Pattern Application Support Framework in Machine Learning Reliability Solution Patterns

TAKUMI AYUKAWA, Waseda University
JATI H. HUSEN, Waseda University
NOBUKAZU YOSHIOKA, Waseda University
HIRONORI WASHIZAKI, Waseda University
NAOYASU UBAYASHI, Waseda University

The Goal Structuring Notation, which models the structure of the argument and clearly and visually shows the relationships among its elements, is a valuable tool for demonstrating the reliability of machine learning systems, especially in critical areas such as autonomous vehicles. Although Machine Learning Reliability Solution Patterns can improve the reliability and efficiency of these systems, there is a risk of misapplication due to human error, particularly when engineers lack deep expertise in machine learning. To address this issue, we propose a framework designed to support the correct application of these patterns. The tool automatically incorporates patterns into user-created diagrams to apply patterns without human error. We validate the framework's effectiveness through a case study, demonstrating its practical utility in helping engineers accurately apply these patterns.

Categories and Subject Descriptors: D.2.10 [**Software Engineering**] Design; I.2.0 [**Artificial Intelligence**] General

General Terms: Software engineering

Additional Key Words and Phrases: Design patterns, Machine learning, Assurance cases

ACM Reference Format:

Ayukawa, T. and Husen, H. and Yoshioka, N. and Washizaki, H. and Ubayashi, N. 2025. Machine Learning Application Argument Patterns: A Catalog for Building Assurance Cases. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 22 (October 2015), 14 pages.

1. INTRODUCTION

Machine learning systems must argue reliability based on requirements and values in domains where consistency and dependability are critical. Domains demanding reliable machine learning systems include autonomous vehicles and medical diagnostics. To guarantee reliability, it must be considered from the design stage. However, systems

This paper is a revised version of a Japanese paper previously presented at the AsianPLoP 2024 conference.

This work was supported by JST-Mirai program grant number JPMJMI20B8 and JSPS KAKENHI program grant numbers 21KK0179 and 23K18470.

Author's address: Takumi Ayukawa, Washizaki and Ubayashi Laboratory, Room 0503, Building 63, Waseda University, 3-4-1 Ohkubo, Shinjuku-ku, 169-8555 Tokyo, Japan; email: ayutaku@fuji.waseda.jp; Jati H. Husen, Washizaki and Ubayashi Laboratory, Room 0503, Building 63, Waseda University, 3-4-1 Ohkubo, Shinjuku-ku, 169-8555 Tokyo, Japan; email: jatihusen@telkomuniversity.ac.id; Nobukazu Yoshioka, Washizaki and Ubayashi Laboratory, Room 0503, Building 63, Waseda University, 3-4-1 Ohkubo, Shinjuku-ku, 169-8555 Tokyo, Japan; email: nobukazu@engineerable.ai; Hironori Washizaki, Washizaki and Ubayashi Laboratory, Room 0503, Building 63, Waseda University, 3-4-1 Ohkubo, Shinjuku-ku, 169-8555 Tokyo, Japan; email: washizaki@waseda.jp; Naoyasu Ubayashi, Washizaki and Ubayashi Laboratory, Room 0503, Building 63, Waseda University, 3-4-1 Ohkubo, Shinjuku-ku, 169-8555 Tokyo, Japan; email: ubayashi@aoni.waseda.jp

must be simultaneously compatible with the technical aspects of machine learning and the requirements and values as systems become larger [Nahar et al. 2022]. For example, consider a case in supervised machine learning where data is categorized into predefine classes. Although increasing the recognition accuracy of one class may be more accurate from a technical standpoint, the accuracy of other classes may decrease. For autonomous vehicles, increasing the recognition rate of automobiles may decrease the recognition rate of humans. This may be inappropriate, especially if the human recognition rate falls below the required accuracy. Consequently, the necessary degree of improvement in accuracy must be established based on requirements and values.

A method to clearly describe the reliability argument is needed, and using patterns is effective. For example, the Goal Structuring Notation (GSN) can model the structure of the argument and clearly and visually show the relationships among its elements [Group 2021]. However, a software engineering perspective is needed to describe the argument efficiently. Specifically, it is necessary to apply patterns when arguing for the reliability of machine learning systems. A pattern is a collection of frequently recurring methods in software development [Alexander 1977][John Gamma and Vlissides 1994]. For example, Machine Learning Reliability Solution Patterns allow the realization of highly reliable and efficient development methods for machine learning systems.

Although applying patterns can effectively describe the reliability argument, the human error problem persists. Some Machine Learning Reliability Solution Patterns are complex. Because engineers are not necessarily machine learning experts, employing patterns correctly can be burdensome. Moreover, pattern application may be inefficient. Consequently, patterns may not be used correctly. In fact, our motivation to improve this situation is from first-hand experience. In our research involving a complex pattern, human error resulted in incorrectly applying a Machine Learning Reliability Solution Pattern in GSN. Due to an unconscious assumption that the pattern was correct, it took time to identify the error.

In this study, we propose a pattern application support framework to mitigate human error when applying the Machine Learning Reliability Solution Pattern and developed a plug-in. In our framework, a user selects a pattern, enters parameters that match the pattern, and presses the Apply button. Then our framework automatically applies the pattern by creating the part that was not entered. This automatic application prevents human error. Furthermore, we verify the practicality of our framework through a case study.

This paper makes the following contributions:

—Proposal of pattern application support framework

—Implementation of tools to support pattern application

—Validation through case study

The rest of this paper is organized as follows. Section 2 introduces GSN, astah* System Safety, and Machine Learning Reliability Solution Patterns. Section 3 details the proposed pattern application support framework. Section 4 confirms the practicality of our framework via a case study. Section 5 discusses the results. Section 6 presents related works. Finally, Section 7 concludes the paper with remarks for future directions.

2. BACKGROUND

2.1 GSN

GSN models the structure of an argument and clarifies the relationships among its elements [Group 2021]. GSN supports visualization of the basis for claims in an argument. Table I shows the meaning of each element, while Fig. 1 expresses each element. For example, Goal is represented by a square, whereas SupportedBy is represented by a black arrow. These combinations clarify the structure of the argumentation. Conventionally, the title of each element is uniquely defined by the initial letter of the element's name and the assignment of a number. GSN can also be used when defining safety goals and functional safety requirements for in-vehicle embedded systems [Organization 2018]. In this study, the reliability of machine learning systems is argued using GSN.

Table I. : Explanation of the Core GSN Elements and Relationships.

