identify potential communication failures that may lead to accidents. Root cause analysis of each failure is followed by the definition of the countermeasures that need to be implemented to ensure the safety of the overall system.
6. **Assign responsibility to the safety case as the model of the "Argumentation" view that captures the solutions implemented in the ML pipeline and related components.** The model specifies the goals of each implementation of solutions. The solutions cover data engineering, training approach, safeguarding, and other aspects necessary to be argumentative. The ML pipeline implements the solution and its configuration reflected in the safety case.
7. **Execute the workflow of the training pipeline according to the information in the solutions described in the "Argumentation" view.** Continuous synchronization provides consistency between decisions to utilize the solutions. Then the result of the training pipeline goes through several steps of testing.
8. **Execute unit tests in the form of ML performance metrics to signal whether the specified minimum performances in the "Goal" view are too optimistic or pessimistic.** This step shows the achievement of ML performance requirements and decides whether a change to the analysis model is necessary. New solutions in the "Argumentation" view or evolution of the ML performance requirements in the "Goal" view may be necessary.

9. **Execute an integration test to validate whether the designed architecture in the "Architecture" view fulfills its purpose.** Like the previous step, the architectural decision in the "Architectural" view may need to be updated if the integration fails to be done or does not demonstrate the desired quality.
10. **Implement continuous monitoring to detect possible drift and wrong specifications of the "ML Task" view.** This step is important because the uncertainty of the ML model cannot be removed from the system. Detection through the described Monitoring element in ML canvas notifies developers when drifts occur.
11. **Employ value monitoring to evaluate whether the proposed values in the "Value" view are achieved.** A misaligned business judgment needs to be monitored closely. A change in the environment may lead to an incorrect business judgment, which requires a shift in the value proposition.

4.2 Integrated metamodel

To achieve consistency between different views, we developed an integrated metamodel, which summarizes the relationship between the elements inside each view, using a metadata modeling process. The metamodeling process focuses on identifying similar elements between different models and connecting parts that lead to a comprehensive connection between all views. The integrated metamodel of M^3S not only covers the models but also their communication with the ML pipeline. To achieve this, the integrated metamodel includes common concepts of the ML pipeline and the experiment management tool. Finally, the general concept of an 'ML Solution' is added to describe the implementation of solutions detailed in the safety case. The integrated metamodel is constructed and evaluated iteratively. The process begins with a metamodel for each utilized model. Then, the connections between all element pairs are evaluated to determine the connection type.

Connections are classified into four categories: same, similar, aggregate, or contribution (El Hamlaoui et al., 2018). "Same" connections mean the exact similarity between two elements, and the description of both elements mimics each other. "Similar" connections represent a generality and specialization between two elements, where one element describes a higher-level explanation while the other provides a more specific description. Most connections are classified as "contributions," which means the two elements are interdependent. Finally, "aggregation" connections show one element as a subset of another. To ensure the correctness of the metamodel, an iterative evaluation and correction process is implemented.

Table 2 shows an example of connecting the elements between two views and their connection types. The value proposition of both AI Project Canvas for the "Value" view and ML Canvas for the "ML Task" view is the same, meaning that the description from one side should match the other. Output and Integration of the "Value" view show an example of a similar connection, where the Impact Simulation of the "ML Task" view and Component of the "Architecture" view should use these elements as a basis, respectively. The safety goal of the "Argumentation" view should have a more specialized description than the goals of the "Goal" view as part of their aggregation. For the contribution, the ML performance generated from the ML testing activities should contribute to the achievement of the ML requirements, which is a specialized type of goal in the "Goal" view.

Figure 5 depicts the entire integrated metamodel of M^3S, including the examples shown in Table 2. It should be noted that the class diagram notation is used to model our integrated metamodel. The "same" connections are modeled as a single node of an element, such as the "Value Proposition" elements of the AI Project Canvas for the "Value" view

Table 2 Examples of connecting elements in the integrated metamodel

Source View & Model	Source Element	Destination View & Model	Destination Element	Type of Connection
Value - AI Project Canvas	Value Proposition	ML Task - ML Canvas	Value Proposition	Same
Value - AI Project Canvas	Output	ML Task - ML Canvas	Impact Simulation	Contribution
Value - AI Project Canvas	Integration	Architecture - Architectural Diagram	Component	Contribution
ML Task - ML Canvas	Value Proposition	Goal - KAOS Goal Model	Goal	Similar
Architecture - Architectural Diagram	Component	Safety - STAMP/STPA	Entity	Similar
Argumentation - Safety Case	Safety Goal	Goal - KAOS Goal Model	Goal	Similar
Safety - STAMP/STPA	Contribute	Argumentation - Safety Case	Solution	Contribution
Workflow Pipeline	ML Performance	Goal - KAOS Goal Model	ML Requirements	Contribution
Workflow Pipeline	ML Solution	Argumentation - Safety Case	Solution	Contribution

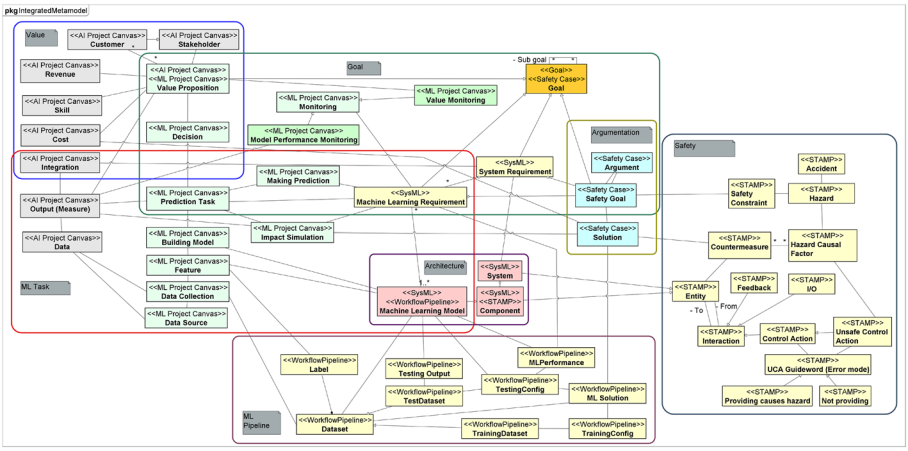


Fig. 5 M^{3}S integrated metamodel

and ML Canvas for the "ML Task" view. Aggregation indicates “aggregate" connections, for example, between "Dataset" and "Label" elements of the ML Pipeline. Generalizations indicate “similar" connections, as shown between the "Goal" element and "Safety Goal." Associations indicate "contribution" connections, including the connection between the "Solution" element of the Safety Case for the "Argumentation" view and the "Countermeasure" of STAMP/STPA for the "Safety" view. A colored box symbolizes that an element may contribute to one or more views.

4.3 Extensibility

M^{3}S was designed to be flexible. Figure 6 summarizes the modification process for the models in M^{3}S. The process of extending the M^{3}S framework is based on the evolutionary thinking approach of ISO/IEC/IEEE 14764:2022 (Iso, iec, ieee international standard - software engineering - software life cycle processes - maintenance, 2022). This extension process allows the adoption of M^{3}S to be lightweight or more extensive, depending on the needs of the case-specific criterion.

The first step in the extension process is to understand the analysis requirements of the ML-system development project. The analysis requirements include the ML concerns relevant to both the processes and products. Information about existing multi-view model-based processes (e.g., the standard goal-oriented multi-view modeling process of M^{3}S) serves as the baseline for extensions into a more fitting process. Model responsibility mapping assigns the analysis models responsible for each aspect described in the analysis requirements. This step specifies the lack or excess of model utilization to cover all necessary views.

Each addition or reduction of the models creates a modification proposal. Each proposal is either accepted or rejected based on the potential impact of the modification. If a proposal is accepted, the integrated model is updated. Finally, continuous monitoring looks for potential concept drifts during the development and system operation. If a major concept drift is detected, another iteration of the extension is conducted to add or remove views.

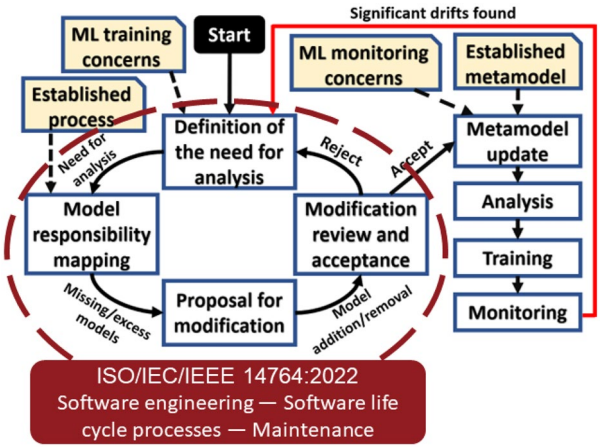


Fig. 6 M[3]S extension process (Husen et al., 2023)

Iterative extensions based on an improved understanding gained from the development and operation should enhance the fitness of the list of utilized models over time.

4.4 Integrated modeling tool

We developed a prototype of an integrated ML system modeling environment to support the modeling and ML training process. The environment is developed as a plugin for the Astah* System Safety[1] because this can reuse existing Astah* System Safety features such as SysML for architecture diagram, Goal Structuring Notation (GSN) for KAOS and safety case, and STAMP/STPA modeling. Additionally, the hyperlink feature of the Astah* System Safety helps connect two different views.

Figure 7 summarizes the extension of the Astah* System Safety to support M[3]S. We created plugins to add ML Canvas and AI Project Canvas to the Astah* System Safety to facilitate all the views. We also extended the SysML requirement object into canvas elements to ensure the elements of ML Canvas and AI Project Canvas can utilize the functionality offered by Astah* System Safety. Figure 8 shows examples of how the canvases work inside the Astah* System Safety environment. The modeling environment is integrated into the Data Version Control[2] (DVC) experiment management tool via a set of plugins. The plugins work as a communication bridge to retrieve and send data to the other side. This integration is developed following the M[3]S metamodel.

For inward communication, we developed the ML performance monitoring plugin. It supports the definition of the ML performance requirements at leaf nodes of the KAOS goal model and fetches the testing result to trace the impact of a version of the ML model in satisfying them. As a proof-of-concept, we monitored the accuracy, precision, recall, and misclassification rate of a classification model. Figure 9 shows the performance requirement setting, ML performance data fetching, and impact tracing feature. The performance requirement setting allows the minimum ML performance to be specified. This minimum can also be set through goals written in parsable descriptions. A pop-up window

[1] https://astah.net/products/astah-system-safety/.
[2] https://dvc.org/.

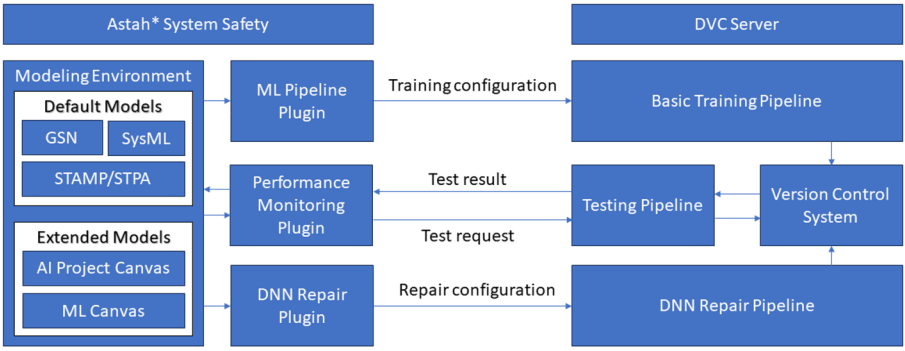


Fig. 7 Architecture of the Astah* System Safety and DVC integration

summarizes all the ML performance requirements and provides a list of versions of the ML model to be evaluated. Finally, the plugin evaluates the achievement of the requirement and traces the failure into all the models.

