Metamodel-Based Multi-View Modeling Framework for Machine Learning Systems

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

[1]*Waseda University, Tokyo, Japan*

[2]*Telkom University, Bandung, Indonesia*

[3]*Musashi University, Tokyo, Japan*

Keywords: Machine Learning, Consistency Check, Metamodel, Multi-View.

Abstract: Machine Learning systems provide different challenges for their development. However, machine learning systems also require specific attention toward traditional aspects of software engineering. This situation often leads developers to use different models to cover different views that need to be handled during machine learning system development which often leads to conflicting information between models. In this research, we developed a multi-view modeling framework for machine learning system development by utilizing a metamodel as its backbone for consistency. We have conducted a case study and controlled experiment to evaluate the framework. In conclusion, our framework does help manage the consistency between different views but relies heavily on the quality of available support tools.

1 INTRODUCTION

Machine learning (ML) systems face unique challenges towards achieving reliability (Husen et al., 2021). ML functionalities work in a probabilistic manner, leading to an unpredictable nature (Vogelsang and Borg, 2019). Functionalities are based on the characteristics of data used for training the deployed ML models. Ensuring data correctness requires a more analytical approach than traditional software functionalities, which are based on the developed source code. Furthermore, constant monitoring is necessary to detect whether the data become outdated or out of sync with actual conditions. Such a situation is known as data drift.

ML systems also encompass traditional aspects of software engineering (National Institute of Advanced Industrial Science and Technology, 2022). The necessity of functionalities towards achieving business goals is important for success. Communication of the ML models as part of a complex software architecture is also a concern. ML systems must not only communicate with other ML models but also with traditional software components.

This condition drives system developers to utilize multiple models to analyze different views of the ML system (Yoshioka et al., 2021). Due to the experimental nature of ML training, canvas-based models such as ML Canvas are often used for rapid brainstorming of the aspect of ML components, including data collection and the expected model performance (Thiée, 2021). However, such a canvas does not cover the relationship between the functionalities and the business goal or more specific analyses such as the safety of ML models when deployed in safety-critical systems.

Deploying different analysis models may lead to issues with consistency (Husen et al., 2022). Information may be misinterpreted between models, especially when the models are created independently. Because inconsistencies cause problems when monitoring detects data drift, they must be handled but the urgency of retraining ML models may not be captured as information about related business goals or safety concerns. Hence, inconsistencies may not be understood.

In this paper, we propose an integrated approach to handle consistencies of different views of ML system analysis named Multi-View Modeling Framework for ML Systems (M^3S). M^3S facilitates systematic analysis of different views of ML systems from higher business-level goals to lower ML training concerns. The analysis consists of several steps and uses different models guided by a metamodel to identify

Husen, J., Washizaki, H., Yoshioka, N., Tun, H., Fukazawa, Y. and Takeuchi, H.
Metamodel-Based Multi-View Modeling Framework for Machine Learning Systems.
DOI: 10.5220/0011699600003402
In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2023)*, pages 194-201
ISBN: 978-989-758-633-0; ISSN: 2184-4348

related elements. To evaluate the capability of M^3S we define the following research questions:

- RQ1. Can a metamodel guide the traceability between elements from different models in the M^3S framework?
- RQ2. Can a metamodel show the traceability links in models developed using the M^3S framework?
- RQ3. How usable are the traceability links in models developed using the M^3S framework for detecting inconsistencies?
- RQ4. How usable are the traceability links in models developed using the M^3S framework for rationalizing inconsistencies?

We conducted a case study to answer RQ1 and RQ2. The case study implemented M^3S to evaluate an autonomous driving vehicle. We also conducted a controlled experiment to compare identified inconsistencies using M^3S with natural language specifications to answer RQ3 and RQ4.

This research provides a new approach that integrates canvas-based analysis with more traditional analysis techniques for machine learning. We also present a more in-depth understanding of how different views of ML systems interact.

The rest of this paper is constructed as follows. Section 2 explains the background, including our motivation and related works. Section 3 describes the M^3S in detail from the utilized models, the metamodel as a foundation, and the overall design process. Section 4 presents the results of our case study and experiment to answer the research questions. Finally, section 5 concludes this paper and provides future works.