Name	Explanation
Goal	Specific claims that the system must meet
Strategy	Approaches and methodologies to break down Goal
Solution	Concrete evidence based on Strategy
Context	Goal and Strategy status
Justification	Logic and reasons validating of the Goal and Strategy
Assumption	Assumptions on which Goal depends
InContextOf	Link to the situation or condition to which the Goal, Strategy, or Solution applies
SupportedBy	Links to relevant elements and evidence supporting the Goal and Strategy

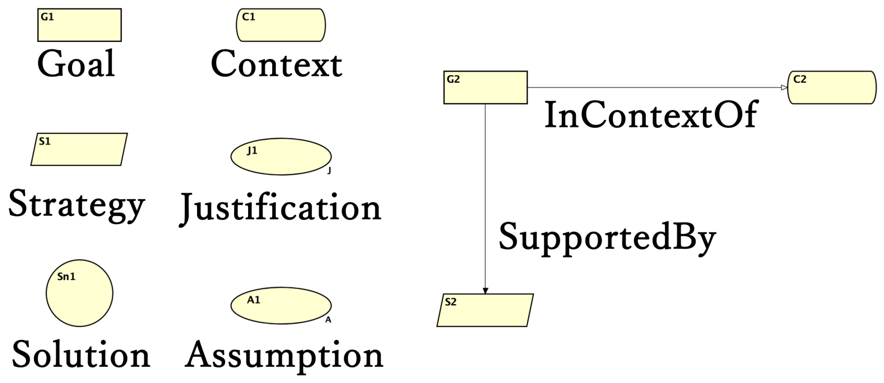


Fig. 1: Core GSN elements

2.2 astah* System Safety

Astah* is the automated visual modeling tool designed for system engineers, software engineers, and safety engineers to tackle the challenges of creating and managing software and systems [Vision 2024a]. Especially in the case of complex system development, there are issues such as ambiguity, misinterpretation, communication gap among stakeholders, inconsistent documents, and so on. Therefore, astah* assists the entire team in system development by enabling them to quickly and easily create diagrams that clearly visualize the system design. There are four major tools in astah* as shown in Table II. Since this study deals with safety critical areas, astah* System Safety is used.

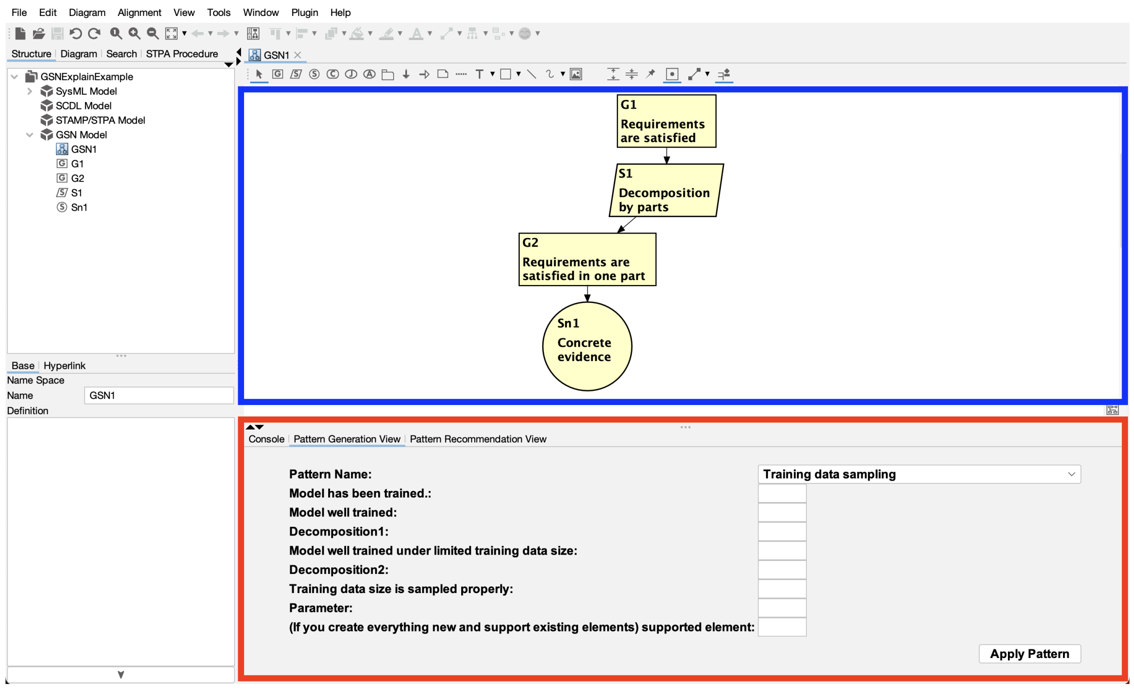
Astah* System Safety is a support tool for system analysisis a model-based systems engineering tool for safety-critical systems to support system architecture modeling, system safety assessment, and analysis [Vision 2024b]. It allows incorporating the modeling languages and techniques in Table III into a single tool. Traceability between models is ensured, and interlinking and linkage between models is possible. For example, when a model of a requirement class created in a requirement diagram of SysML is dragged and dropped to a GSN diagram,

Table II. : Explanation of astah*.

Tool name	Explanation
astah* professional	Total software design tool with UML, ERD, flowchart, DFD, and more
astah* UML	Lightweight UML diagramming tool with mind mapping abilities
astah* System Safety	Model-based systems engineering tool that includes SysML, GSN, STPA/STAMP, ASAM SCDL for safety engineers
astah* SysML	Lightweight SysML modeling tool for systems engineering

Table III. : Explanation of models in astah* System Safety.

Model name	Explanation
SysML (OMG Systems Modeling Language)	SysML is a general-purpose modeling language for modeling systems that is intended to facilitate a model-based systems engineering approach to engineer systems. This includes representing the requirements, structure, and behavior of the system, and the specification of analysis cases and verification cases used to analyze and verify the system.
UAF (Unified Architecture Framework)	UAF is based on UML, SysML and so on. It is a framework used to support enterprise architecture and systems engineering.
STAMP/STPA (Systems-Theoretic Accident Model and Processes/System-Theoretic Process Analysis)	STAMP is a safety analysis approach based on systems theory. STPA is a specific technique within STAMP used for hazard analysis and identifying potential accidents in complex systems.
GSN/D-Case (Goal Structuring Notation/Dependability Case)	GSN is a graphical notation used to create structured safety arguments. D-Case is an extension of GSN, used for creating dependability cases, which argue for the dependability of a system, including safety, reliability, and other attributes.
ASAM SCDL (Association for Standardization of Automation and Measuring Systems Safety Case Description Language)	ASAM SCDL is a language used for describing safety cases in a standardized format. It ensures that safety cases are consistent, traceable, and reusable across different projects and systems.