For outward communication, we developed the DNN repair plugin. This plugin facilitates parameter configuration for the repair process and its execution from the modeling environment. The configuration is specified in the solution element on the "Argumentation" view. The configuration is recorded in the safety case for argumentation purposes. Figure 10 overviews the flow of the configuration setting and repair execution. The solution is defined through a parsable description or GUI support. Another window acts as the execution point to select the version to be repaired and the resulting version's name. The ML pipeline plugin provides a simple function of executing training of the ML model using several hyperparameters as inputs. A message containing the value of each hyperparameter is then sent as a trigger for the new execution of the ML training process. The result of the training process consists of the trained ML model, its

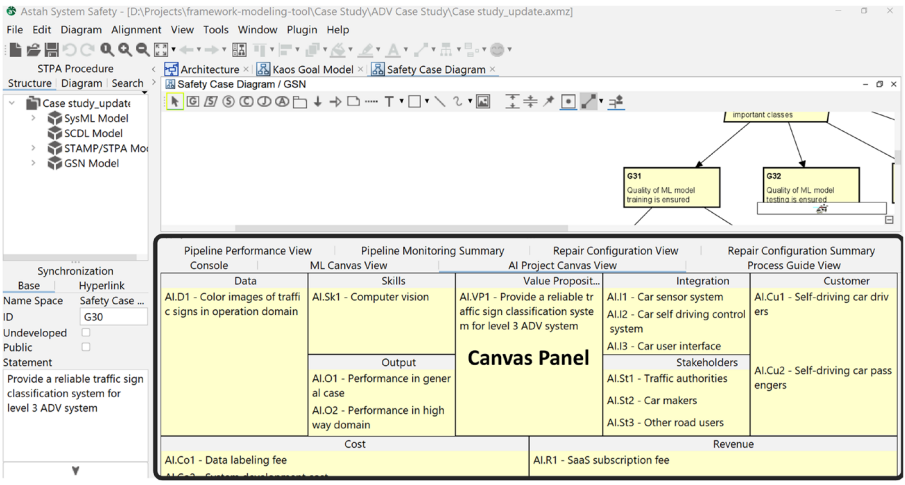


Fig. 8 Implementation of AI Project Canvas on Astah* System Safety

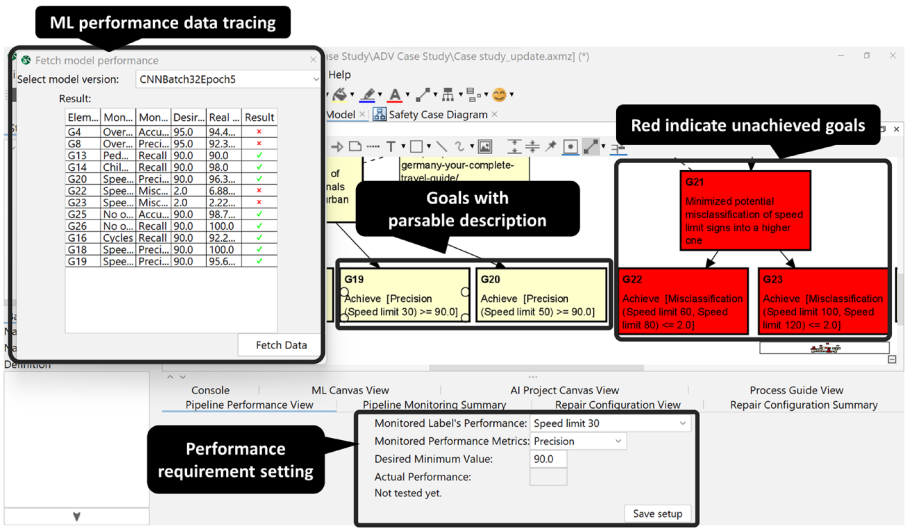


Fig. 9 Performance monitoring feature

metadata, and the validation result. The data is stored in the version control system for future access.

5 Case study

This case study aimed to evaluate the capability of M^{3}S to facilitate comprehensive analysis. We chose a case study as the evaluation method for two reasons. First, it can evaluate the modeling side of M^{3}S from the first step, "Value" view development, to the eighth step,

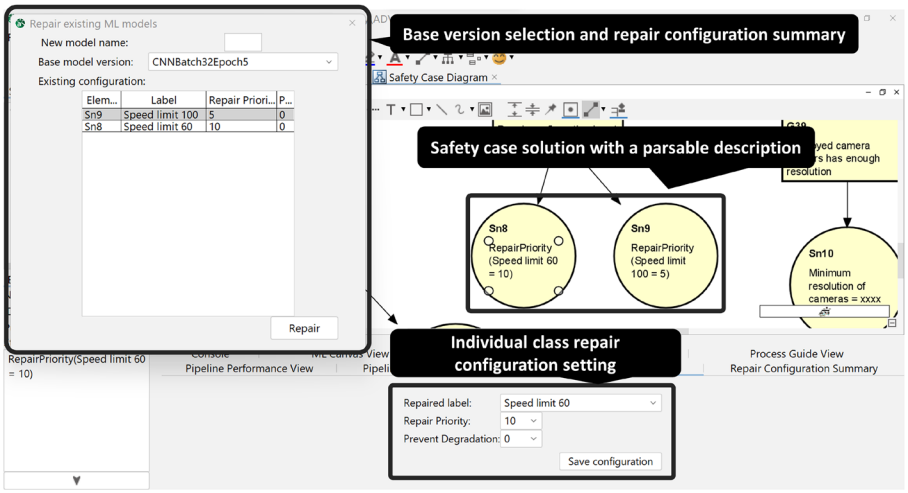


Fig. 10 DNN repair feature

Table 3 Overview of the case study

Aspect	Description
Overall System	Autonomous Driving Car
Mandatory Top Goal	Safe highway operation
Preferable Top Goal	Safe urban road operation
ML Task	Traffic sign multiclass classification
Input	Color images
Output	Classified traffic sign for driving decision-making ML model
Dataset	GTSRB
Available ML techniques	DNN Repair (Athena)

the unit test of trained ML models. Second, it is suitable with respect to the extensive time needed to evaluate the overall process with external participants.

5.1 Utilized case

Table 3 summarizes the case study, which is based on object classification ML models for autonomous driving cars (ADV). Here, the scope of the classification task is limited to traffic sign classification. The inputs for the ML model classification are color images from an embedded camera system as the sensors of the overall ADV system. The classification result is sent to the decision-making ML model as a decision-making input for the car's control system.

ADV is required to work at level three of vehicle autonomy. In level three conditions, the autonomous part returns the driving responsibility to the driver when ADV is operated outside the preferable domain. To train and test the model, we used the publicly available German Traffic Sign Recognition Benchmark (GTSRB) dataset (Stallkamp et al., 2012, 2011). The GTSRB dataset consists of images of German traffic signs that fit the case study. Figure 11 shows samples for each of the 43 traffic sign classes.

The case study considers two operation domains (Fig. 12). The first one is the highway. Because this domain is free of pedestrians and bikes, traffic signs indicating such objects are non-existent. The second one is suburban roads, where pedestrians and bikes are more prevalent. The highway domain is prioritized from an economic standpoint, whereas the suburban road domain is preferable from the user standpoint. The JAMA Framework Japan Automobile Manufacturers Association (2021) and Aurora's safety case framework for ADVs[3] serve as the basis to ensure that the case study reflects the real world as much as possible. To fit the ML model to both cases, DNN repair may be utilized to improve the performance of important classes. However, if no version of trained ML models satisfies both cases, the highway domain is prioritized. The configuration of the repair process must reflect such concerns.

[3] https://blog.aurora.tech/safety/safety-case-framework-development-and-tailoring.

Fig. 11 Sample of images in the GTSRB dataset (Hosseinzadeh Kassani & Teoh, 2016)

Fig. 12 Illustration of the difference between highway and urban road domain

5.2 Research Questions

This case study aims to answer the following research questions (RQs):

- **RQ1. Does the integrated metamodel ensure consistency in the multi-view modeling process of M^{3}S?** RQ1 assesses the benefits and necessity of utilizing an integrated metamodel to facilitate the M^{3}S process. This question should validate the integrated metamodel, which serves as the guideline for the M^{3}S modeling process.
- **RQ2. Does the integrated modeling tool facilitate validating higher-level goals compared to existing ML performances?** RQ2 evaluates the capability of the tool-supported M^{3}S process to maintain and utilize backward traceability between the ML test result and the specified ML performance and other related requirements.
- **RQ3. Does the integrated modeling tool facilitate rationalizing ML-specific solutions and their impact?** RQ3 examines the capability of the tool-supported M^{3}S process to maintain traceability between ML-specific solutions, including the configuration and implementation of such solutions in the ML pipeline.

The process and results of the case study address RQ1. Evaluating the impact analysis function of the integrated tool answers RQ2. Finally, the evaluation of the solution integration function of the DNN repair answers RQ3.

5.3 Results

The case study followed the process steps described in the Subsection 4.1. The numbering in this subsection reflects the numbering of steps of the M^{3}S process.

1. We initially modeled the "Value" view using the AI Project Canvas based on the information from the case study. The result of the modeling can be seen at the top of the Fig. 13.
2. Figure 13 shows the result of the AI Project Canvas for the "Value" view and the ML Canvas for the "ML Task" view along with the related metamodel part that guides the derivation of information from the AI Project Canvas to the ML Canvas. Firstly, the value proposition developed in the "Value" view was copied directly into the "ML Task" view. Then other elements in the "ML Task" view were derived based on the value proposition. On the elements where a specific connection was described in the metamodel (e.g., the outputs of the "Value" view and the impact simulations of the "ML Task" view), more detailed information was derived. At this point, the information necessary for training the initial versions of the ML models was complete. Various versions of ML models using different hyperparameter configurations could be trained using the information of the dataset and ML task in ML Canvas on the DVC side. In parallel, modeling continued for the "Architectural" and "Goal" view.
3. The integration part of the AI Project Canvas dictated the development of an architectural diagram for the "Architectural" view (Fig. 14). Each piece of information in AI Project Canvas's integration was translated into the system requirements prior to dividing into specialized components inside the architectural diagram. The connection between the components was subsequently analyzed and specified to complete the architectural diagram.

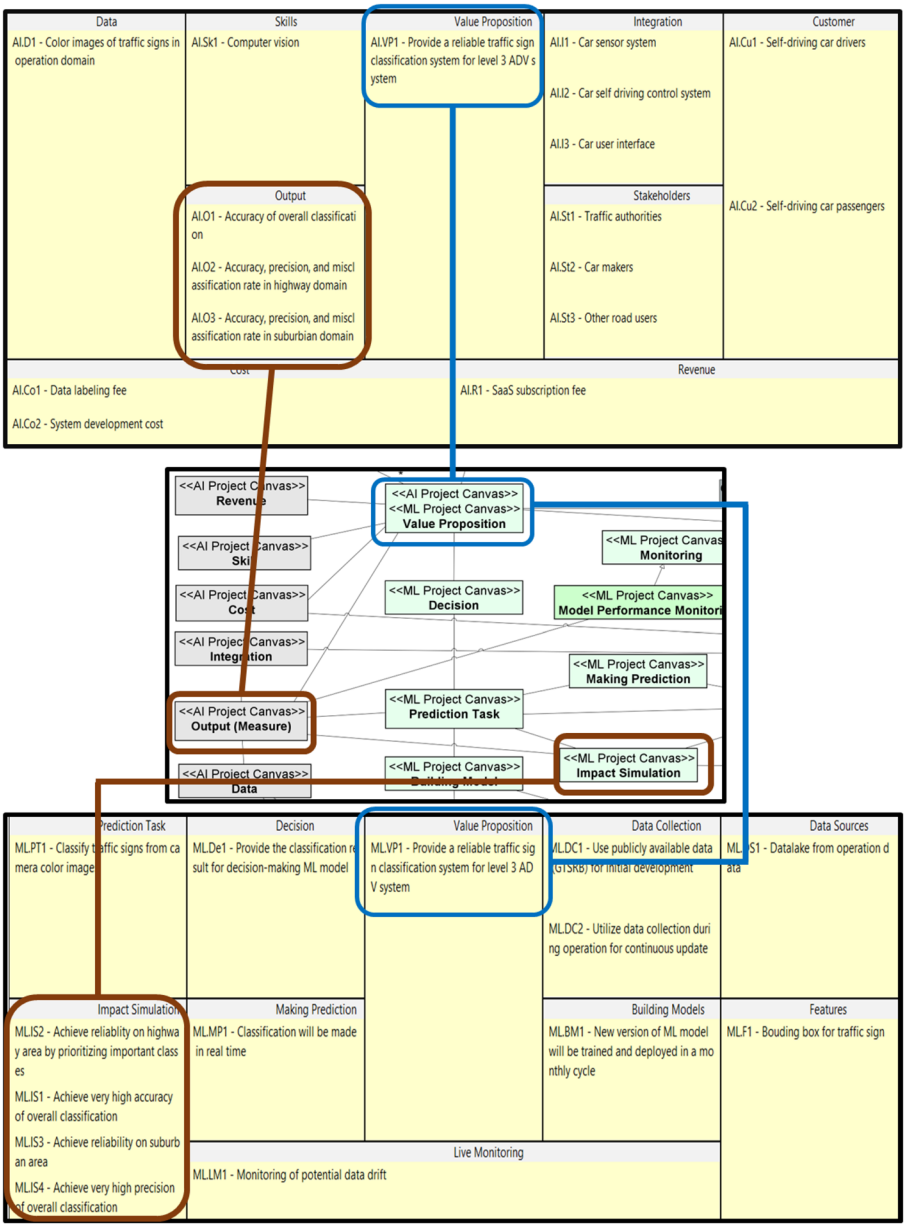


Fig. 13 AI Project Canvas (top) and ML Canvas (bottom) with associated metamodel elements (middle)

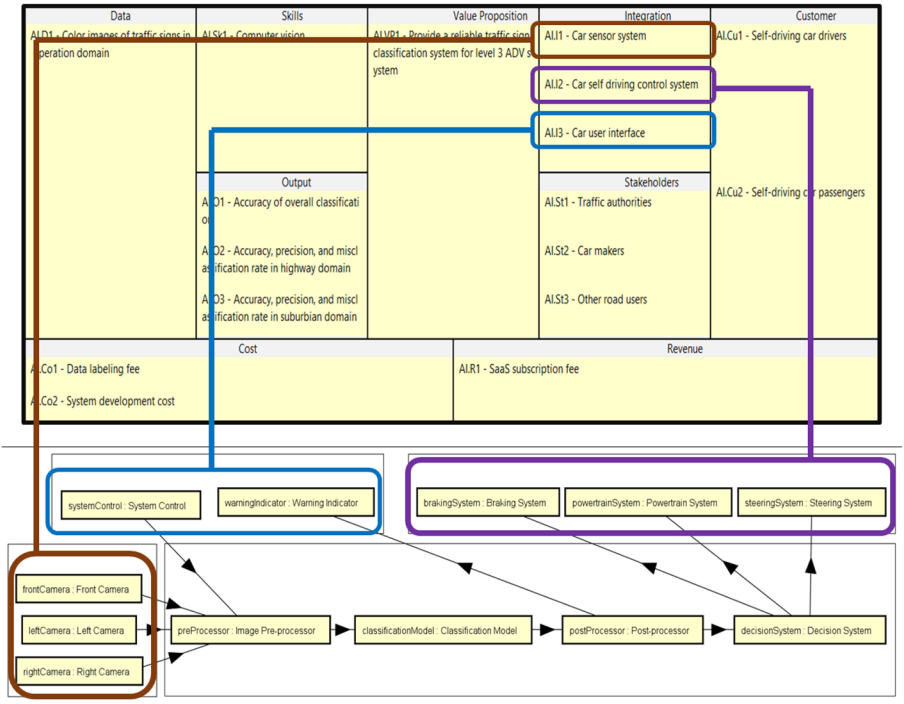


Fig. 14 Derivation of AI Project Canvas' Integration (top) into the Architectural Diagram's components (bottom)