2 BACKGROUND

2.1 Motivating Example

Figure 1 shows our motivating example for this research, which includes a ML Canvas (Dorard, 2015) and safety case analysis for a ML component for object detection. The safety case facilitates safety analysis that is not covered in ML Canvas. Although both models represent level 3 self-driving cars on a highway, they were developed separately by different people during our pilot research. Inconsistencies between the two models may have occurred due to differences in understanding the scenarios included in the operational domain.

As described in the impact simulation section in ML Canvas, the operational design domain (ODD) expectation is described generally. However, the

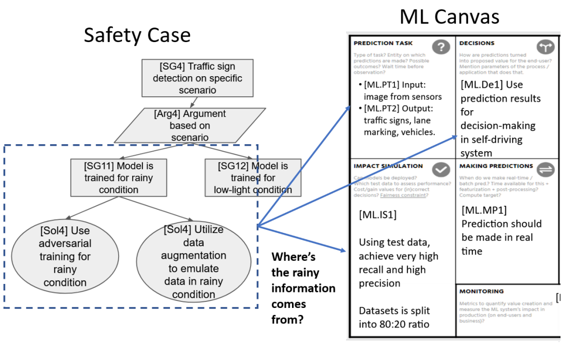

Safety Case
[SG4] Traffic sign detection on specific scenario
[Arg4] Argument based on scenario
[SG11] Model is trained for rainy condition
[SG12] Model is trained for low-light condition
[Sol4] Use adversarial training for rainy condition
[Sol4] Utilize data augmentation to emulate data in rainy condition
Where's the rainy information comes from?
ML Canvas
PREDICTION TASK
Type of task? Entity on which predictions are made? Possible outcomes? Wait time before observation?
[ML.PT1] Input: image from sensors
[ML.PT2] Output: traffic signs, lane marking, vehicles.
DECISIONS
How are predictions turned into proposed value for the end-user? Mention parameters of the process / application that does that.
[ML.De1] Use prediction results for decision-making in self-driving system
IMPACT SIMULATION
Can models be deployed? Which test data to assess performance? Cost/gain values for (in)correct decisions? Fairness constraint?
[ML.IS1]
Using test data, achieve very high recall and high precision
Datasets is split into 80:20 ratio
MAKING PREDICTIONS
When do we make real-time / batch pred.? Time available for this + featurization + post-processing? Compute target?
[ML.MP1] Prediction should be made in real time
MONITORING
Metrics to quantify value creation and measure the ML system's impact in production (on end-users and business)?

Figure 1: Motivating Example.

safety requirements in safety case analysis specifically describe the rainy condition as risky. Due to the difference in information between the two related models, the correctness of the information becomes questionable as the risk assessment may or may not include the rainy conditions. Such a situation may induce problems because the definition of dataset quality relies on both elements. The difference between these elements generates uncertainty in the prepared datasets. Incorrect assumptions may result in unhandled safety risks and accidents. Our framework aims to eliminate the uncertainty caused by inconsistencies inside the models.

2.2 Related Works

Several works have modeled a ML software system. Chuprina et al. proposed an artefact-based approach for modeling requirements of ML software (Chuprina et al., 2021). Their goal was to bridge the conceptual and application layer of ML systems, which is similar to ours. Nalchigar et al. proposed a modeling framework for ML software requirements. Their approach used three perspectives: business, analytics design, and data preparation (Nalchigar and Yu, 2018). Ishikawa and Matsuno also proposed an evidence-driven approach to model goals of a ML-based system (Ishikawa and Matsuno, 2020).

Other works have investigated safety-critical ML systems. Pereira and Thomas explained their findings on challenges in managing safety in ML systems (Pereira and Thomas, 2020). They explained hazards, which may arise due to inadequate definitions in requirements engineering activities such as data, object definition, and performance measures. Hawkins et al. proposed an assurance framework for an autonomous system named AMLAS (Hawkins et al., 2021).

In general, existing multi-view approaches for ML systems do not emphasize safety concerns, whereas existing safety approaches do not handle the connection between safety concerns and higher-level re-

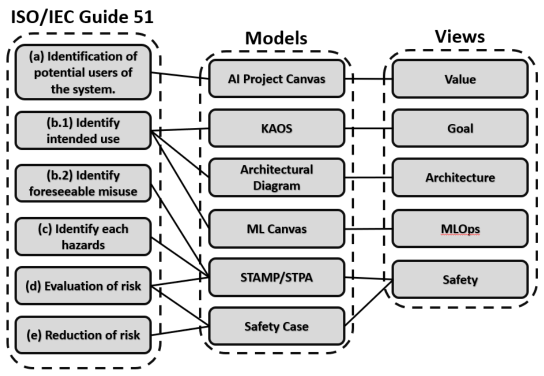


Figure 2: Relation of models of M^3S to ISO/IEC Guide 51.

quirements. M^3S aims to facilitate safety analysis of ML components while ensuring the connection between different aspects of development.