File Edit Diagram Alignment View Tools Window Plugin Help
Structure Diagram Search STPA Procedure
GSN1
GSNExplainExample
SysML Model
SCDL Model
STAMP/STPA Model
GSN Model
GSN1
G1
G2
S1
Sn1
Base Hyperlink
Name Space
Name GSN1
Definition
G1 Requirements are satisfied
S1 Decomposition by parts
G2 Requirements are satisfied in one part
Sn1 Concrete evidence
Console Pattern Generation View Pattern Recommendation View
Apply Pattern

Pattern Name:	Training data sampling
Model has been trained.:	
Model well trained:	
Decomposition1:	
Model well trained under limited training data size:	
Decomposition2:	
Training data size is sampled properly:	
Parameter:	
(If you create everything new and support existing elements) supported element:	

Fig. 2: UI of the astah* System Safety

it is reflected as a Goal in the GSN diagram. Then, the model can be related to each other by interlinking, so that changes in one model will be reflected in the other model as well. This ensures consistency and traceability. Not only does it allow engineers to analyze systems more efficiently, but it also can describe a GSN diagram. In this study, GSN diagram is used in an astah* System Safety. Fig. 2 shows the UI of astah* System Safety. Each element's GSN was manipulated in the window in the upper right corner of Fig. 2. In this study, we developed a plug-in for the astah* System Safety. For example, Fig. 2 contains an extended tab in the lower right corner, which allows users to input data.

2.3 Machine Learning Reliability Solution Patterns

Previously, we identified Machine Learning Reliability Solution Patterns for machine learning argumentation. Table IV shows the Machine Learning Reliability Solution Patterns. To assure reliability, these patterns must not only be considered from the design phase, but they must also be compared to the requirements and values of the machine learning system. However, some aspects depend on the training results and other factors. Since these patterns reflect the unique characteristics of machine learning, this study considers their application. Furthermore, a structural representation of these elements using GSN diagrams allows arguments to be debunked step-by-step. Hence, the problem of guaranteeing the reliability of machine learning systems can be divided into subproblems and third parties can clarify parts of the system with guaranteed reliability. Since they are patterns, these arguments can be reused. For example, the training data sampling pattern can be represented in a GSN diagram as shown in Fig. 3.

3. PROPOSED PATTERN APPLICATION SUPPORT FRAMEWORK

This section proposes the pattern application support framework to address human error when applying Machine Learning Reliability Solution Patterns and describes the overall system structure. To prevent human error in pattern

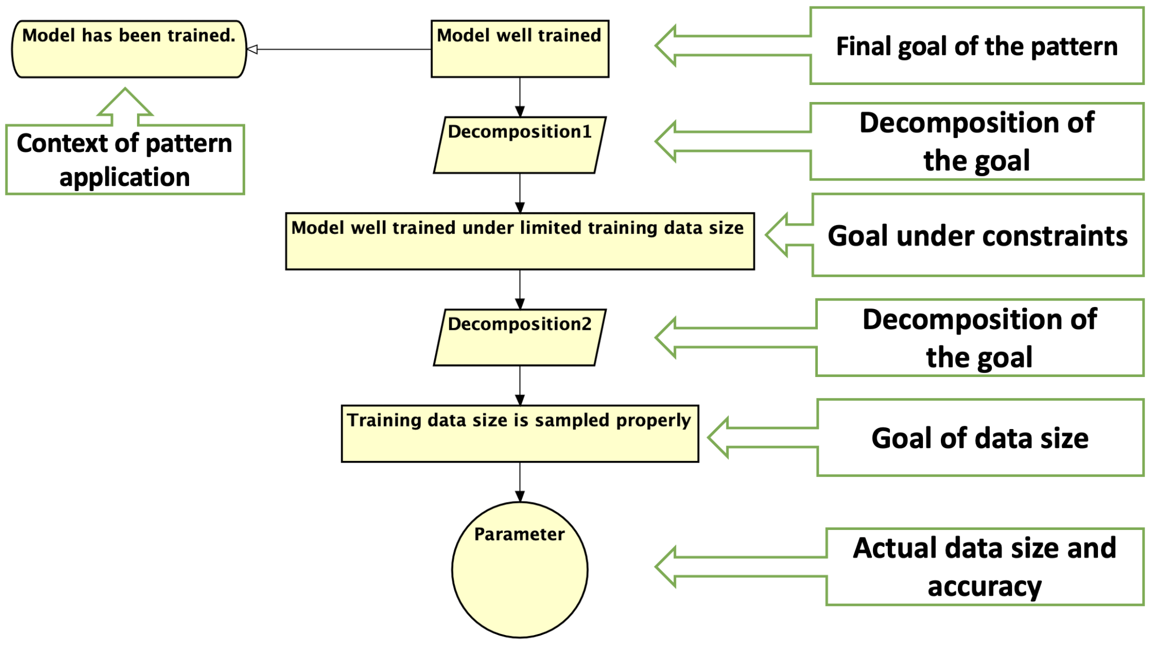

Model has been trained.
Model well trained
Final goal of the pattern
Context of pattern application
Decomposition1
Decomposition of the goal
Model well trained under limited training data size
Goal under constraints
Decomposition2
Decomposition of the goal
Training data size is sampled properly
Goal of data size
Parameter
Actual data size and accuracy

Fig. 3: GSN of the training data sampling pattern

Table IV. : Extracted Machine Learning Application Argument Patterns

ID	Pattern Name	Category	Context	Problem	Solution
P1	Selective repair pattern [Sohn et al. 2023]	Reliability	The model is already trained, and some classes are more important than others in machine learning classification.	It is impossible to analyze the cause of deficiencies or to consider corresponding countermeasures.	Machine learning repair tool
P2	Adversarial example defense pattern [Xue et al. 2020]	Security	The model is not trained and is assumed to face malicious attacks from adversarial examples.	It is necessary to respond to malicious users (control by security attacks, misrecognition of images, falsification of learning data, etc.).	Adversarial training
P3	Reprioritized accuracy pattern [Sohn et al. 2023]	Reliability	The model is already trained and concept drift has been detected.	Recognition of critical scenes cannot be guaranteed and retraining data may reduce the recognition rate.	Machine learning repair tool
P4	Training data sampling pattern [Kabkab et al. 2016]	Efficiency	The model has already been trained and large training data are unavailable.	A large amount of training and test data is needed to ensure reliability.	Data sampling method
P5	Model smoke-testing pattern [Herbold and Haar 2022]	Efficiency	The model has just been retrained and needs reverification.	The cost is high for the reverification and retesting of safety and reliability when the system is changed.	Smoke-testing method
P6	Rule-based safeguard pattern [Washizaki et al. 2020]	Reliability	The machine learning system's application domain is clearly defined.	Safe system shutdown within the warranty period cannot be guaranteed.	Rule-base safeguard function
P7	Security requirement satisfaction argument pattern [Zeroual et al. 2023]	Security	The model is already trained and verification is needed.	Security requirements must be evaluated in a data and model-driven manner.	Test data and formal verification
P8	DNN robustness case verification pattern	Security	DNNs are deployed in security-critical domains where adversarial attacks are particularly significant.	A security case based on the formal verification of robustness has yet to be verified.	Arguments over the model verification inputs, verification itself and model integration into the system as a whole

applications, the user inputs the parts that match the selected pattern. Then the proposed framework automatically applies the pattern by creating parts that were not inputted. As a concrete mechanism, Fig. 4 schematically diagrams a research system. The area in orange depicts the proposed framework. The framework consists of the following four major steps:

Step 0 Define the patterns: An expert defines the patterns. In this case, the Machine Learning Reliability Solution Patterns definition is stored.

Step 1 Select the pattern: The user selects a pattern. Then the framework accepts the selected pattern and stores the pattern name.

Step 2. Input the parameters: The framework accepts the necessary parameter inputs from the user by referring to the pattern name and pattern definition. This data is stored as pattern parameters.

Step 3: Press the Apply button: When the user presses the Apply button, the framework retrieves the GSN diagram created by the user in advance on astah* System Safety and stores it. Next, the framework creates the

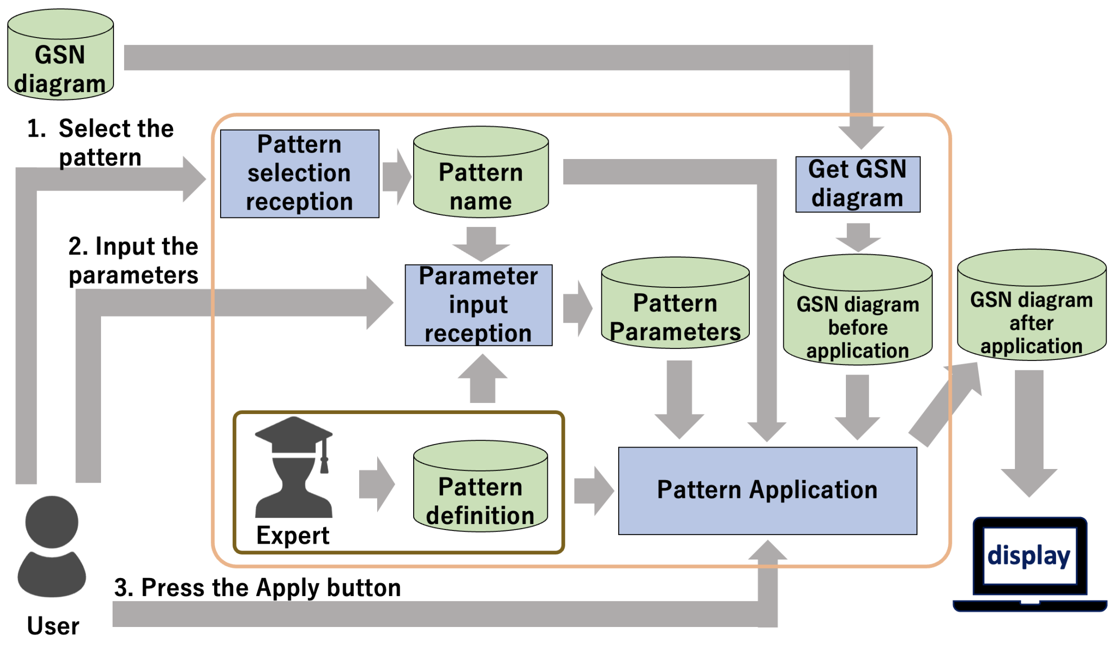


Fig. 4: Schematic diagram of a system focusing on data flow

necessary elements by referring to the pattern definition, pattern name, pattern parameters, and GSN diagram before pattern application. The data is outputted as a GSN diagram after the pattern is applied and displayed on astah* System Safety.

Below we describe a concrete example where the training data sampling pattern is applied to the model shown in Fig. 5. First, the engineer specifies the pattern shown in Fig. 6. In this case, the training data sampling pattern is selected. Then the engineer inputs the corresponding value for each pattern (Fig. 7). In this case, we assume that "Model well trained" corresponds to G1 and "Decomposition1" corresponds to S1. Finally, by pressing the Apply button, the necessary models are automatically created and merged with the diagram being created. Fig. 8 shows the model in this case. This allows the necessary Goal, Solution, Context, etc. to be automatically applied to the diagram in a way that matches the pattern. In addition, the pattern can be applied efficiently as it is integrated with the newly created diagram.

Let's consider a special case where none of the parts match the pattern and all parts must be created. In this case, to connect existing elements with SupportedBy, they should be entered in the supported element field. Fig. 5 shows an example where all the elements of the training data sampling pattern are newly created and S1 of the existing elements are connected by SupportedBy. In this case, S1 can be entered in the supported element field. Pressing the Apply button gives the pattern shown in Fig. 9.

Let's consider the case where additional variables are needed, which is a problem specific to machine learning systems. Since this study deals with Machine Learning Reliability Solution Patterns, the number of elements in some patterns changes dynamically with the number of classes. Therefore, additional variables may be required. Specifically, the Selective repair pattern, the Prioritize accuracy pattern, and the Model Smoke Testing pattern fall into this category. For example, in the Model Smoke Testing pattern, the number of elements depends on the value in the number of class field. Fig. 10 and 11 show the number of elements when the number of class field is 1 and 2, respectively. For the Model Smoke Testing pattern, the number of elements can be increased according to