4. The value proposition and impact simulation defined in the "ML Task" served as the basis of higher-level goals in the KAOS goal model for the "Goal" view (Fig. 15). Then it was further decomposed to obtain a goal achievable by a single performance metric. With the required ML performance defined in a semi-formal way, the M^{3}S modeling tool parsed the information and used it to check if the requirements were satisfied. Figure 16 compares the ability of three different ML models to satisfy the requirements. By examining the color-coded elements, version A clearly outperformed the other two. However, no version completely satisfied all the requirements. Thus, we continued developing the "Safety" view with a clear understanding of the limitations of the available ML models.
5. STAMP/STPA inside the "Safety" view indicated how the hazard from the user perspective connects to the limitation of the ML model. The architecture diagram was translated into the control structure diagram of STAMP/STPA (Fig. 17). Unsafe control actions between the ML model and the other components of the ML system were analyzed. Then countermeasures were given based on the limitation of existing ML models towards each hazard causal factor of each unsafe control action.
6. Figure 18 shows the connection of the top goal of the goal model to the top goal of safety constraint and the countermeasures from STAMP/STPA in the safety case of the "Argumentation" view. For the solution that utilized DNN repair, we implemented repair strategies and patterns to improve the best-performing version of the ML model, which was version A. Figure 19 summarizes the patterns utilized and the difference between

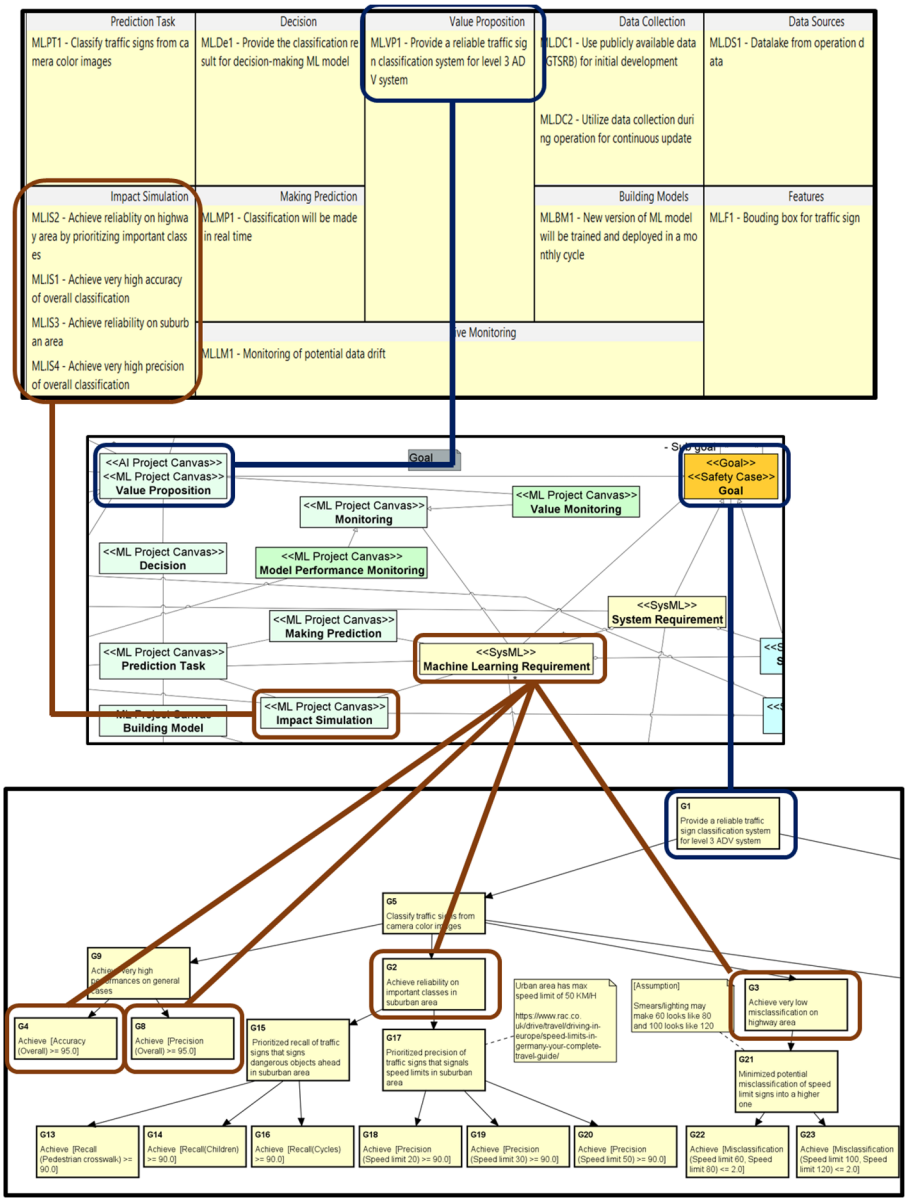


Fig. 15 ML Canvas (top) and part of the KAOS Goal Model (bottom) and its associated metamodel elements (middle)

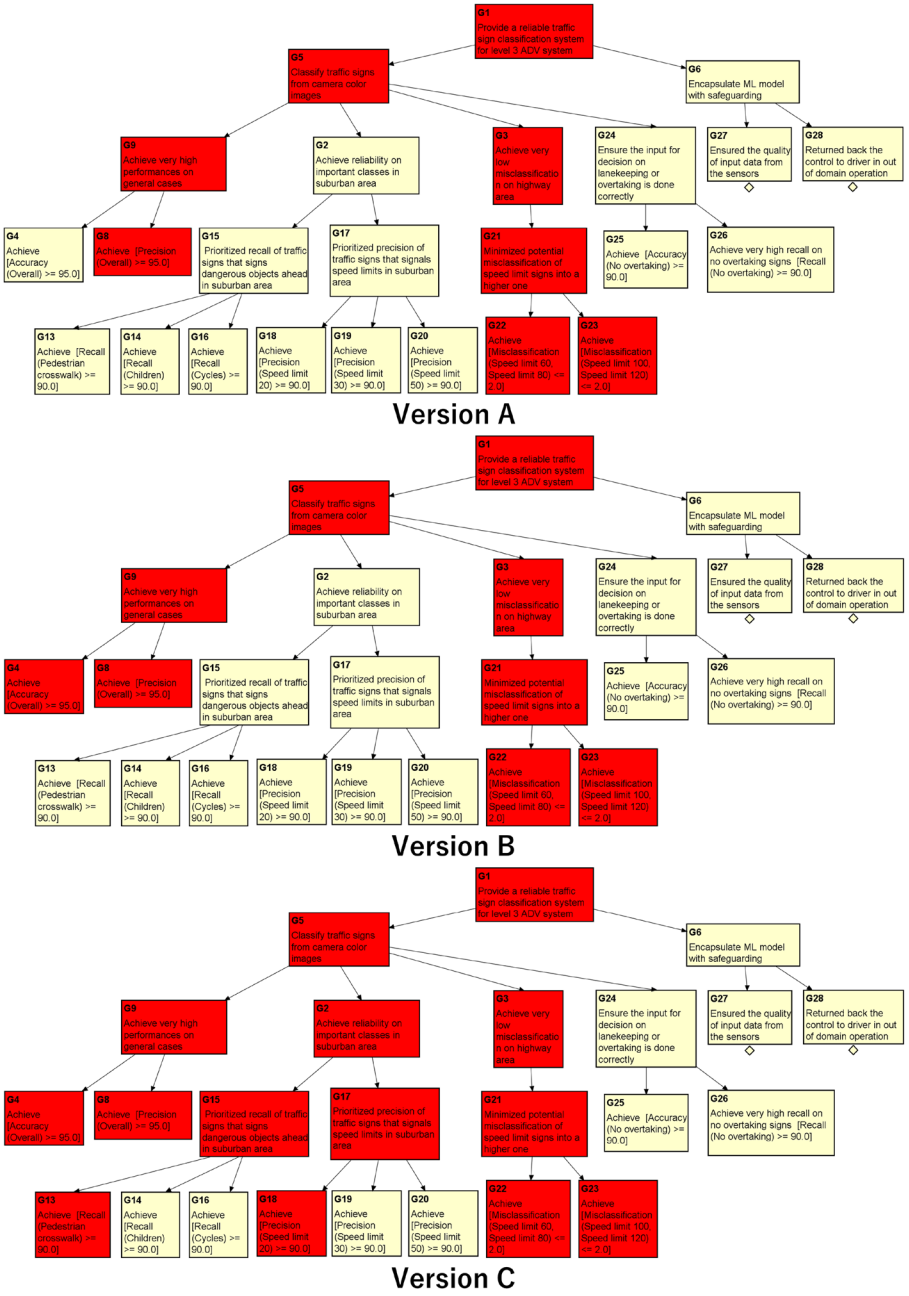


Fig. 16 Comparison of the results from different ML model versions. Red means the performance requirement is not fulfilled

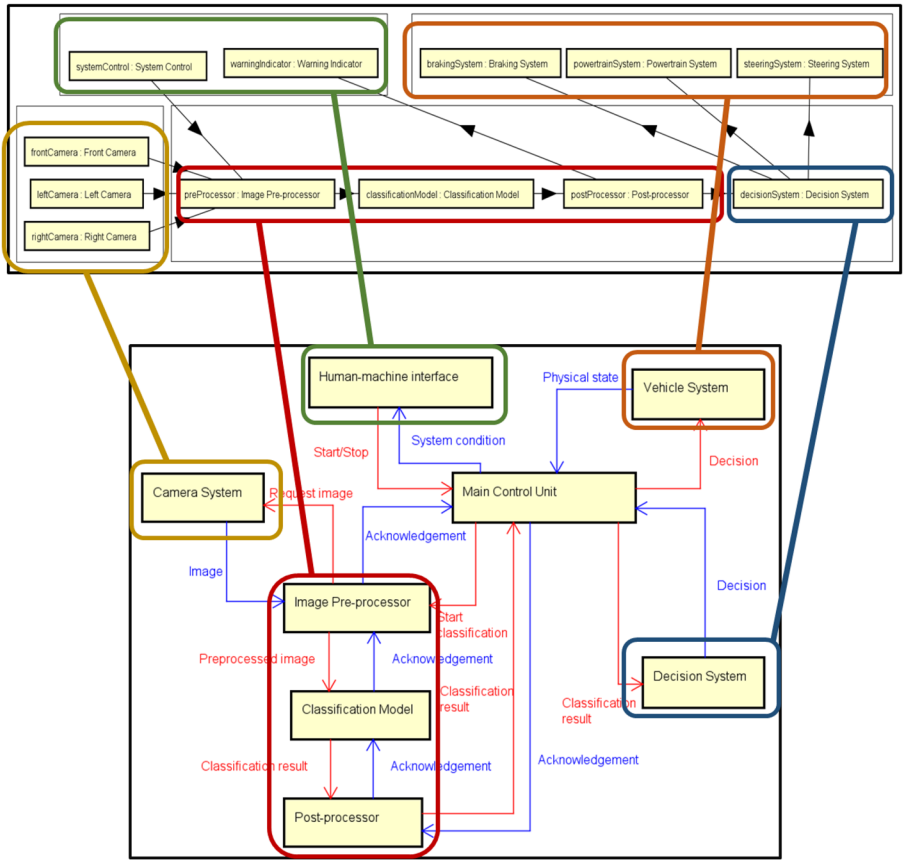
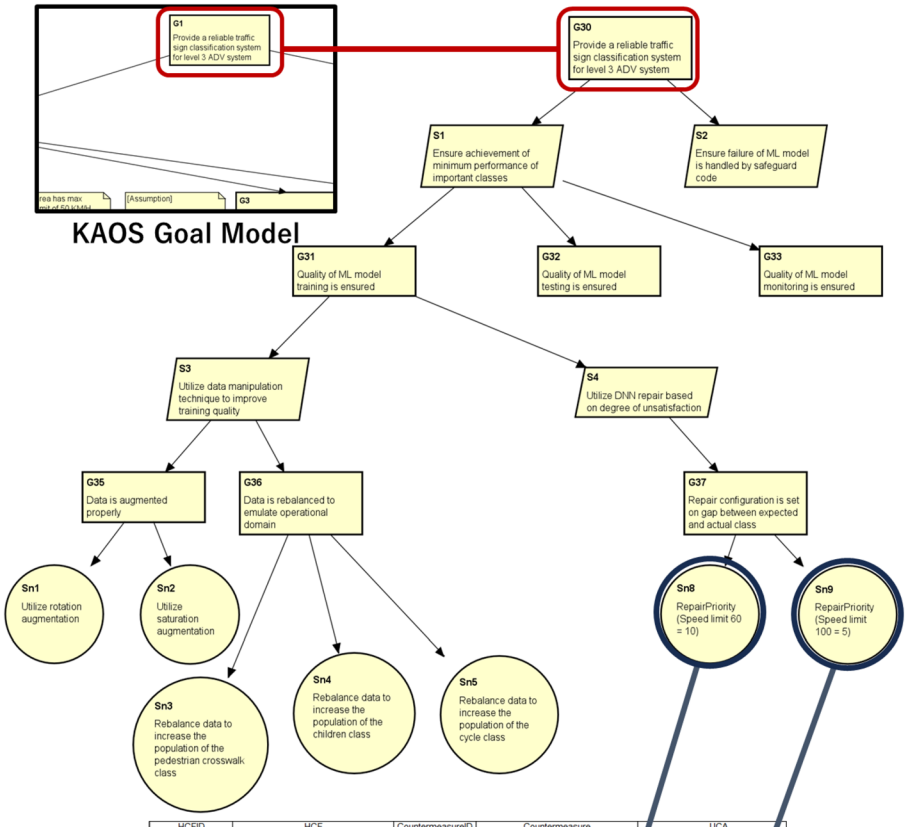


Fig. 17 Derivation of Architectural Diagram's components (top) into STAMP/STPA's Entities (bottom)

the repair results. The first pattern was a balanced approach, where both classes were treated equally with the same priority weighting. The other one prioritized fixing the worse-performing class.