3 MULTI-VIEW MODELING FOR MACHINE LEARNING SYSTEM (M^3S)

3.1 Views of M^3S

To comprehensively understand a ML system, we define five views for analysis: value, MLOps, goal, architecture, and safety. All views are related to each other and together construct a comprehensive view of a reliable ML system.

Value is the center of all views. Value covers the description of the expected contribution of the ML system as a whole. It contains the definition of success and motivation of the project. Value describes the main business goal and the environment surrounding the ML system development project.

The second view is goal. Goals are objectives that must be completed for the project to provide the value and be successful. They are derived from the main business goal to describe the expected capabilities of the ML system and its components. Capabilities include both functional and non-functional ones.

The third view is architecture. Architecture describes how the developed ML models run their expected functionalities inside the software system. It explains how the ML models interact with other software components, receive the required input, and provide outputs to be processed by other components.

The fourth view is MLOps. MLOps describes how training and testing will develop ML models. It analyzes the needs of data for training and testing, the functionalities necessary due to the training results, and the expected performance of the ML models.

The final view is safety. ML models require specific attention to safety due to their probabilistic nature. Safety analyzes the safety risks and hazards related to the failure of ML systems and components. It generates countermeasures to reduce risks and act as an argumentation for the safety of the ML system.

3.2 Models of M^3S

M^3S is developed in compliance with ISO/IEC Guide 51. The selection for each view facilitates the view and the necessary process of ISO/IEC Guide 51. Figure 2 summarizes how the utilized models in M^3S bridge the view and process of ISO/IEC Guide 51. The process of ISO/IEC Guide 51 itself is defined as follows:

- a. Identify potential users of the system, including vulnerable users and others affected by the system;
- b. Identify the intended use of the system;
- c. Identify potential misuse of the system;
- d. Identify hazards that arise during different stages of usage;
- e. Estimate and evaluate risks to affected parties caused by each hazard;
- f. Minimize each intolerable risk.

To analyze the value, M^3S uses the AI Project Canvas as its model. The AI Project Canvas analyzes the requirements specific to the AI development project. It provides a platform to identify the main value provided by the ML system to the customer, the required skills of the development team to complete the project, and the project's revenue stream. The stakeholder and customer elements of the AI Project Canvas define the potential user and vulnerable parties during operations of the ML system.

Goal view utilizes KAOS goal modeling to divide the main value in the AI Project Canvas into smaller goals. The top-down approach of KAOS goal modeling derives achievable sub-goals from the business objectives by the capabilities of the ML components (Zickert,). The sub-goals describe the information needed to identify the system's intended use in ISO/IEC Guide 51. Additionally, the KAOS goal modeling approach is deployed into the uncertain condition of the ML system (Ishikawa and Matsuno, 2020).

Architecture view employs an architecture diagram to map the interaction between ML models and other software components. The architecture diagram describes how data is transformed and transferred between the different components inside the ML system.

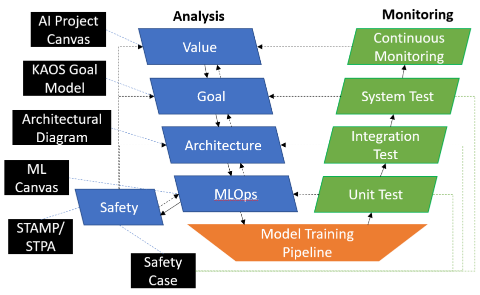


Figure 3: The M^3S process.

Information about the architectural design identifies potential hazards caused by a failure in the communication flow, which will be identified in STAMP/STPA later. In M^3S, the architecture is modeled in a SysML block diagram.

The ML Canvas defines the requirements from the MLOps view and describes the desired functionality and performance (Dorard, 2015). It also expresses the data source, the collection process, the scheduled retraining, and the redeployment frequency. The definition of data quality is the basis for defining solutions in safety case analysis. ML Canvas is used for each ML component in the ML system.