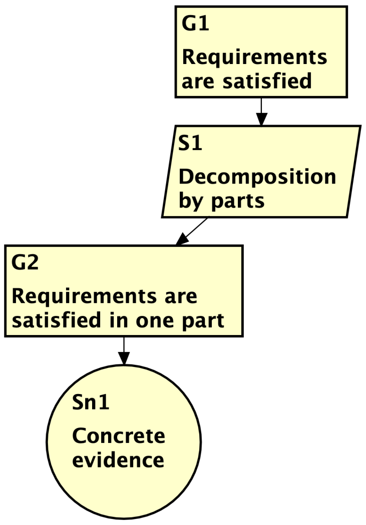
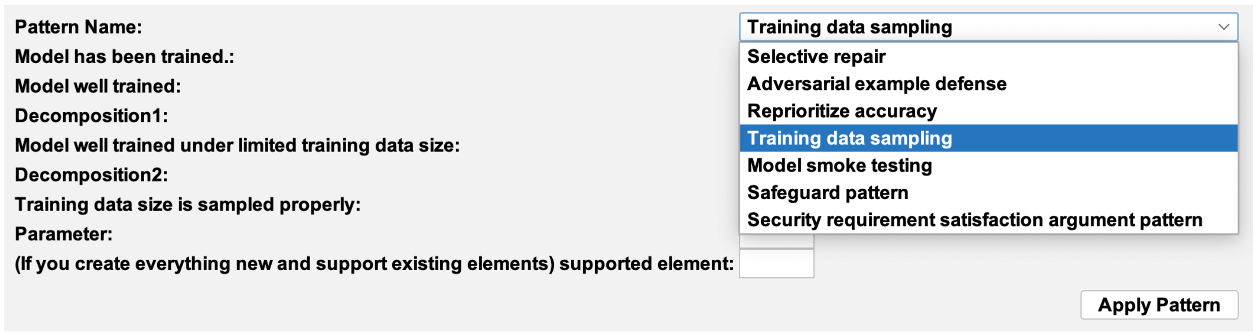
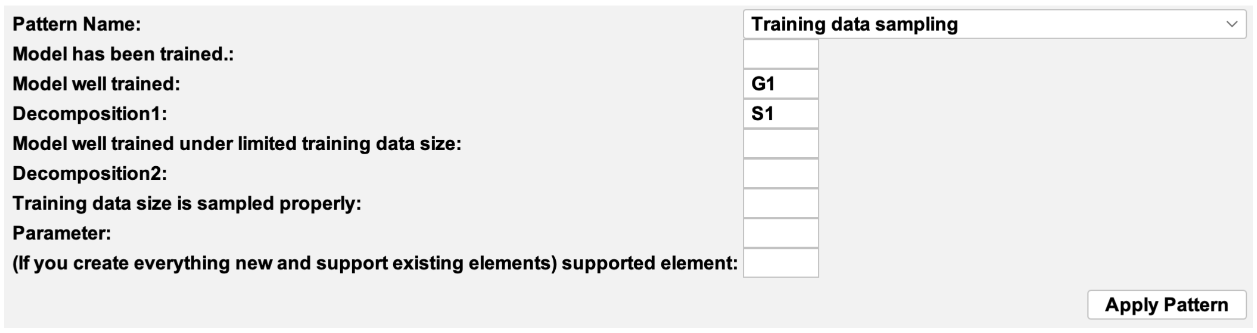

G1
Requirements are satisfied
S1
Decomposition by parts
G2
Requirements are satisfied in one part
Sn1
Concrete evidence

Fig. 5: Example of a GSN diagram



Pattern Name:	Training data sampling
Model has been trained.:	
Model well trained:	
Decomposition1:	
Model well trained under limited training data size:	
Decomposition2:	
Training data size is sampled properly:	
Parameter:	
(If you create everything new and support existing elements) supported element:	

Fig. 6: Pattern Selection UI



Pattern Name:	Training data sampling
Model has been trained.:	
Model well trained:	G1
Decomposition1:	S1
Model well trained under limited training data size:	
Decomposition2:	
Training data size is sampled properly:	
Parameter:	
(If you create everything new and support existing elements) supported element:	

Fig. 7: Element Selection UI

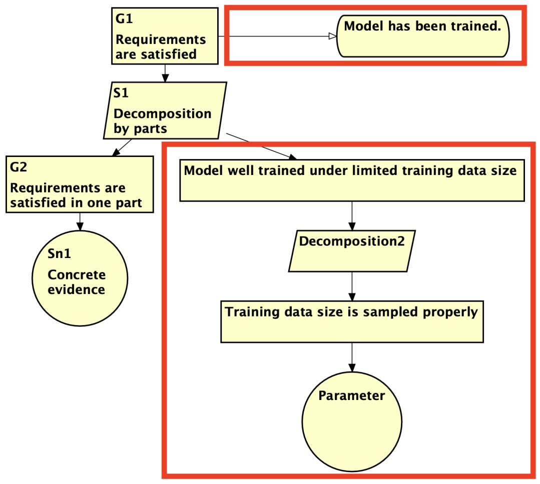

G1
Requirements are satisfied
Model has been trained.
S1
Decomposition by parts
G2
Requirements are satisfied in one part
Model well trained under limited training data size
Decomposition2
Sn1
Concrete evidence
Training data size is sampled properly
Parameter

Fig. 8: Example of a GSN diagram after applying the training data sampling pattern

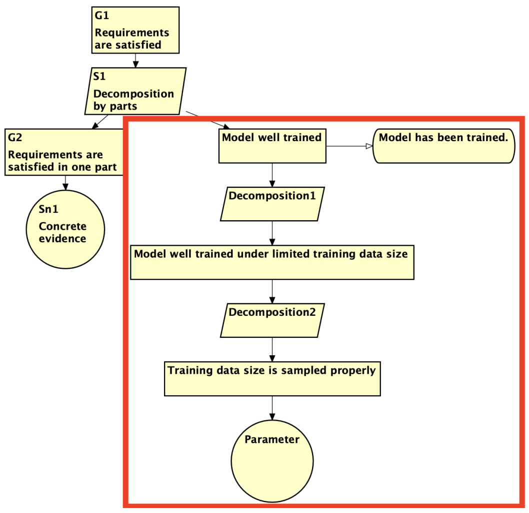

G1
Requirements are satisfied
S1
Decomposition by parts
G2
Requirements are satisfied in one part
Model well trained
Model has been trained.
Sn1
Concrete evidence
Decomposition1
Model well trained under limited training data size
Decomposition2
Training data size is sampled properly
Parameter

Fig. 9: Example of a GSN diagram for the training data sampling pattern application when all parameters are newly created

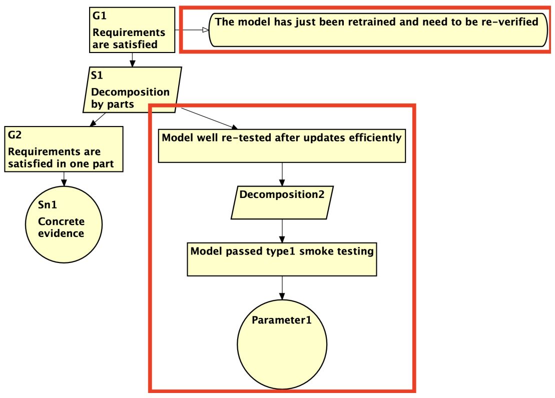

G1
Requirements are satisfied
The model has just been retrained and need to be re-verified
S1
Decomposition by parts
G2
Requirements are satisfied in one part
Model well re-tested after updates efficiently
Decomposition2
Sn1
Concrete evidence
Model passed type1 smoke testing
Parameter1