7. DNN repair processes were executed for both cases using the specified configuration in the "Argumentation" view. The repair resulted in two new versions of the ML model, improved from the selected A version as the base model for repair. All new versions of the ML model are stored on the DVC side.
8. Although the execution of the repair improved the performance in both patterns, neither achieved the required ML performance. Further evaluation of the misclassified images showed that the test data quality might not be suitable for real-life situations. The test data in Fig. 20 was too extreme for the target operational domain, considering the development goal only required level 3 self-driving capability. As such, further manipulation of the test data, such as the exclusion of extremely low-quality images, may be necessary to measure the capability of the ML model properly. Moreover, the quality of the sensors also came into question. The ability of the cameras to provide quality images is integral to ensure that the ML model is not exposed to extreme cases. However, both solutions increased development costs. Figure 21 shows how the solutions are reflected in the "Argumentation" view and the associated costs of the solu-



HCFID	HCF	CountermeasureID	Countermeasure	UCA
HCF4-P-1-1	ML model does not have high enough recall for dangerous object traffic signs classes	M1	Utilize rotation augmentation	(UCA4-P-1) Classified model misclassified traffic signs for dangerous object [SC1][SC2][SC3]
HCF4-P-1-1	ML model does not have high enough recall for dangerous object traffic signs classes	M2	Utilize saturation augmentation	(UCA4-P-1) Classified model misclassified traffic signs for dangerous object [SC1][SC2][SC3]
HCF4-P-1-1	ML model does not have high enough recall for dangerous object traffic signs classes	M3	Rebalance data to increase the population of the pedestrian crosswalk class	(UCA4-P-1) Classified model misclassified traffic signs for dangerous object [SC1][SC2][SC3]
HCF4-P-1-1	ML model does not have high enough recall for dangerous object traffic signs classes	M4	Rebalance data to increase the population of the children class	(UCA4-P-1) Classified model misclassified traffic signs for dangerous object [SC1][SC2][SC3]
HCF4-P-1-1	ML model does not have high enough recall for dangerous object traffic signs classes	M5	Rebalance data to increase the population of the cycle class	(UCA4-P-1) Classified model misclassified traffic signs for dangerous object [SC1][SC2][SC3]
HCF4-P-2-1	Misclassification rate of ML model for speed limit signs is too high	M6	Prioritize sign "Speed limit 60 KM/h" during repair	(UCA4-P-2) Classified model misclassified speed limit signs as higher speed limit [SC4]
HCF4-P-2-1	Misclassification rate of ML model for speed limit signs is too high	M7	Prioritize sign "Speed limit 100 KM/h" during repair	(UCA4-P-2) Classified model misclassified speed limit signs as higher speed limit [SC4]

Fig. 18 Development of a safety case (top) from KAOS Goal Model's Top Goal (snippet) and STAMP/STPA's Countermeasures (bottom)

tions in the AI Project Canvas for the "Value" view due to the part of the metamodel that specifies the interconnected changes. Hereafter, we adopted the balanced repair version of the ML model for further integration. However, the need for better-quality camera sensors is noted as a future improvement.

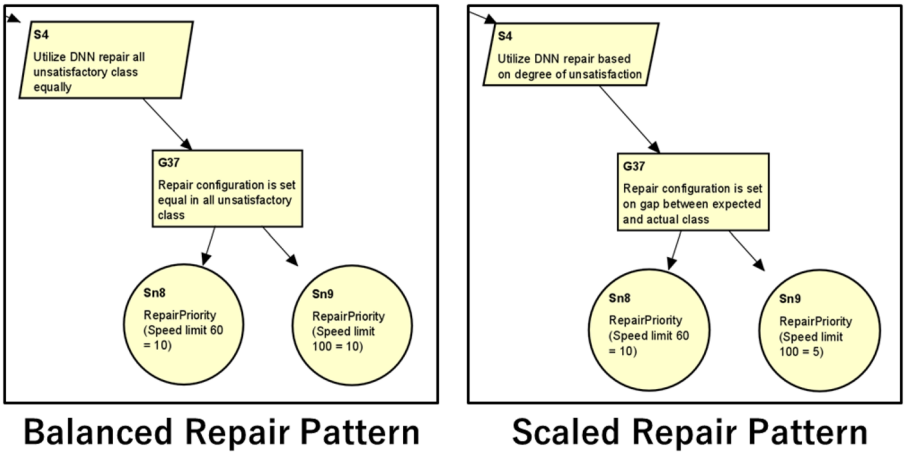


Fig. 19 Repair strategy patterns utilized in the case study

5.4 Answers to research questions

Here, the research questions are answered. Each subsection is dedicated to one research question.

5.4.1 RQ1. Does the integrated metamodel ensure consistency in the multi-view modeling process of M^3S?

The results highlight how the metamodel guides the development of new elements. The metamodel guides the consistency from business-level decisions into the ML training aspects (Fig. 13). Then the information in the AI Project Canvas serves as the basis to

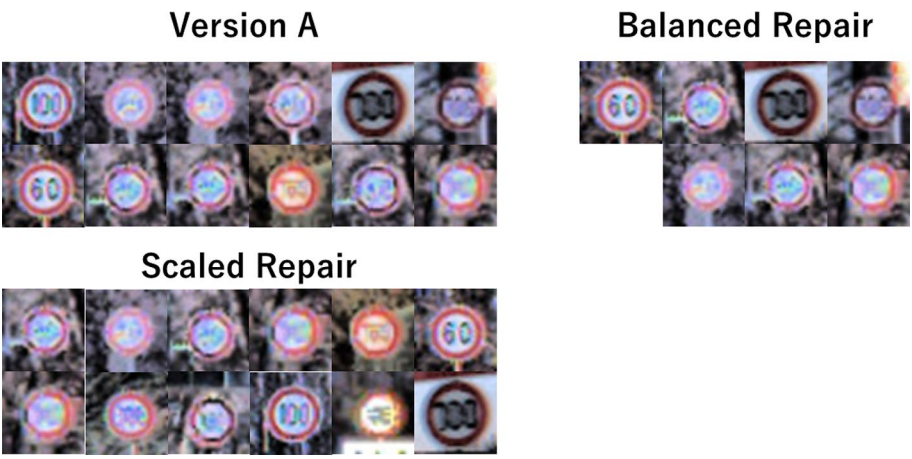


Fig. 20 Misclassified test data for each version of the ML model

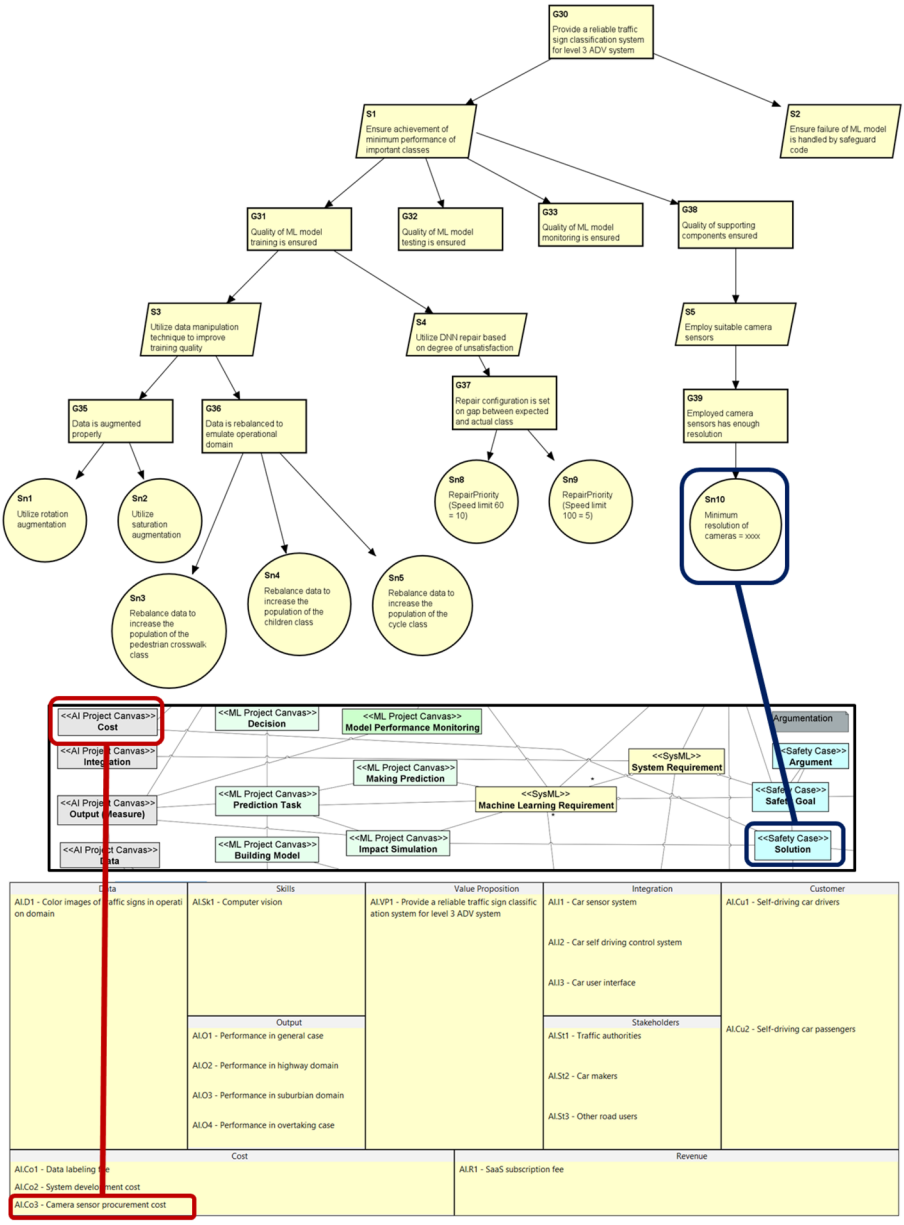


Fig. 21 Inclusion of newly found solutions in the safety case (top) and the associated update on AI Project Canvas (bottom) based on the metamodel (middle)

construct the architectural diagram (Fig. 14) while composing the abstract description in the ML canvas realizes achievable ML performance requirements (Fig. 15). Finally, the architectural decision should be the foundation of the STAMP/STPA control structure (Fig. 17).

The metamodel guides the process in higher-level decisions to update the models during the loop. The costs in the AI project canvas should be added for each newly proposed decision (Fig. 21) because the metamodel must not only work in the initial development but also in the later stages when decisions may need to be changed.

This case study demonstrates the capability of the metamodel. Hence, the integrated metamodel of M^3S can ensure consistency in the multi-view modeling process.

> The integrated metamodel of M^3S ensures the consistency of the multi-view modeling process. Elements of different models can be traced and connected using the integrated metamodel not only during the initial development but also as the analysis models evolve as new solutions update the elements of another view.

5.4.2 RQ2. Does the integrated modeling tool facilitate validating higher-level goals compared to existing ML performances?

The integrated modeling tool can be configured to monitor specific ML metrics through the goals of the KAOS goal model (Fig. 15). The color coding in Fig. 16 demonstrates the tool's capability to communicate with the ML pipeline to fetch the test result and mark the achievement of the configured goal nodes and the associated elements of other models. This format visualizes the impact of the ML model on the achievement of the higher-level goals. The integrated modeling tool utilizes visualization techniques to support the traceability between models. Consequently, the integrated modeling tool can validate the achievement of higher-level goals from existing ML performances.

> The integrated modeling tool of M^3S successfully validates the achievement of higher-level goals using the traceability between the business and system-level goals and lower-lever ML performance goals. The fetched ML performance from the DVC server can be automatically traced to other views such as the "Value" and "Architecture" views.

5.4.3 RQ3. Does the integrated modeling tool facilitate rationalizing ML-specific solutions and their impact?

Figure 18 demonstrate how the information of ML-specific solutions is captured inside the safety case of M^3S. The solution spans from the data layer to ML training and architectural decisions made. The metamodel supported the addition of solutions on other aspects. Moreover, the integrated DNN repair tool, which is used as an implementation example of an integrated solution, shows promising results. Solutions can be captured inside the analysis model and subsequently executed in a single flow, ensuring synchronization of decision-making and solution execution during the iterative addition of solutions. The impact of the solution on ML performance and the higher-level goals can also be traced using the performance checking function (Fig. 16).

It should be noted that these findings are based on a single integrated solution. More implementations of integrated solutions, especially on different aspects such as data manipulation, are necessary to understand the full capability and limitation of integrating

solutions into both the analysis and model training because the solutions may work differently from the DNN repair, which works directly with the ML model.