The safety view utilizes two models with different philosophies. STAMP/STPA conducts a safety analysis from the perspective of the interaction between the system and its components to identify possible misuse at both the system and component levels (Leveson, 2012). Then a safety case analyzes safety concerns in a more top-down manner to find the root cause of ML inadequacies in the training process (Hawkins et al., 2021). Both models understand, evaluate, and counter potential risks in ML systems and components.

3.3 Process of M^3S

M^3S has two main objectives. The first facilitates consistent analysis of different ML system development views. This objective mainly works as a prescriptive description of ML system requirements. The second efficiently identifies the impact of drifts in ML operations. This identification is achieved by traceability between the monitoring results and the related requirements. Figure 3 summarizes the M^3S process.

The objectives drive the process design. The process consists of two main phases: the analysis phase and the monitoring phase, which target the first and second phases, respectively. The training phase is another phase where model training is executed using a ML training pipeline developed according to the analysis phase results. The process is iterative. The monitoring phase results re-trigger the analysis phase. The analysis phase leads to the execution of the training phase. Finally, the training phase returns to the monitoring phase.

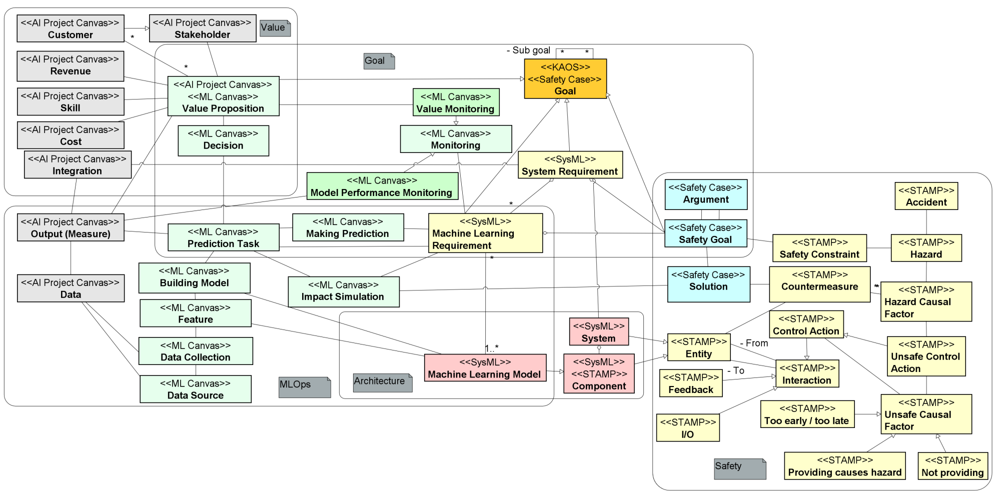
The analysis phase is conducted in a top-down manner. It begins from the value view. Then the value is divided into smaller achievable objectives in the goal view. Each capability is then assigned to the components in the view of architecture. Next, the development of ML components is defined in the MLOps view. The safety risks and countermeasures are specified in the safety view. For each view analysis, changes due to the analysis of the preceding views may occur to incorporate new findings such as implementing countermeasures in the safety view to the architectural design or expected ML performance.

The monitoring phase consists of four layers. Each layer is related to different views that must be reevaluated in case of failure. The lowest layer is the unit test result monitoring, which is related to the ML performance described in the MLOps view and the ML component capabilities in the goal view. Integration result monitoring is conducted after the unit test is passed successfully. This layer monitors successful integration, which is related to the design decision in the architectural view. The capabilities of successful integration are monitored in the system testing layer. Success in this layer is related to the capabilities in the goal view and countermeasures in the safety view. Finally, continuous monitoring is conducted for the ML system. Any changes in the success rate of achieving the value during the operation triggers reanalyzing and retraining of the ML model.

3.4 Metamodel of M^3S

M^3S provides a metamodel to define the relation of elements from different models utilized by each view. The metadata modeling process generates the metamodel. The generated metamodel provides an integrated view of all M^3S models. Figure 4 shows an integrated M^3S model implemented in a safety-critical system.