Fig. 10: Applying the Model Smoke Testing pattern when the number of classes is 1

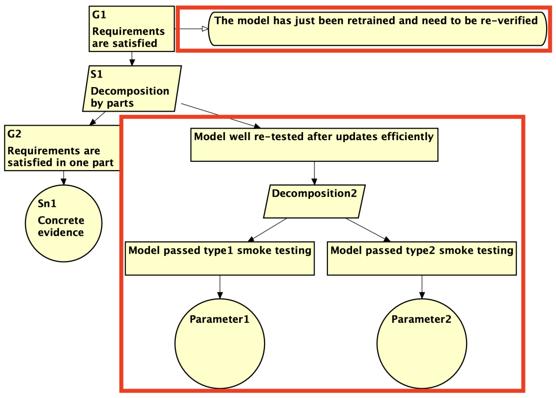

G1
Requirements are satisfied
The model has just been retrained and need to be re-verified
S1
Decomposition by parts
G2
Requirements are satisfied in one part
Model well re-tested after updates efficiently
Decomposition2
Sn1
Concrete evidence
Model passed type1 smoke testing
Model passed type2 smoke testing
Parameter1
Parameter2

Fig. 11: Applying the Model Smoke Testing pattern when the number of classes is 2

the number of test types. In this way, patterns with dynamically changing number of elements, which is unique to machine learning systems, are supported by setting variables.

4. CASE STUDY

A case study demonstrates the practicality of the proposed framework. Fig. 12 shows the GSN diagram of a machine learning system to classify traffic signs in the process of creation [Husen et al. 2024]. The purpose of this GSN diagram is to argue by argument and evidence that the top goal is indeed established. The top goals in this case are to provide a reliable traffic sign classification system for level 3 automated driving vehicle (ADV) system, as shown in G30. A level 3 ADV system is a conditional driving automation system in which the system can perform all driving operations under limited conditions. The driver must be able to return to driving at any time if requested by the system. To achieve this goal, G30 is divided into S1 and S2, and G30 is decomposed into two aspects:

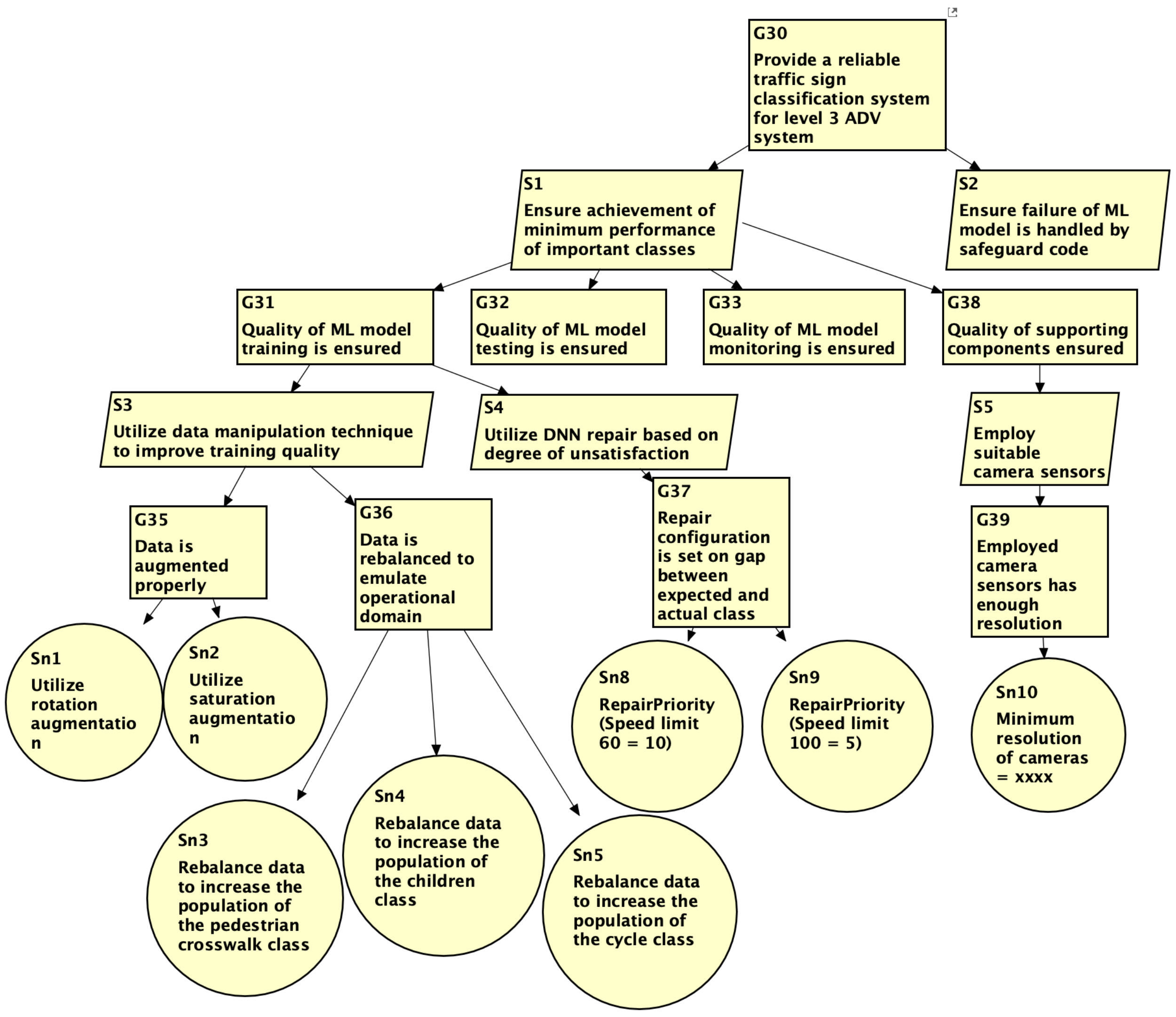
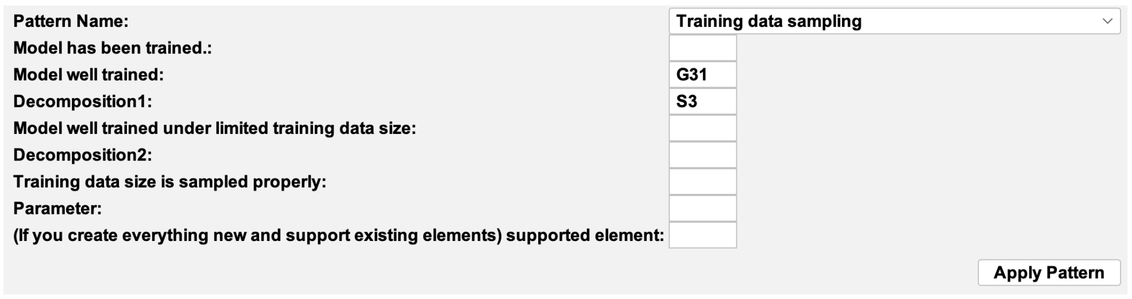