> The integrated solutions of M^{3}S allow decision-making and execution of solutions to maintain consistency during the iterative and experimentative development process. However, this finding is limited to the context of DNN repair. Further implementation and evaluation of different types of solutions such as data manipulation are necessary to understand the benefits and limitations of integrated ML solutions.

5.5 Threat to validity

The main threat to our case study is the validity of the case itself. An unrealistic case study questions the uncertainty of whether the results of the case study reflect the real situation. We implemented two strategies to ensure that the case study is suitable to evaluate M^{3}S. First, the case study is based on reliable documents. We followed the JAMA framework for required capability and possible failures. We also followed Aurora's safety case framework for ADVs to design the overall system. Second, we solicited input from industry practitioners. We continuously reviewed the case study and the results with industrial experts to verify the quality of the case and analysis results.

6 Controlled experiment

The controlled experiment focused on evaluating the usability of M^{3}S to execute the integrated pipeline. The experiment consisted of two parts. The first one evaluated the capability of M^{3}S to facilitate impact analysis of the integrated pipeline. The second assessed the capability of M^{3}S to execute an integrated solution, which in this experiment is represented in the form of a DNN repair.

6.1 Research questions

We aimed to answer the following research questions (RQs):

- **RQ4. Can M^{3}S efficiently facilitate the impact analysis from ML model performance?** RQ4 evaluates the time needed for M^{3}S to complete the impact analysis task on ML performance testing. A lower time compared to the control group indicates a better efficiency compared to ad-hoc approaches.
- **RQ5. Can M^{3}S efficiently facilitate the analysis of parameters for repair activities?** RQ5 evaluates the time needed for M^{3}S to fully configure the integrated solutions. A lower time compared to the control group indicates a better efficiency compared to ad-hoc approaches.
- **RQ6. Does M^{3}S help users train a better ML model through integrated solutions?** RQ6 evaluates the capability of M^{3}S to help developers effectively incorporate solutions to train better ML models.

- **RQ7. How confident are users about the impact analysis result of M^{3}S?** RQ7 evaluates the users' acceptance toward the result generated by M^{3}S in analyzing the impact of the result of ML performance testing.
- **RQ8. How confident are users about the usability of the impact analysis of M^{3}S?** RQ8 evaluates users' acceptance toward the support provided by the support tool for M^{3}S in analyzing the impact of ML performance testing.
- **RQ9. How confident are users about the result of solution integration in M^{3}S?** RQ9 evaluates users' acceptance toward the result generated by M^{3}S in the configuration and execution of the integrated solution.
- **RQ10. How confident are users about the usability of the solution integration in M^{3}S?** RQ10 evaluates users' acceptance toward the support provided by the support tool for M^{3}S for the configuration and execution of the integrated solution.

The time spent by the participants finishing their tasks answers RQ4 and RQ5. The performance of the ML models repaired by the participants answered RQ6. RQ7, RQ8, RQ9, and RQ10 are addressed based on the participant's responses to the post-experiment questionnaire.

6.2 Experiment design

The experiment is designed to answer the research questions. Here, the design of the flow, participants, and data collection method for the experiment are detailed.

6.2.1 Experiment flow

Our experiment consists of two parts. The first part evaluates the effectiveness and efficiency of the M^{3}S modeling-training pipeline integration for performance monitoring and tracing. The second part evaluates the effectiveness and efficiency of M^{3}S modeling-DNN repair pipeline integration as a sample of modeling-solution integration.

The final goal of the experiment involving the participants is to identify a version of the ML model that best meets the requirements. The model can be an existing or repaired model. If no version satisfies all the ML requirements, they must indicate the changes necessary for immediate deployment of the most suitable model. Additionally, the participants need to finish the tasks given by the proctors to achieve such goals in the available time for each part.

We separated the participants into two: the framework and control groups. The framework group used M^{3}S to achieve their goal, while the control group used an ad-hoc approach. The experiment employed natural language specification as a control for comparison to the M^{3}S multi-view models. To ensure the similarity, the natural language was translated from the M^{3}S models already specified for the experiment group. The framework group used the standard command-line interface (CLI) execution of the pipeline works for the integrated solution.

Figure 22 outlines the ML performance evaluation part of the experiment. Both groups started the experiment with a general briefing. The briefing consisted of an introduction of the group members and an explanation of the goals and tasks. Then the groups were physically separated to work with their own approaches. Both groups started by understanding the requirements for their respective group (e.g., the multi-view models for the framework group or in a natural language for the control group).

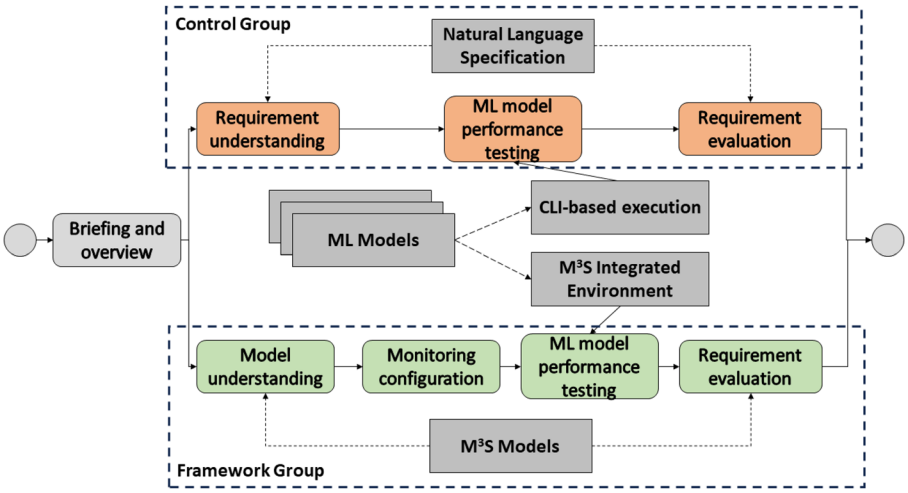


Fig. 22 Experiment flow for ML performance monitoring part

The next step was to execute a tool to work with the ML model. The framework group began by configuring the required performance for the leaf goals they wanted to monitor. Then they executed the fetching of ML model performance from the pipeline. In contrast, the control group executed the pipeline using the command lines in the CLI to determine the performance of each version of the ML model. Finally, each group decided which version of the ML model was the most suitable to satisfy the existing requirements, and if the existing versions of the ML model did not satisfy all requirements, they indicated which requirements were not satisfied.

Figure 23 overviews the second part of the experiment. It began with an explanation of how DNN repair works and what parameters need to be configured for the repair to work. Both groups were then tasked with repairing the version of the ML model they found most suitable in the first part of the experiment to fulfill as many requirements as possible. The experiment group worked with the safety case model to specify the

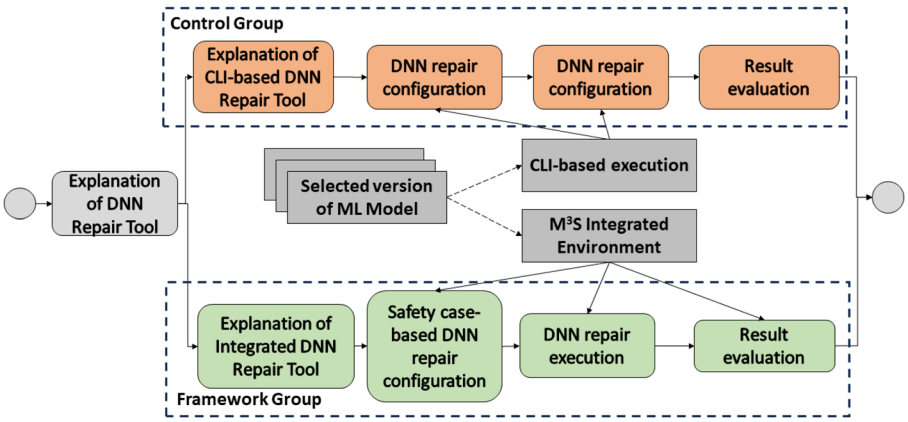


Fig. 23 Experiment flow for the integrated DNN repair pipeline part

configuration, while the control group worked with the configuration file directly to set up the repair process. Both groups then executed the repair using their approach. Then they evaluated the success of the repair process using a similar approach. Finally, they decided whether to use the repaired or the original version based on the suitability to satisfy the requirements. If they chose the repaired version and it did not satisfy all requirements, they had to mark which requirements were not satisfied.

Afterward, the participants completed a post-experiment questionnaire. The experiment concluded with a short discussion session between both groups. During the discussion, a moderator captured the subjective opinions of the participants about what worked well and what could be improved.

6.2.2 Participants

Our experiment attracted thirteen participants from practitioners, academia, and graduate students with varying roles and experience levels. Table 4 summarizes the participants. For the practitioners and academia, we collected the experience based on how long they have worked in their role, whereas the year in graduate school was collected for studies. The personal identities of each participant were obscured to protect their privacy. The participants were divided into four groups: the practitioner control group, the practitioner framework group, the student control group, and the student framework group.

We split the participants into practitioners and students before assigning them to the framework and control groups for two reasons. The first one was to isolate the experience levels to detect differences between the perspectives based on experience. The second one was for flexibility as students had more time available to test the tools. Although the experience and backgrounds were balanced between the framework and control groups, the participants were randomly assigned to a group. For example, two participants had 30 years of experience; one was assigned to each practitioner group, but it was random which one was in each group. Similarly, two students had limited industrial experience and were assigned to different student groups.

Table 4 Summary of the participants. The groups include practitioner control (C-P), practitioner framework (FW-P), student control (C-S), and student framework (FW-S)

No	ID	Background	Role	Experience (Years)	Group
1	P1	Academia	Project Manager	5	C-P
2	P2	Academia	Project Manager	1	C-P
3	P3	Industry	Quality Assurance	5	C-P
4	P4	Academia	Quality Assurance	5	FW-P
5	P5	Industry	Quality Assurance, Software Engineer	5	FW-P
6	P6	Industry	Project Manager	30	C-P
7	P7	Industry	Data Scientist	30	FW-P
8	P8	Academia	Server Reliability Engineer	5	FW-P
9	S1	Student	Master Student	1	C-S
10	S2	Student	Master Student, Software Engineer	1	C-S
11	S3	Student	Master Student	1	FW-S
12	S4	Student	Master Student	2	FW-S
13	S5	Student	Master Student, Project Manager	1	FW-S

Table 5 Post-experiment four-scale Likert questionnaire

No.	Question	Related RQ
Q1	I'm confident in the impact analysis results produced by our team.	RQ7
Q2	I can easily evaluate the performance of different versions of the ML model.	RQ8
Q3	I can easily evaluate the effect of different versions of the ML model in satisfying the ML performance requirements.	RQ8
Q4	I can easily evaluate the effect of different versions of the ML model in satisfying the overall requirements.	RQ8
Q5	I'm confident in the configuration of the repair tool generated by our team.	RQ9
Q6	I can easily decide the value of the configuration for different labels of repair tools.	RQ10
Q7	I can easily evaluate the improvement and regression of the overall requirements satisfaction due to the repair activity.	RQ10

6.2.3 Questionnaire

A post-experiment questionnaire was employed to capture the participants' subjective opinions from both groups. The questionnaire employed a Likert scale to capture the participants' impression (Likert, 1932-1985). Each question was designed to answer RQ3, RQ4, RQ5, and RQ7. A four-point Likert scale was used to reduce bias from selecting the neutral option. Table 5 shows the questions of the post-experiment questionnaire and their corresponding RQs.

The results of the questionnaire from the control and experiment groups were compared. The difference in the average between the groups was used to answer the related RQ. The analysis used a weighted calculation to separate the extreme options of 'Highly Disagree' and 'Highly Agree' from the 'Disagree' and 'Agree' options.

6.3 Results

6.3.1 Time for completion

Figure 24 summarizes the time required for each group to finish the ML performance monitoring part. The student framework group performed three iterations of monitoring during the experiment, with each iteration fulfilling the completion criteria. To evaluate the time properly, we divided their time into three parts to reflect each iteration. We also separated the time for discussion and tool operation. The discussion consisted of conversations about the tool, case, and solution. However, further classification was difficult since the topics were often mixed.

The framework groups tended to use time for tool operation more efficiently than the control groups. The practitioner and student control groups required more than 10 minutes and 13 minutes, respectively, whereas the practitioner and student framework groups each required about 3 minutes. Even when considering the multiple iterations executed by the student framework group, the time remained consistent. Similarly, the discussion time was longer in the framework group when the results were compared by experience level (practitioners or students). For a given experience level, the discussion time of the framework group was almost twice that of the control group.

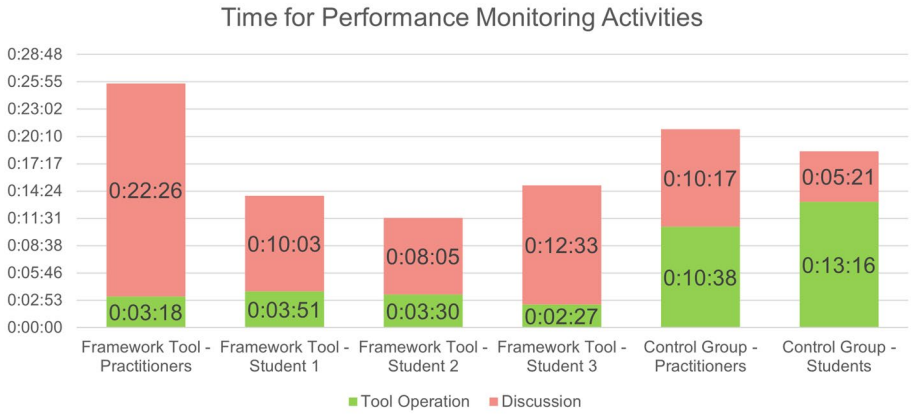


Fig. 24 Time needed for each group to finish the ML performance monitoring task

Figure 25 summarizes the time required for each group to finish the DNN repair part. Similar to the ML performance monitoring part, the student framework group completed three iterations of the task. The practitioner framework group completed two iterations. Following the same approach as the ML performance monitoring part, we divided their time to reflect the iterations. We also separated the tool operation and discussion time.