The metamodel is constructed and evaluated iteratively. The construction begins with metamodeling the concept behind each utilized model. Then the team evaluates the correctness of each metamodel. Afterward, each metamodel integrates elements from different models with similar concepts and either merges them into a single concept or connects them using four types of mapping: ”same”, ”similarity”, ”aggregation”, and ”contribution” (El Hamlaoui et al., 2018). Construction and evaluation of the metamodels are conducted internally by the authors. External parties then evaluated all results that passed the in-



<<AI Project Canvas>> Customer
<<AI Project Canvas>> Stakeholder
Value
<<AI Project Canvas>> Revenue
<<AI Project Canvas>> Skill
<<AI Project Canvas>> Cost
<<AI Project Canvas>> Integration
<<AI Project Canvas>> <<ML Canvas>> Value Proposition
<<ML Canvas>> Decision
Goal
- Sub goal
<<KAOS>> <<Safety Case>> Goal
<<ML Canvas>> Value Monitoring
<<ML Canvas>> Monitoring
<<SysML>> System Requirement
<<ML Canvas>> Model Performance Monitoring
<<AI Project Canvas>> Output (Measure)
<<ML Canvas>> Prediction Task
<<ML Canvas>> Making Prediction
<<SysML>> Machine Learning Requirement
<<AI Project Canvas>> Data
<<ML Canvas>> Building Model
<<ML Canvas>> Impact Simulation
<<ML Canvas>> Feature
<<ML Canvas>> Data Collection
<<ML Canvas>> Data Source
MLOps
Architecture
<<SysML>> Machine Learning Model
<<SysML>> System
<<SysML>> <<STAMP>> Component
<<Safety Case>> Argument
<<Safety Case>> Safety Goal
<<Safety Case>> Solution
<<STAMP>> Accident
<<STAMP>> Safety Constraint
<<STAMP>> Hazard
<<STAMP>> Countermeasure
<<STAMP>> Hazard Causal Factor
<<STAMP>> Control Action
<<STAMP>> Entity
- From
- To
<<STAMP>> Unsafe Control Action
<<STAMP>> Feedback
<<STAMP>> Interaction
<<STAMP>> I/O
<<STAMP>> Too early / too late
<<STAMP>> Unsafe Causal Factor
Safety
<<STAMP>> Providing causes hazard
<<STAMP>> Not providing

Figure 4: Integrated metamodel of M^3S.

ternal evaluation. Revisions are made based on comments on the external evaluation.

Table 1 shows an example of the results from concept mapping. The metamodel merges elements with the same concept such as value propositions from AI Project Canvas and ML Canvas into a single concept. The concept of goal from KAOS goal modeling is the most connected to other models due to its similarity with safety goals from the Safety Case, value propositions from the AI Project Canvas, and value propositions from ML Canvas. Solutions from safety case are contributing to the countermeasures from STAMP/STPA.

3.5 Traceability Links

The main feature of the framework is the traceability between information contained in elements from different models. Figure 5 summarizes this process. Traceability information is stored in a traceability matrix, which consists of data describing the type of relationship for each one-to-one pairing of all elements from all models inside the M^3S model. Information stored in the traceability matrix can be used for different purposes such as changing the impact simulation or consistency checking between different models.

The M^3S framework is a gateway to validate model modifications. Validation is conducted for every modification towards models inside M^3S based on conformance with relationships defined in the metamodel, leading to acceptance or rejection. For every accepted modification, traceability links are updated to ensure information on existing links is updated. In contrast, the framework rejects invalid modifications.

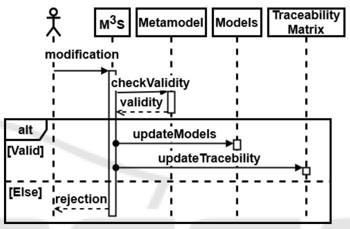


Figure 5: Flow of treaceability links management.

3.6 Consistency Checking

Our framework covers which elements should be connected to manage the traceability as this information can define the consistency that must exist between them. Some elements act as the basis to generate other elements, while others are copied to elements in other models. Consistency between safety and functional requirements is essential to ensure software systems' safety. For ML intensive systems, the need for consistency arises due to highly unpredictable conditions of the quality of utilized datasets and the operation domains. Although changes occur to adapt to uncertainty, some result in ad-hoc alterations of the ML system capabilities, which may cause safety risks while operating. The framework evaluates the consistency after such changes and determines whether another adaptation to the requirements is necessary.

3.7 Support Tool

We partially developed a tool to support the modeling process of M^3S. The tool facilitates navigation of the traceability between related elements on different models. As a basis, we used the astah* Sys-

Table 1: Examples of concept mapping for metamodel construction.