G30
Provide a reliable traffic sign classification system for level 3 ADV system
S1
Ensure achievement of minimum performance of important classes
S2
Ensure failure of ML model is handled by safeguard code
G31
Quality of ML model training is ensured
G32
Quality of ML model testing is ensured
G33
Quality of ML model monitoring is ensured
G38
Quality of supporting components ensured
S3
Utilize data manipulation technique to improve training quality
S4
Utilize DNN repair based on degree of unsatisfaction
S5
Employ suitable camera sensors
G35
Data is augmented properly
G36
Data is rebalanced to emulate operational domain
G37
Repair configuration is set on gap between expected and actual class
G39
Employed camera sensors has enough resolution
Sn1
Utilize rotation augmentation
Sn2
Utilize saturation augmentation
Sn3
Rebalance data to increase the population of the pedestrian crosswalk class
Sn4
Rebalance data to increase the population of the children class
Sn5
Rebalance data to increase the population of the cycle class
Sn8
RepairPriority (Speed limit 60 = 10)
Sn9
RepairPriority (Speed limit 100 = 5)
Sn10
Minimum resolution of cameras = xxxx

Fig. 12: Case study

guaranteeing performance in critical classes and normal shutdown in case of unexpected behavior. Satisfying these two aspects guarantees reliability. For example, according to S1, which says to ensure the achievement of minimum performance of important classes, it is decomposed into the subgoals G31, G32, G33, and G38. This is broken down by the goal of guaranteed quality in each of the phases of machine learning training, testing, and monitoring, as well as the goal of guaranteed quality for the input. For example, to satisfy the quality of supporting components ensured in G38, the resolution of the camera is discussed in S5 and G39, and the evidence of the minimum resolution of the camera in Sn10 confirms that G38 is satisfied. Also, G31, which says that quality of ML model training is ensured, is decomposed into the perspective of machine learning data and the perspective of model preparation, as in the S3 and S4. After S3, the machine learning training data is discussed in G35 and G36, arguing that the data quality is adequate due to the reinforcement of the data in Sn1-5 and evidence of rebalancing



Pattern Name:	Training data sampling
Model has been trained.:	
Model well trained:	G31
Decomposition1:	S3
Model well trained under limited training data size:	
Decomposition2:	
Training data size is sampled properly:	
Parameter:	
(If you create everything new and support existing elements) supported element:	