The control group for a given experience level spent more time on tool operation. The student groups showed a significant difference; the discussion time of the control group was almost twice that of the framework group. For the practitioners, the control group took slightly longer than the framework group. The second iteration from the practitioner framework group was less than half the time for the single iteration of the practitioner control group. However, the learning curve effect must be considered in this comparison. The discussion time for the control groups from both sections was significantly higher than that for the framework groups. Overall, regardless of their

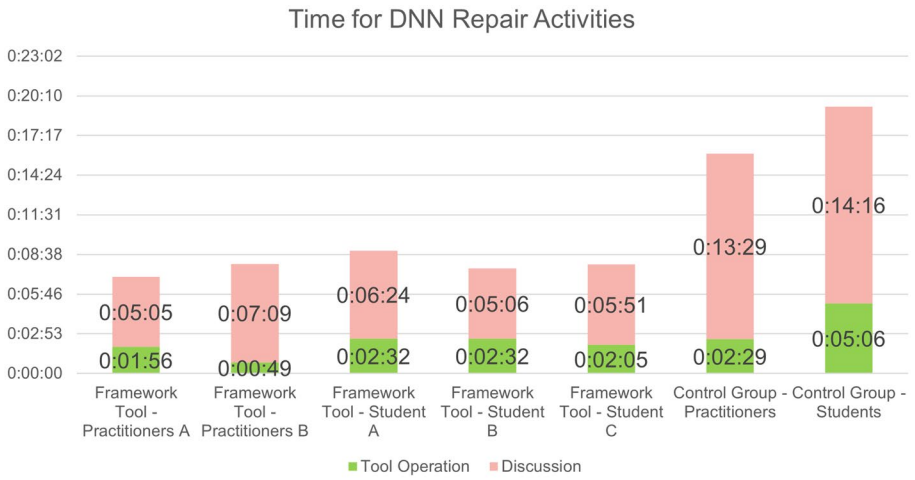


Fig. 25 Time needed for each group to finish the DNN repair task

Table 6 Summary of ML model performances trained by each group. Groups include practitioner control (C-P), practitioner framework (FW-P), student control (C-S), and student framework (FW-S). Underlined values indicate failure in satisfying the desired value

No	Performance Metrics	Desired Value	C-P	FW-P	C-S	FW-S
1	Overall accuracy	≥0.90	0.96	0.95	0.96	0.94
2	Overall precision	≥0.80	0.94	0.94	0.95	0.93
3	Misclassification rate from the label "Speed Limit 60" to "Speed Limit 80"	≤0.020	**0.022**	0.011	**0.038**	0.020
4	Misclassification rate from the label "Speed Limit 100" to "Speed Limit 120"	≤0.20	0.009	0.008	0.004	0.009
5	Precision of label "No Overtaking"	≥0.80	0.94	0.90	0.94	0.91
6	Recall of label "No Overtaking"	≥0.80	1.00	1.00	1.00	0.95

experience level, the control groups took more time than the framework groups to finish the DNN repair part.

6.3.2 Repaired ML models performances

Table 6 summarizes the performances of repaired ML models by the group, along with performance expectations. The aim of the experiment is for ML models to satisfy all desired values. The ML models from both framework groups satisfied all the desired values. In contrast, the control groups failed to satisfy the desired misclassification rate from the label ”Speed Limit 60” to ”Speed Limit 80” but satisfied the remaining desired values.

Except for the misclassification rates, both control groups produced ML models with better performance than their respective framework groups. In both practitioner and student groups, the misclassification rate for the label “Speed Limit 60” to “Speed Limit 80” of the framework group’s ML model was half that of the control group’s ML model. A similar reduction was observed for the student group in the case of “Speed Limit 100” to “Speed Limit 120” for the students, but the difference in practitioners was much smaller. The characteristics of the misclassification rate should be discussed as a performance metric to understand the findings.

6.3.3 Questionnaire result

Here, the results are visualized using diverging bar charts (Heiberger & Robbins, 2014) because the differences in perceptions can easily be identified. Our analysis focused on the positivity or negativity of the sentiment by group.

Figure 26 summarizes the answers to Q1. The framework group had a more positive sentiment (72.3%) than the control group (62.5%). This was a difference of 11.2%, suggesting that the control group felt their tracing result was more prone to mistakes.

Figure 27 summarizes the answers to Q2, Q3, and Q4. As the tracing comprehensiveness increased, the sentiment of the control groups decreased from 85.7% positive in Q2 to 50% positive in Q3 and 12.5% positive for Q4. In contrast, the framework groups showed 100% positivity for Q2 and Q3 and had a slight drop to 88.8% positivity in Q4.

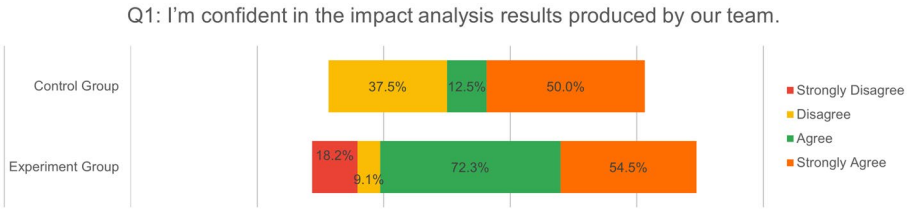


Fig. 26 Summary of the answers to Q1

The answers to Q5 suggest that the M^{3}S implementation of the DNN repair solution has major weaknesses (Fig. 28). The control group indicated 100% positivity, whereas the framework group had only 62.5% positivity in their answers. The significant gap between the groups implied that something is not working well for the framework groups and needs to be fixed.

Figure 29 shows the answers to Q6. The 1.5% difference in positivity by group suggested a slight disadvantage in using M^{3}S in terms of efficiency of deciding the input of the DNN repair tools. However, the answers to Q7 showed that M^{3}S had a significant advantage for evaluating the output as the framework group showed a 38.8% higher positivity than the control group.

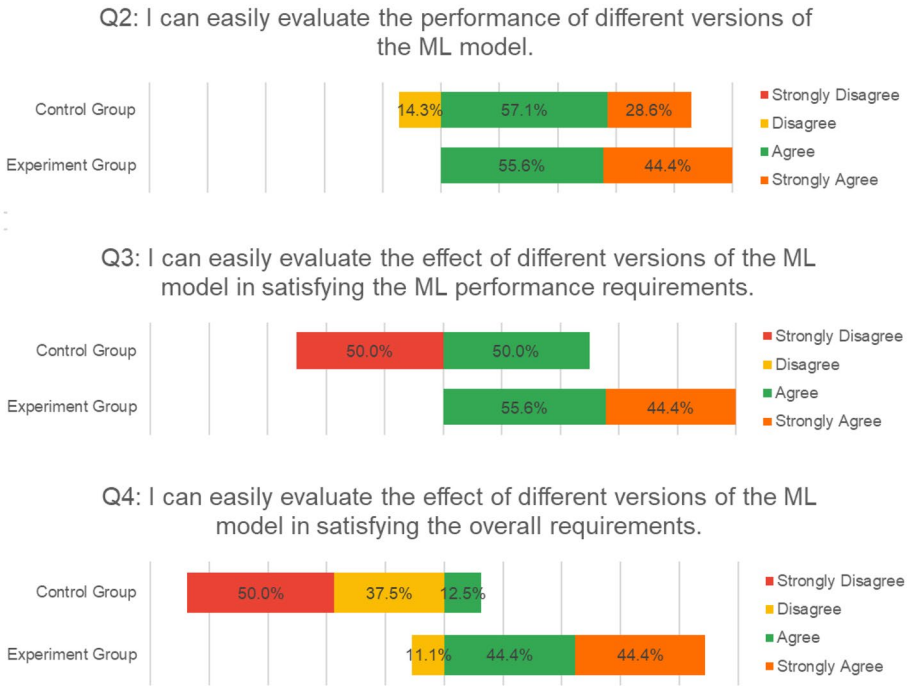


Fig. 27 Summary of the answers to Q2, Q3, and Q4

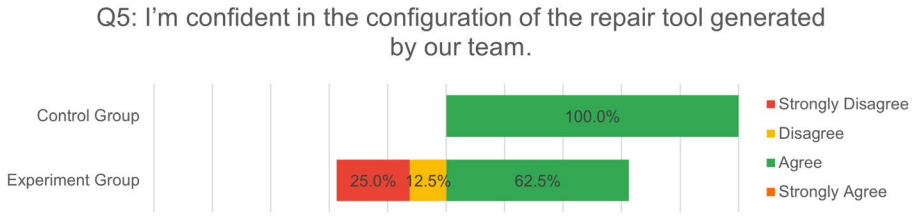


Fig. 28 Summary of the answers to Q5

6.4 Answers to research questions

6.4.1 RQ4. Can M^{3}S efficiently facilitate the impact analysis from ML model performance?

For the tool operation, M^{3}S is a more efficient approach than the natural language specification and CLI-based tool combination. However, the difference in discussion time must be considered to properly understand the reason for the time difference. This finding is relevant for the practitioner framework group, which spent a lot of time discussing. An interesting finding from their post-experiment discussion is the comments about model correctness. The practitioner framework group took the time to discuss whether the models' logical points are correct. None of the other groups engaged in this type of analysis. It is plausible that the time spent by the practitioner framework group discussing the correctness is due to the clarity of the connection between the elements rather than the difficulties in finishing their task. With that in mind, we argue that the M^{3}S approach is more efficient than the ad-hoc approach used by the control group. Moreover, the experience level did not affect the required time for completion.

> The M^{3}S approach is more efficient than the ad-hoc approach. This is consistent with the fact that the framework group needed much less time than the control group, regardless of experience level.

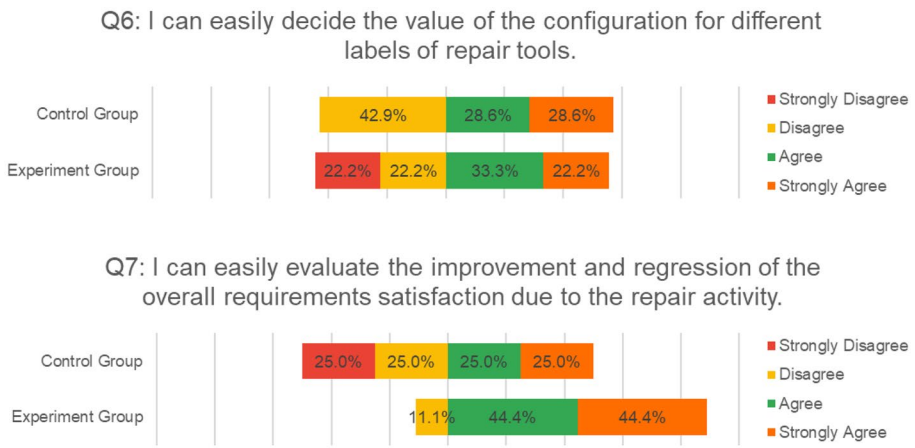


Fig. 29 Summary of the answers to Q6 and Q7

6.4.2 RQ5. Can M^{3}S efficiently facilitate the analysis of parameters for repair activities?

Although the difference by group is not significant, the difference by group and experience level is. In both cases, the framework groups took less time to complete their task than the control groups. This finding is more pronounced in the students. The operation time of the student control group is almost twice that of the student framework group. Additionally, the participants' comments further support this. One participant in the control group commented on the need for GUIs to complete all the DNN repair tasks.

Each group employed a similar format in their discussions: identify important classes to repair, decide the exact number for the configuration, and evaluate the result of the DNN repair. Although similar formats were utilized, the control groups had longer discussions. Comments from the practitioner control group emphasized their difficulties in deciding the value of the configuration and analyzing the side effects of the process.

> The M^{3}S approach more efficiently facilitates parameter analysis in the repair activity than a more ad-hoc approach. M^{3}S supports a faster analysis to make and evaluate the decisions of the DNN repair activities. However, experience level also affects the analysis.

6.4.3 RQ6. Does M^{3}S help users train a better ML model through integrated solutions?

Participants in the framework groups repaired the model to satisfy all the desired values of ML performance. In contrast, the control groups failed to repair the model to satisfy the desired misclassification rate. It should be noted that the control groups' models outperformed the framework groups' models for other performance metrics. This finding leads to interesting questions about the nature of the requirements to be satisfied.

An important difference between achieving the misclassification rate and other performance metrics is its relationship with multiple labels. Accuracy, precision, and recall simply focus on the population of a single label, whereas the misclassification rate requires a deeper analysis of the connection between the label misclassified from and the label being misclassified into. This leads to more complex decision-making when configuring the priority of labels in DNN repair. We argue that M^{3}S helps guide decision-making in more complex situations. This argument is supported by the fact that the framework groups better repaired models for the misclassification rate while simultaneously satisfying all desired values.

> The M^{3}S approach helps the user train a better model in complex situations where the relationships between labels are important. However, the efficacy in a straightforward situation when only a single metric is important is lower compared to the ad-hoc approach.