Source Element	Source Model	Destination Element	Destination Model	Mapping Result
Value Proposition	AI Project Canvas	Value Proposition	ML Canvas	Same
Value Proposition	AI Project Canvas	Goal	KAOS Goal Model	Similar
Safety Goal	Safety Case	Goal	KAOS Goal Model	Similar
Component	SysML	Entity	STPA	Similar
Solution	Safety Case	Countermeasure	STPA	Contribution

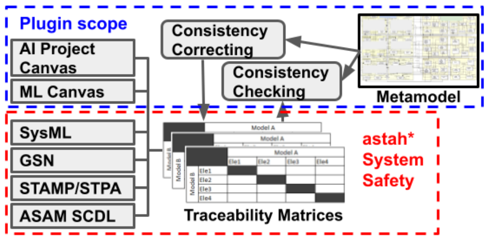


Figure 6: Overview of extended functionalities of astah* System Safety.

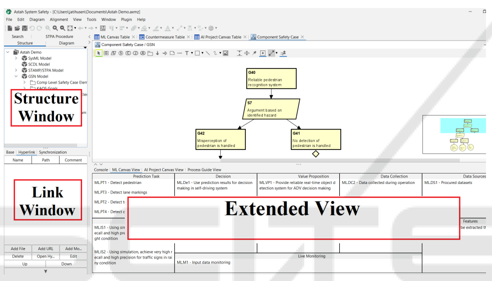


Figure 7: Overview of M^3S Support Tool.

tem Safety[1] modeling tool. The astah* System Safety provides modeling support for SysML, which can be used for architectural diagrams, Goal Structuring Notion (GSN) for KAOS and Safety Case Analysis, and STAMP/STPA. Links between different models are also facilitated partially between SysML and GSN. We extend these basic functionalities as a plugin to transform the astah* System Safety to fit the M^3S framework. Figure 6 depicts the overall concept to extend the functionality of the astah* System Safety, while Fig. 7 shows our implementation.

We implemented support for the ML and AI Project Canvas using the extended view support of the astah* System Safety. The data is an extension of the existing SysML requirement element to ensure that existing features of astah* System Safety also work in the ML and AI Project Canvas elements. We also implemented simple consistency checking as a blocker when users tried to add an incorrect connection between elements. Figure 8 shows a snapshot of consistency checking between an ML model and safety case elements. Further development and evaluation for the tool will facilitate connection types of the links, loading and editing metamodel information,and recommendation of possible links of an element based on a metamodel.

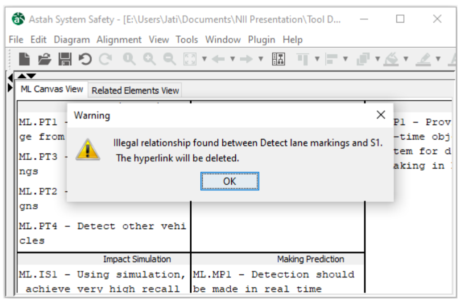
[1]https://astah.net/products/astah-system-safety/

4 EVALUATION

4.1 Case Study

To answer RQ1, a case study assessed whether the framework works as intended when implemented in a real-world case of sign detection necessary for an autonomous driving vehicle (ADV). Because a reliable source is necessary to describe ADV operation conditions to design the proper case, we employed the Japan Automobile Manufacturers Association Automated Driving Safety Evaluation Framework to generate the driving conditions for previously described environmental conditions (Japan Automobile Manufacturers Association, 2021). The case is limited to a highway setting for three environmental conditions: day, night, and rain. Additionally, we limited the ADV operation to lane-keeping maneuvers.

Figure 9 shows the M^3S version of the motivating example generated during the case study. Comparing it with the metamodel shows that the impact simulation on ML Canvas is associated with the solution for the safety case. We can deduce that the process revises existing models based on the result of the safety analysis because the safety case is developed with more information. Hence, the impact analysis can be updated to follow the solution shown in Fig. 9.


Astah System Safety - [E:\Users\Jati\Documents\NII Presentation\Tool D...
File Edit Diagram Alignment View Tools Window Plugin Help
ML Canvas View
Related Elements View
Warning
OK
Impact Simulation
Making Prediction

Illegal relationship found between Detect lane markings and S1.
The hyperlink will be deleted.

```
ML.PT1 -          P1 - Prov
ge from           -time obj
ML.PT3 -          tem for d
ngs               aking in
ML.PT2 -
gns
ML.PT4 - Detect other vehi
cles
ML.IS1 - Using simulation, ML.MP1 - Detection should
achieve very high recall   be made in real time
```

Figure 8: Example of consistency checking for ML model elements in the tool prototype.

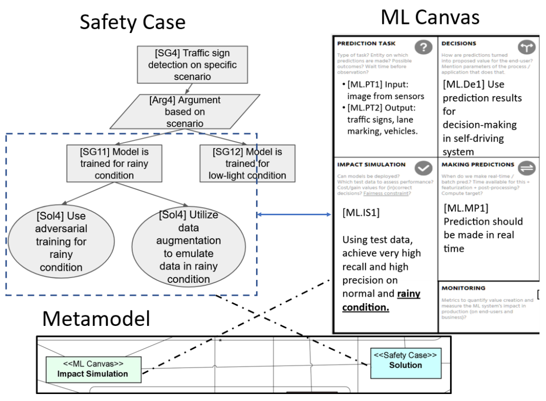

Safety Case
ML Canvas
[SG4] Traffic sign detection on specific scenario
[Arg4] Argument based on scenario
[SG11] Model is trained for rainy condition
[SG12] Model is trained for low-light condition
[Sol4] Use adversarial training for rainy condition
[Sol4] Utilize data augmentation to emulate data in rainy condition
PREDICTION TASK
• [ML.PT1] Input: image from sensors
• [ML.PT2] Output: traffic signs, lane marking, vehicles.
DECISIONS
[ML.De1] Use prediction results for decision-making in self-driving system
IMPACT SIMULATION
[ML.IS1]
Using test data, achieve very high recall and high precision on normal and rainy condition.
MAKING PREDICTIONS
[ML.MP1] Prediction should be made in real time
MONITORING
Metamodel
<<ML Canvas>> Impact Simulation
<<Safety Case>> Solution

Figure 9: Snippet of Case Study.

Table 2: Number of inconsistencies found by both groups.

Group	Inconsistencies Found
Experiment Group	9
Control Group	12

4.2 Controlled Experiment

To answer RQ2, an experiment evaluated the framework's usability to detect inconsistencies between different models. In this experiment, the participants were tasked to find inconsistencies between models developed by our frameworks and explain the rationale for marking elements as inconsistent. This experiment demonstrates whether the participants can find inconsistencies systematically. Another team of participants was given the same task except that they used the model to find inconsistencies in natural language specifications on exact requirements. The experiment was conducted without the support tool to level the capabilities of both groups in moving between models or specifications.

Six participants were divided equally into the experimental and control groups. Both groups were briefed on the task. Then each group was assigned a narrator, who explained the models or specifications they needed to work on. Each member of the group wrote down all identified inconsistencies. Afterwards, members discussed their findings to generate a list of inconsistencies that the group agreed upon.

Table 2 shows the number of inconsistencies found by groups and the distribution of types of consistencies. Based only on the number of consistencies, the control group found significantly more inconsistencies than the experiment group. Thus, the control group was more successful in this task than the experimental group.

Table 3 summarizes the distribution of the rationales behind the inconsistencies. The control group identified inconsistencies based solely on experience and intuition rather than comparing different specifications. In contrast, the experiment group found inconsistencies mostly by comparing elements between models as only 25% of their findings were made using intuition and experience. The difference between the two groups is extreme.

Table 3: Distribution of rationale used per group.

Group	Model Inconsistencies	Experience & Intuition
Experiment Group	75%	25%
Control Group	0%	100%

4.3 Answers to Research Questions

For RQ1, the metamodel is an effective tool to guide traceability. The case study showed that following the relationship realizes a more systematic approach to generate each element. Following the metamodel allows the usage of existing elements to act as a basis for deciding the content of elements on other models.

For RQ2, some support for expressing the links based on the metamodel may be needed. Although connections between elements can be made by comparing the elements with the metamodel, the process is not intuitive and can be time consuming. Employing a tool, which represents the link in the metamodel in an effective manner, should facilitate efficient usage of the framework.

For RQ3, the current state of the M^3S framework is inadequate in terms of ease of detecting inconsistencies. The experiment revealed that the control group found 50% more inconsistencies than the experimental group. The experimental group had issues navigating and processing the information. This finding suggests that a support tool is necessary to facilitate the use of this framework.