6.4.4 RQ7. How confident are users about the impact analysis result of M^{3}S?

Participants in the framework groups are more confident in their ML performance monitoring task results compared to the control groups. This is supported by the positive comment about the M^{3}S capability from the practitioner framework group, which stated

that the automatic detection of satisfied and unsatisfied elements helped them navigate the effect of ML model performance even if they had to recheck it. In contrast, both control groups were concerned about their result if the requirements were larger than the ones used in the experiments.

> The automatic impact signaling of M^{3}S increases users' confidence in the impact analysis result compared to a more ad-hoc approach because the only result needs to be rechecked using M^{3}S instead of analyzing everything from scratch. The impact becomes more significant as the number of requirements increases.

6.4.5 RQ8. How confident are users about the usability of the impact analysis of M^{3}S?

The answers to Q2, Q3, and Q4 show interesting results. The difference in the sentiments becomes more significant as the scope of the requirements broadens. Both groups started with similar confidence levels. However, the sentiments of the control group about their capability to evaluate the overall requirements became extremely negative, implying that the ad-hoc approach is highly unreliable in such cases. In contrast, the framework groups showed stable responses for all cases. The participants who used M^{3}S felt confident that they can work with all levels of requirements, not just the ML performance requirements, indicating that M^{3}S has a better usability for evaluating the impact than a more ad-hoc approach.

> The M^{3}S approach is more usable than ad-hoc approaches. This sentiment is more pronounced when impact analysis requires a more comprehensive approach. M^{3}S should be more beneficial when the scope of the impact is broad.

6.4.6 RQ9. How confident are users about the solution integration in M^{3}S?

The integration of DNN repair into our framework has some major issues. Unfortunately, the reason is unclear from the answers to Q5. Both the practitioner and student framework groups completed the repair and satisfied all the ML performance requirements, whereas neither control group did. However, the post-experiment discussion may provide some insight. One participant noted that the DNN repair process is not visible from the modeling side. Another participant reported unfamiliarity with the tool and the need to see the detailed process of the DNN repair. Based on the post-discussion comments, we assume our integration has some issues. Our integration is overly encapsulated and lacks the transparency of more traditional approaches. Additional information is necessary to properly answer this question.

> The confidence of the integrated DNN repair in M^{3}S result is low. This may be due to the lack of transparency, especially when users are unfamiliar with the solution. In the future, an in-depth evaluation of the internal process, especially the generality of the findings on different types of solutions, should be conducted.

6.4.7 RQ10. How confident are users about the usability of the solution integration in M^{3}S?

M^{3}S shows a slight disadvantage compared to the ad-hoc groups, but the difference may be caused by the weighting of extreme options. Nevertheless, the problems from both groups should be evaluated to understand whether they felt the same difficulties during the experiment. The control group noted that an ad-hoc feeling during the decision of the configuration value made it difficult. The framework group did not make a similar comment, suggesting the groups encountered different problems. Moreover, a participant in the control group stated the need for a GUI-based approach for the repair, whereas the framework group indicated the ease of not having to write CLI commands manually. However, the framework group mentioned that the randomness of the ML solution is an issue. This situation is likely to be felt since both sections of the framework group conducted the repair during the experiment more than once and tried to make sense of the detailed effect of the configuration. Unfortunately, randomness is a characteristic of the ML training process. Addressing randomness is beyond the scope of this experiment.

Overall, the comments suggest that our approach has better usability than more ad-hoc approaches for deciding the value of the configuration, which is a limitation of the ML solution itself. The answers to Q7 show that the sentiment of the framework group is significantly more positive than the control group, indicating that the solution integration implemented in M^{3}S has a higher usability than ad-hoc approaches.

> Users have a higher confidence in the solution integration's usability of M^{3}S than the ad-hoc approach. The main reason for is the availability of GUI support in M^{3}S allows for more reason-based decision-making, especially when evaluating the solution's successes and side effects.

6.5 Threat to validity

There are several threats to the validity of the experiment. One internal threat is the participants' familiarity with the methods. To mitigate this threat, we randomized the groups while ensuring that both groups had similar experience levels. Another internal threat is the participants' bias towards a particular method. We countered potential bias by not sharing the aim of this research with the participants.

An external threat is sampling bias. Our experiment included two different sections to improve the generality of the results. We also selected participants with differing levels of experience in ML and software development to ensure the quality of the sampling.

7 Discussion

This section addresses benefits, limitations, and other insights of M^{3}S gained from the case study and controlled experiment.

7.1 Benefits - comprehensive feedback loop

The case study highlighted that the repair process and proposal to improve the sensors are based on feedback from previous actions. Similarly, the proposal to improve the camera sensors was also made by understanding the limitations of the repaired ML model. At the same time, the controlled experiment has shown positive sentiment toward the impact analysis inside M^{3}S on top of the fact that the users of the framework managed to create better ML models. The development process in M^{3}S drives an informed decision, which is implemented again in another development process, creating a feedback loop between the analysis and the implementation.

Table 7 compares the scope of M^{3}S to other approaches. Compared to other multi-view analysis approaches (Villamizar et al., 2022; Nalchigar et al., 2021), M^{3}S covers more aspects, namely safety and argumentation. Additionally, M^{3}S facilitates an integrated feedback loop between the analysis and the ML training and testing pipeline. Compared to pure analysis approaches, M^{3}S supports a dynamic environment for continuous evaluation and improvement of decisions behind ML system development through the traceability provided by the integration (Galvao and Goknil, 2007).

The analysis scope of M^{3}S differs from model-transformation approaches (Moin et al., 2022; Koseler et al., 2019). Model-transformation approaches fit better with the ML model training without considering other aspects required for an overall ML system. In contrast, M^{3}S supports the development of more robust ML systems that integrate ML model analysis and higher-level requirements necessary for a successful large-scale ML system development. The integration facilitates the validation of potentially unrealistic expectations of the capability of the ML model and provides guidance for their refinement into a more realistic one (Nahar et al., 2023).

7.2 Benefits - documented integrated solutions

An underlying principle of M^{3}S is solution integration to improve the quality of the ML model in the multi-view analysis part. The concept of integration is not unique. Table 7 summarizes other works that have explored the idea of using model transformation to generate training pipeline source code. The distinction between M^{3}S and other approaches lies in the generated part of the pipeline through model transformation; M^{3}S produces the configuration of the preferred solutions to improve the ML model quality. The underlying benefit of this approach is that the experimentation of the proposed solution is tightly synchronized with the analysis part, enhancing both traceability and reproducibility.

The experimentation using different approaches to DNN repair is documented directly inside the solution of the safety case (Fig. 19). This facilitates tracing the improvement or degradation of the ML model quality because the configuration of proposed solutions is well represented inside the models. Although the case study only evaluated a single solution, other solutions should show similar benefits. For example, the decision to experiment with data augmentation to balance data distribution can be properly reflected in the model with the configuration of the augmentation written as a description of the solution.

The benefits of the M^{3}S style of integration should be more apparent in longer loops of adding and removing solutions to the ML model training compared to the model transformation approaches summarized in Table 7. The decisions behind each experiment are more properly reflected in the model rather than directly placing the ML

Table 7 Comparison of M^3S with other approaches

Aspect	M^3S	Villamizar et al. (2022)	Nalchigar et al. (2021)	ML-Quadrat (Moin et al., 2022)	Koseler et al. (2019)
Business requirements	Value	ML Objectives	Business	-	-
Users	Value	UX	Business	-	-
ML task and performance	ML Task, Goal	Model	Analysis	Textual model	Class diagram
Architecture	Architecture	Infrastructure	-	-	-
Data	ML Task	Data	Data Preparation	Textual model	Class diagram
Safety	Safety	-	-	-	-
Argumentation	Argumentation	-	-	-	-
Data engineering	-	-	-	-	-
ML model training	Integrated pipeline	-	-	Code generation	Code Generation
ML performance testing	Integrated pipeline	-	-	Code generation	Code Generation
Other ML solutions	Integrated pipeline	-	-	-	-

pipeline design in the model. Given the characteristic of ML systems where monitoring for model degradation from drifts is prominent (Bayram et al., 2022), the ability to retrace past decisions is highly beneficial.

7.3 Limitation - variation of machine learning tasks

One issue discovered while developing and validating M^{3}S is the vast possibilities of tasks in the ML model. The case study and controlled experiment only considered the multi-class classification task as other ML tasks were beyond the scope of this research. This limits our findings to the multi-class classification task. Although this study demonstrates the versatility of M^{3}S in handling different ML tasks, in the future, the generality of M^{3}S in handling different tasks should be validated.

A comparison to other works utilizing a framework for different ML tasks suggests that M^{3}S has the versatility necessary to handle various ML tasks. Our early works on the framework extensibility showed promising results in named entity recognition (NER) problems for a rule-based text transformation (Takeuchi et al., 2023). Extension of M^{3}S with activity-driven analysis demonstrated the ability to handle optical character recognition (OCR) tasks (Tanaka et al., 2023). By comparing the characteristics of the NER and OCR task to the image classification in our case study, we can explore the features of M^{3}S that work differently from the case study we presented.

The performance metrics are one crucial aspect that differs from our case study. The performance metrics of image classification mainly rely on classical ones, such as accuracy, precision, and recall while The OCR's accuracy varies between characters and word-level error rates. This difference is also true for other ML tasks, such as the mean square error (MSE) for regression and intersection-over-union (IoU) for semantic segmentation. While the analysis side of this difference can be handled by the Value, ML Task, and Goal views, implementation in the integrated training pipeline will differ. A modification or extension to facilitate the variation of ML performance metrics is necessary to fit the requirements of individual projects.

The same is also true for the available solutions to improve the quality of the ML model. The DNN repair in the case study works mainly for image classification problems and only one variant among all possible DNN repair tools, as each technique has its own parameters and processes. Additionally, different ML tasks may not have the same solutions available. Moreover, even the same type of solution, such as data augmentation, will have variations for different types of data required by each ML task. An extension is necessary to facilitate the specific needs of solutions for each possible case because exhaustively providing all possible solutions is expensive.

In conclusion, the limitation in handling the myriad of ML tasks lies in the implementation. Although other works have explored the versatility of M^{3}S on the analysis side, challenges on the implementation side have yet to be solved. In the future, a general, extensible interface between the modeling side and the training pipeline should be investigated to enable an efficient extension of the integrated pipeline to facilitate different configurations of the ML performance and parameters into integrated solutions.

7.4 Limitation-platform-agnosticism

The support tool in the case study and controlled experiment is based on the Astah* System Safety and DVC. We selected this platform due to its suitability for the reuse of existing functions. However, this also raises a concern about the platform-agnosticism of M^{3}S and its usability among other software development processes and tools. The concern is important as developers commonly have existing processes and tools working. This subsection will address such concerns and guide future works in this direction.

For analysis and modeling, we suggest that any modeling tool that supports model analysis can be utilized. Since modeling approaches are not developed in a platform-specific manner, they should work in any modeling environment. However, applying the metamodel and integration into the training pipeline may be challenging. The metamodel-based modeling approach of M^{3}S requires a modeling environment that supports such functions either natively or via custom functionality. Integration also needs a custom communication function, which requires extensible modeling tools. Two common modeling environments come with those functions: Sparx Systems Enterprise Architect[4] and Eclipse Modeling Framework[5]. However, other modeling environments with those features are also possible to support M^{3}S.

Various ML training pipelines can be implemented in the integrated training pipeline. We argue that our metamodel facilitates the integration of the modeling part with the training pipeline based on similar research by (Idowu et al., 2022). M^{3}S' integrated metamodel works in a similar manner to Idowu's proposed general metamodel for experiment management tools aimed at general integration. The integrated metamodel connects artefacts generated by the pipeline, which is managed by experiment management tools, into the concepts inside the models. This allows for the implementation of the integrated pipeline using different platforms because such pipelines support the general artifacts of ML training described in the integrated metamodel (Idowu et al., 2023).

The present case study and controlled experiment did not consider the extensibility process of the framework, as described in subsection 4.3. A case study involving customization and extension has been discussed briefly (Husen et al., 2023). The extension process should allow developers to evaluate and customize M^{3}S to their needs and constraints, including the use of existing processes and tools. Moreover, efforts have been made to generalize currently implemented solutions (Runpakprakun et al., 2023).

7.5 Limitation - internal solution uncertainty

Uncertainty due to the probabilistic nature of ML models is a crucial problem in ML system development. M^{3}S aims to support decision-making and management by documenting the decisions made during the development and operation of ML systems. The extensive nature of M^{3}S should support understanding the impact of varying ML performance due to uncertainties and other aspects. Additionally, M^{3}S should be able to evaluate if previously utilized solutions are still relevant under new conditions.

[4] https://sparxsystems.com/.

[5] https://eclipse.dev/modeling/emf/.

However, the solution's internal uncertainty remains in the decision-making. This situation is especially true during the DNN repair process in the controlled experiment. The non-deterministic nature of the method leads to low confidence for participants working with M^{3}S as they retried the repair. They found that their configuration was inconsistent with their expectations. Combined with the black-box approach provided by the integrated tool, the perception of how the solution works was too vague for the developers' comfort.

8 Conclusion and future works

This paper proposes and evaluates M^{3}S, which is an approach to facilitate a consistent and comprehensive analysis of ML systems. Herein, we elucidate the benefits of M^{3}S through a case study and experiment. The evaluation demonstrated that M^{3}S clarifies existing decisions and enhances the performance evaluation of ML models. The case study involved a series of guided decisions through different views from the top business goals into executable solutions and testable ML performance requirements, while the experiment confirmed the evaluation ease of the ML performance requirements and the related higher-level decisions.