For RQ4, the enormous difference between the rationales indicates the answer. The experiment group paid more attention to the connection of information inside different elements. In contrast, the control group showed an unwanted ignorance toward the differences between requirements. These results suggest that the M^3S framework facilitates a systematic approach to ensuring consistency.

4.4 Threat to Validity

There are several threats to the validity. First, the study involved a small number of participants because the experiment focused on practitioners as the framework's users. Hence, those without prior experience in developing ML could not be included. Second, each participant had differing abilities as well as familiarity with models, specifications, and general ML development. We tried to minimize this threat by selecting participants with similar expertise and experience. Additionally, we randomized the groups to minimize the threat.

5 CONCLUSION AND FUTURE WORKS

This paper proposes and evaluates the Multi-View Modeling for ML System (M^3S) framework. The case study and experiment revealed several vital findings. On the positive side, the framework facilitates the traceable multi-view approach to analyze safety-critical ML systems. However, efficient utilization of the framework requires a support tool for decision-making of the solutions.

In the future, we plan to explore several different directions. We will continue developing and evaluating the support tool. We want to explore the possibility of creating a guide by extracting existing solutions into a catalog and extend the framework's traceability into the ML model's training pipeline.

ACKNOWLEDGEMENTS

This work was supported by JST-Mirai Program grant Number JPMJMI20B8, JST SPRING grant Number JPMJSP2128, JSPS JPJSBP 120209936, and JSPS KAKENHI 21KK0179.

REFERENCES

Chuprina, T., Méndez, D., and Wnuk, K. (2021). Towards artefact-based requirements engineering for data-centric systems. In *Joint Proceedings of REFSQ 2021 Workshops, OpenRE, Poster and Tools Track, and Doctoral Symposium co-located with the 27th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2021), Essen, Germany, April 12, 2021*, volume 2857. CEUR-WS.org.

Dorard, L. (2015). Machine learning canvas.

El Hamlaoui, M., Bennani, S., Nassar, M., Ebersold, S., and 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, page 180–191, Setubal, PRT. SCITEPRESS - Science and Technology Publications, Lda.

Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., and Habli, I. (2021). Guidance on the assurance of machine learning in autonomous systems (amlas). *CoRR*, abs/2102.01564.

Husen, J. H., Tun, H., Yoshioka, N., Washizaki, H., and Fukazawa, Y. (2021). Goal-oriented machine learning-based component development process. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 643–644. IEEE Computer Society.

Husen, J. H., Washizaki, H., Tun, H., Yoshioka, N., Fukazawa, Y., and Takeuchi, H. (2022). Traceable business-to-safety analysis framework for safety-critical machine learning systems. In *2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 50–51. IEEE Computer Society.

Ishikawa, F. and Matsuno, Y. (2020). Evidence-driven requirements engineering for uncertainty of machine learning-based systems. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 346–351. IEEE Computer Society.

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

Leveson, N. G. (2012). *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press.

Nalchigar, S. and Yu, E. (2018). Business-driven data analytics: A conceptual modeling framework. *Data & Knowledge Engineering*, 117:359–372.

National Institute of Advanced Industrial Science and Technology (2022). Machine learning quality management guideline, 2nd english edition. Technical report, Digital Architecture Research Center / Cyber Physical Security Research Center / Artificial Intelligence Research Center, Digiarc-TR-2022-01 / CPSEC-TR-2022002.

Pereira, A. and Thomas, C. (2020). Challenges of machine learning applied to safety-critical cyber-physical systems. *Machine Learning and Knowledge Extraction*, 2(4):579–602.

Thiée, L.-W. (2021). A systematic literature review of machine learning canvases. In *INFORMATIK 2021*, pages 1221–1235. Gesellschaft für Informatik, Bonn.

Vogelsang, A. and Borg, M. (2019). Requirements engineering for machine learning: Perspectives from data scientists. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 245–251.

Yoshioka, N., Husen, J. H., Tun, H., Chen, Z., Washizaki, H., and Fukazawa, Y. (2021). Landscape of requirements engineering for machine learning-based ai systems. In *2021 28th Asia-Pacific Software Engineering Conference Workshops (APSEC Workshops)*, pages 5–8. IEEE Computer Society.

Zickert, F. Evaluation of the goal-oriented requirements engineering method kaos recommended citation evaluation of the goal-oriented requirements engineering method kaos. In *AMCIS 2010 Proceedings*.