Moreover, the consistency between decisions and implementation is managed efficiently during training. The decisions made in the models directly influence the execution of solutions implemented in the training pipeline (e.g., the solution part of the safety case). However, M^{3}S provides limited assistance in navigating the internal uncertainty of solutions, resulting in a less positive response regarding the trustworthiness of solution implementation. This limitation may have several origins, such as the lack of transparency in the internal processes. Further improvements in the solution implementation are necessary.

Finally, the generality of the framework to handle different ML tasks remains unclear. Although there are some indications of generality, the evaluation is insufficient to draw a proper conclusion. In the future, experiments on the benefits and limitations of utilizing M^{3}S for ML tasks other than classification are necessary to demonstrate the generality of the framework.

Other future tasks include exploring more views and the extensibility of M^{3}S, as well as improving the implementation of integrated solutions. One direction is to generalize supported modeling, integrated solutions, and experimentation environments. A second direction is to provide a plug-and-play approach to the extensibility of M^{3}S. A third direction is to evaluate the universality of the M^{3}S framework, especially on different ML tasks, and to extend the case study to include continuous monitoring.

Author contributions J.H.H. wrote the main manuscript text and the design of the overall process. J.R. developed the integration mechanism between modeling and ML training environment. H.T.T, H.W., and N.Y. developed and extended the integrated metamodel. H.T. contributed to the evaluation of the design of the extension process of the framework. H.W., N.Y., and Y.F. supported the validations. All authors reviewed the manuscript.

Funding This work was supported by JST-Mirai program grant number JPMJMI20B8 and JST SPRING grant number JPMJSP2128.

Availability of data and materials All the diagrams and ML models developed during the case study have been deposited in https://doi.org/10.5281/zenodo.8429584. The dataset utilized for training during the case study and experimentation, GTSRB, is available as a public dataset by the original authors at https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html

Declarations

Competing interest The authors declare no competing interests.

References

Batot, E. R., Cabot, J., & Gérard, S. (2021). (not) yet another metamodel for traceability. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pp 787–796. https://doi.org/10.1109/MODELS-C53483.2021.00125

Bayram, F., Ahmed, B. S., & Kassler, A. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems, 245*, 108632. https://doi.org/10.1016/j.knosys.2022.108632

Bishop, C. (2013). Model-based machine learning. *Philosophical Transactions Series A, Mathematical, physical, and Engineering Sciences, 371*, 20120222. https://doi.org/10.1098/rsta.2012.0222

Chuprina, T., Méndez, D., & Wnuk, K. (2021). *Towards artefact-based requirements engineering for data-centric systems*, vol. 2857. Essen, Germany. https://ceur-ws.org/Vol-2857/re4ai1.pdf

Dorard, L. (2015). Machine Learning Canvas. https://www.machinelearningcanvas.com/

El Hamlaoui, M., Bennani, S., Nassar, M., Ebersold, S., & Coulette, B. (2018). A mde approach for heterogeneous models consistency. In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering. ENASE 2018*, pp. 180–191. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT. https://doi.org/10.5220/0006774101800191

Galvao, I., & Goknil, A. (2007). Survey of traceability approaches in model-driven engineering. In *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pp 313–313. https://doi.org/10.1109/EDOC.2007.42

Heiberger, R., & Robbins, N. (2014). Design of diverging stacked bar charts for likert scales and other applications. *Journal of Statistical Software,57*, 1–32. https://doi.org/10.18637/jss.v057.i05

Hosseinzadeh Kassani, P., & Teoh, A. (2016). A new sparse model for traffic sign classification using soft histogram of oriented gradients. *Applied Soft Computing,52*. https://doi.org/10.1016/j.asoc.2016.12.037

Husen, J. H., Washizaki, H., Tun, H. T., Yoshioka, N., Fukazawa, Y., Takeuchi, H., Tanaka, H., & Munakata, K. (2023). Extensible modeling framework for reliable machine learning system analysis. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pp 94–95. https://doi.org/10.1109/CAIN58948.2023.00022

Husen, J., Washizaki, H., Yoshioka, N., Tun, H., Fukazawa, Y., & Takeuchi, H. (2023). Metamodel-Based Multi-View Modeling Framework for Machine Learning Systems. In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering - MODELSWARD*, pp 194–201. SciTePress, Lisbon, Portugal. https://doi.org/10.5220/0011699600003402. INSTICC.

Idowu, S., Strüber, D., & Berger, T. (2022). *Emmm: A unified meta-model for tracking machine learning experiments*, pp 48–55. Institute of Electrical and Electronics Engineers Inc., Gran Canaria, Spain. https://doi.org/10.1109/SEAA56994.2022.00016

Idowu, S., Strüber, D., & Berger, T. (2023). Asset management in machine learning: State-of-research and state-of-practice. *ACM Computing Surveys, 55*, 1–35. https://doi.org/10.1145/3543847

Iso, iec, ieee international standard - software engineering - software life cycle processes - maintenance. (2022). *Standard*. Geneva, CH: International Organization for Standardization.

Japan Automobile Manufacturers Association, I. (2021). Automated driving safety evaluation framework ver 2.0. *Technical Report*.

Khomh, F., Adams, B., Cheng, J., Fokaefs, M., & Antoniol, G. (2018). Software engineering for machine-learning applications: The road ahead. *IEEE Software, 35*(5), 81–84. https://doi.org/10.1109/MS.2018.3571224

Kirchhof, J. C., Kusmenko, E., Ritz, J., Rumpe, B., Moin, A., Badii, A., Günnemann, S., & Challenger, M. (2022). Mde for machine learning-enabled software systems: A case study and comparison of montianna & ml-quadrat. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. MODELS '22*, pp. 380–387. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3550356.3561576

Koseler, K., McGraw, K., & Stephan, M. (2019). Realization of a machine learning domain specific modeling language: A baseball analytics case study. In *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development. MODELSWARD 2019*, pp 13–24. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT. https://doi.org/10.5220/0007245800130024

Langford, M. A., Chan, K. H., Fleck, J. E., McKinley, P. K., & Cheng, B. H. C. (2021). Modalas: Model-driven assurance for learning-enabled autonomous systems. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pp 182–193. https://doi.org/10.1109/MODELS50736.2021.00027

Letier, E. (2001). Reasoning about agents in goal-oriented requirements engineering.

Leveson, N. G. (2012). Engineering a Safer World: Systems Thinking Applied to Safety. *The MIT Press, Cambridge, Massachusetts.* https://doi.org/10.7551/mitpress/8179.001.0001

Likert, R. (1932-1985). *A Technique for the Measurement of Attitudes* / by Rensis Likert. Archives of psychology ; no. 140. [s.n.], New York.

Lima, A., Monteiro, L., & Furtado, A. (2022). *Mlops: Practices, maturity models, roles, tools, and challenges – a systematic literature review*, pp. 308–320. SCITEPRESS - Science and Technology Publications, Online. https://doi.org/10.5220/0010997300003179

Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M., & Wagner, S. (2022). Software engineering for ai-based systems: A survey. *ACM Transactions on Software Engineering and Methodology,31*(2). https://doi.org/10.1145/3487043

Matulevičius, R., & Heymans, P. (2007). *Visually effective goal models using kaos, 4802*, 265–275. https://doi.org/10.1007/978-3-540-76292-8_32

Minka, T., Winn, J. M., Guiver, J. P., Zaykov, Y., Fabian, D., & Bronskill, J. (2018) *Infer.NET 0.3*. Microsoft Research Cambridge. http://dotnet.github.io/infer

Moin, A., Wattanavaekin, U., Lungu, A., Challenger, M., Badii, A., & Günnemann, S. (2022). Enabling automated machine learning for model-driven AI engineering. CoRR abs/2203.02927. https://doi.org/10.48550/arXiv.2203.02927

Moin, A., Challenger, M., Badii, A., & Günnemann, S. (2022). A model-driven approach to machine learning and software modeling for the iot: Generating full source code for smart internet of things (iot) services and cyber-physical systems (cps). *Software and Systems Modeling, 21*, 987–1014. https://doi.org/10.1007/s10270-021-00967-x

Nahar, N., Zhang, H., Lewis, G., Zhou, S., & Kästner, C. (2023). A meta-summary of challenges in building products with ml components – collecting experiences from 4758+ practitioners. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pp 171–183. https://doi.org/10.1109/CAIN58948.2023.00034

Nahar, N., Zhou, S., Lewis, G., & Kästner, C. (2022). Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process. In *Proceedings of the 44th International Conference on Software Engineering. ICSE '22*, pp 413–425. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3510003.3510209

Nalchigar, S., Yu, E., & Keshavjee, K. (2021). Modeling machine learning requirements from three perspectives: a case report from the healthcare domain. *Requirements Engineering, 26*, 1–18. https://doi.org/10.1007/s00766-020-00343-z

Ozkaya, I. (2020). What is really different in engineering ai-enabled systems? *IEEE Software, 37*(4), 3–6. https://doi.org/10.1109/MS.2020.2993662

Pereira, A., & Thomas, C. (2020). Challenges of machine learning applied to safety-critical cyber-physical systems. *Machine Learning and Knowledge Extraction,2*. https://doi.org/10.3390/make2040031

Rahman, M. S., Khomh, F., Hamidi, A., Cheng, J., Antoniol, G., & Washizaki, H. (2023). Machine learning application development: practitioners' insights. *Software Quality Journal,* pp. 1–55. https://doi.org/10.1007/s11219-023-09621-9

Reineke, J., Stergiou, C., & Tripakis, S. (2019). Basic problems in multi-view modeling. *Software & Systems Modeling,18*. https://doi.org/10.1007/s10270-017-0638-1

Runpakprakun, J., Husen, J. H., Washizaki, H., Yoshioka, N., & Fukazawa, Y. (2023). Towards integrated model-based machine learning experimentation framework. In *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*, pp 593–594. https://doi.org/10.1109/DSA59317.2023.00086

Safety aspects - guidelines for their inclusion in standards. (2014). *Standard*. Geneva, CH: International Organization for Standardization.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. -F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2. NIPS'15, pp 2503–2511. MIT Press, Cambridge, MA, USA.

Software engineering - systems and software quality requirements and evaluation (square) - quality model for ai systems. (2023). *Standard*. Geneva, CH: International Organization for Standardization.

Sotoudeh, M., & Thakur, A. V. (2021). Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. PLDI 2021*, pp. 588–603. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3453483.3454064

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks,* pp 1453–1460.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks,32*, 323–332. https://doi.org/10.1016/j.neunet.2012.02.016. Selected Papers from IJCNN.

Takeuchi, H., & Yamamoto, S. (2020). Business analysis method for constructing business-ai alignment model. *Proceedings of the 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES)(Procedia Computer Science),176*, 1312–1321. https://doi.org/10.1016/j.procs.2020.09.140

Takeuchi, H., Husen, J. H., Tun, H. T., Washizaki, H., & Yoshioka, N. (2023). Enterprise architecture-based metamodel for a holistic busines–it alignment view on machine learning projects. In *2023 IEEE International Conference on e-Business Engineering (ICEBE),* pp 8–15. https://doi.org/10.1109/ICEBE59045.2023.00013

Tanaka, H., Ide, M., Munakata, K., Washizaki, H., & Yoshioka, N. (2023). Activity-based modeling strategy for reliable machine learning system analysis targeting gui-based applications. In *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*, pp 135–143. https://doi.org/10.1109/DSA59317.2023.00026

Thiée, L. -W. (2021). A systematic literature review of machine learning canvases. Gesellschaft für Informatik, Bonn. https://doi.org/10.18420/informatik2021-101

Villamizar, H., Kalinowski, M., & Lopes, H. (2022). Towards perspective-based specification of machine learning-enabled systems. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp 112–115. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/SEAA56994.2022.00025. https://doi.ieeecomputersociety.org/10.1109/SEAA56994.2022.00025

Vogelsang, A., & Borg, M. (2019). Requirements engineering for machine learning: Perspectives from data scientists. In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp 245–251. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/REW.2019.00050. https://doi.ieeecomputersociety.org/10.1109/REW.2019.00050

Wan, Z., Xia, X., Lo, D., & Murphy, G. C. (2021). How does machine learning change software development practices? *IEEE Transactions on Software Engineering, 47*(9), 1857–1871. https://doi.org/10.1109/TSE.2019.2937083

Wolf, C. T., & Paine, D. (2020). Sensemaking practices in the everyday work of ai/ml software engineering. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW'20*, pp. 86–92. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3387940.3391496. https://doi-org.waseda.idm.oclc.org/10.1145/3387940.3391496

Xiang, Q., Zi, L., Cong, X., & Wang, Y. (2023). Concept drift adaptation methods under the deep learning framework: A literature review. *Applied Sciences,13*(11). https://doi.org/10.3390/app13116515

Authors and Affiliations

Jati H. Husen[1,2] · Hironori Washizaki[1] · Jomphon Runpakprakun[1] · Nobukazu Yoshioka[1] · Hnin Thandar Tun[1] · Yoshiaki Fukazawa[1] · Hironori Takeuchi[3]

✉ Jati H. Husen
jati.h@asagi.waseda.jp; jatihusen@telkomuniversity.ac.id

Hironori Washizaki
washizaki@waseda.jp

Jomphon Runpakprakun
k-jomphon@moegi.waseda.jp

Nobukazu Yoshioka
nobukazuy@acm.org

Hnin Thandar Tun
hninthandar003@gmail.com

Yoshiaki Fukazawa
fukazawa@waseda.jp

Hironori Takeuchi
h.takeuchi@cc.musashi.ac.jp

[1] Waseda University, Tokyo 169-8050, Japan

[2] Telkom University, Bandung 40257, Jawa Barat, Indonesia

[3] Musashi University, Tokyo 176-8534, Japan