Fig. 13: Input parameters of the training data sampling pattern

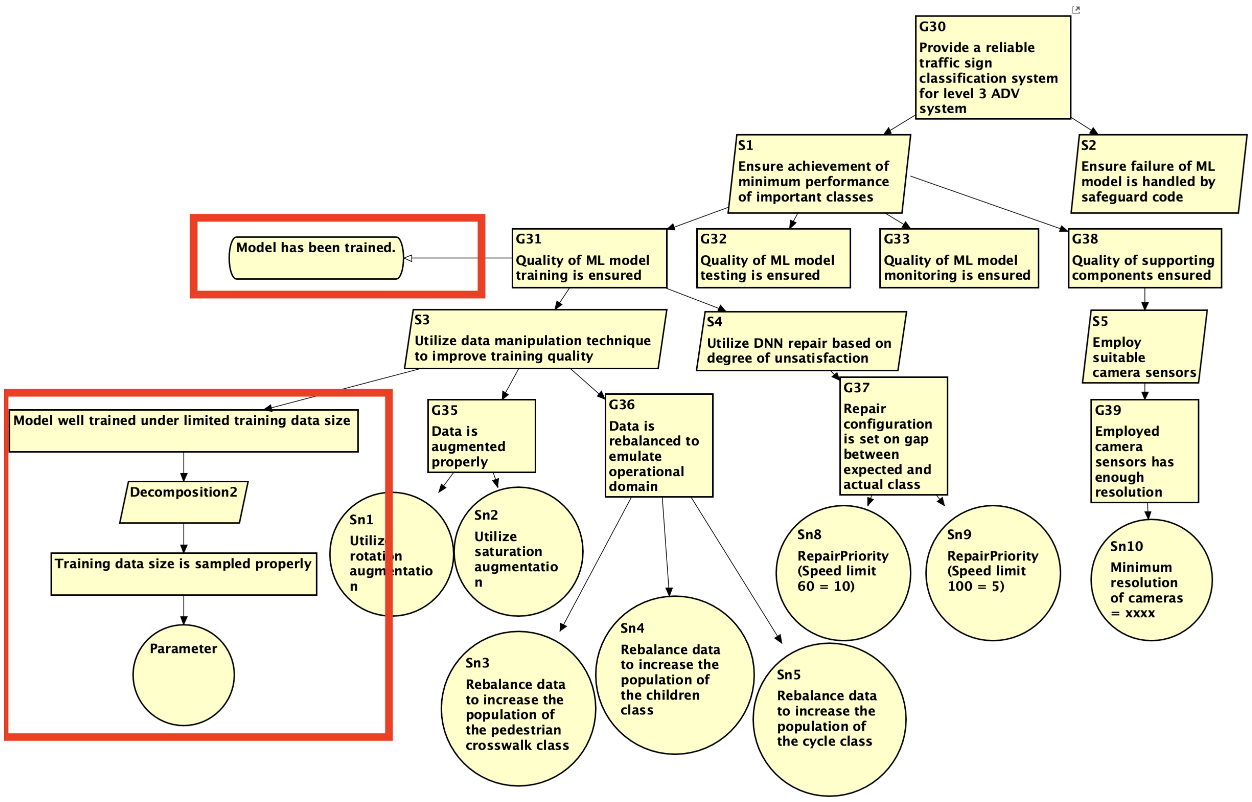

G30 Provide a reliable traffic sign classification system for level 3 ADV system
S1 Ensure achievement of minimum performance of important classes
S2 Ensure failure of ML model is handled by safeguard code
Model has been trained.
G31 Quality of ML model training is ensured
G32 Quality of ML model testing is ensured
G33 Quality of ML model monitoring is ensured
G38 Quality of supporting components ensured
S3 Utilize data manipulation technique to improve training quality
S4 Utilize DNN repair based on degree of unsatisfaction
S5 Employ suitable camera sensors
Model well trained under limited training data size
Decomposition2
Training data size is sampled properly
Parameter
G35 Data is augmented properly
G36 Data is rebalanced to emulate operational domain
G37 Repair configuration is set on gap between expected and actual class
G39 Employed camera sensors has enough resolution
Sn1 Utiliz rotation augmentatio n
Sn2 Utilize saturation augmentatio n
Sn3 Rebalance data to increase the population of the pedestrian crosswalk class
Sn4 Rebalance data to increase the population of the children class
Sn5 Rebalance data to increase the population of the cycle class
Sn8 RepairPriority (Speed limit 60 = 10)
Sn9 RepairPriority (Speed limit 100 = 5)
Sn10 Minimum resolution of cameras = xxxx

Fig. 14: Application of the training data sampling pattern in the case study

the amount of data in important classes. Also, after S4, repair configurations are discussed in G37 and prioritized by raising the repair priority of traffic signs with a speed limit of 60 to 10 in Sn8 and setting the repair priority of traffic signs with a speed limit of 100 to 5 in Sn9. The above discusses that G30 is satisfied.

Here, we consider the case where large amounts of training and test data are necessary to ensure reliability. We assume that the machine learning model has already been trained and the limitation is that large training data is not available. In the case study, an engineer who is not a machine learning expert applies the training data sampling pattern in the GSN diagram shown in Fig. 12. However, the pattern may not be accurately applied. Furthermore, issues may arise when manually applying the pattern due to human error.

The proposed framework solves this problem. Fig. 13 depicts the input. First, the engineer selects the training data sampling pattern and inputs known parameters. In this case, an engineer select training data sampling pattern as Pattern Name like Fig. 13 the Statement of G31 is "Quality of ML model training is ensured", matches the Goal of "Model well trained" in this pattern because the meaning is same. Also, the Statement of S3 is "Utilize data manipulation technique to improve training quality", which matches the goal of Decomposition1 of this pattern because the trainig data sampling pattern is a pattern related to data manipulation, so the contents of S3 also related to data manipulation and thus match. Therefore, an engineer input G31 in the "Model well trained" and S3 in the "Decomposition1". No inputs are entered for the other parts because they are not matched.Instead, these are created by the proposed framework. If the Apply button is pressed in this state, the training data sampling pattern will be applied (Fig. 14). In this example, the information in Fig. 13 shows that the user-created diagrams do not include anything other than G31 and S3, so new elements are automatically created. They are then automatically connected to G31 and S3 by links. This allows for reliability assurance even when large amounts of training data are used due to the added discussion of sampling of training data.

Hence, this framework allows engineers who are not experts in machine learning to use the knowledge of experts and apply the pattern accurately. In addition, the automatic application of the pattern resolves the issue of human error. The above results confirm the practicality of pattern application in the case study.

5. DISCUSSION

Although we believe that the application of patterns in this study can reduce human error, further innovations are necessary. First, the automatic application of patterns should mitigate human error. For example, the pattern in this case study is correctly applied because SupportedBy links do not bind to unintended elements. However, the possibility of human error remains. If an element's value is misspelled or misread, it could cause the pattern to be applied to unintended areas. Two measures can remedy this issue. One is to improve the UI. A dropdown list of inputs would resolve potential spelling errors. The other is to develop a tool to detect patterns after they have been applied. If a pattern is incorrectly applied, a warning message could be displayed to prevent errors.

This framework has two limitations: the need for experts and prior knowledge. Experts are essential because the proposed framework requires their skills to define patterns. Additionally, automation remains a challenge, limiting the available patterns. Second, prior knowledge is necessary for the engineer to select the pattern. Our framework may be insufficient for engineers without prior knowledge of patterns.

In the future, a pattern selection support tool should be developed. The pattern selection tool can adopt two mechanisms: graph pattern matching and GraphPrompts, which is a large language model. For graph pattern matching, experts define the conditions to recommend patterns in advance. Then the support tool recommends patterns when the pre-defined conditions are met. For example, consider the case of recommending a training data sampling pattern. The condition is defined as a case where the element of a GSN is “goal” and the words “model” and “train” are included in the sentence. In this example, the condition “Quality of ML model training is ensured” in G31, which matches the condition in Fig. 12. Therefore, the training data sampling pattern can be recommended. In contrast, GraphPrompts is a pattern for handling graph structures in large language models. Experts define prompts that include information about the patterns. Then patterns are recommended by matching the output. However, the possibility of hallucination must be considered in large language models [Ji et al. 2023]. It is expected that a pattern selection support tool employing these methods should enhance pattern applications when the engineer lacks prior knowledge.

6. RELATED WORK

Husen et al. [Husen et al. 2023] proposed a Multi-View Modeling Framework for ML Systems (M^{3}S) to address the issue of referencing different views and models during machine learning system development. Similar to our study, their research focused on designing machine learning systems and developing a plug-in by astah* System Safety. M^{3}S allows different models such as GSN diagrams, SysML diagrams, and STAMP/STPA diagrams to be related. Their study effectively developed AI Project Canvas and ML Canvas. One difference between these studies is the main objective. Our study focused on pattern applications, while their study aimed to align with different models. Additionally, our study can be located in the Safety and Safety Case part of the M^{3}S process.

Lakshmanan et al. [Lakshmanan et al. 2020] compiled a catalog of design patterns in machine learning. Their book summarizes 30 design patterns that can be used in each step of the development process of an entire machine learning system. Their book also mentioned the connections between the patterns. This reference was useful because it organized the design patterns in an easy-to-use format. However, the book aimed to systemize design patterns, whereas our study strives to apply patterns and develop support tools. Additionally, their book described general problems and solutions for machine learning systems, while our study uses GSN diagrams.

Shiroma et al. [Shiroma et al. 2010] discussed pattern applications. Design patterns can be applied by specifying the pattern and the corresponding part of the design pattern. In addition, a function to detect design patterns was developed. Their study successfully applied multiple related design patterns by marking them with UML stereotypes and accurately implemented multiple design patterns by referencing them. However, their target differs from our study. They handled design patterns for security in model-driven development and not machine learning patterns. Another difference is that their study used UML diagrams instead of GSN diagrams.

Yu et al. [Yu et al. 2017] reported that tool development automated pattern detection and resolution with respect to security patterns in goal models. They developed and applied ATLAS Transformation Language tools to security patterns in model-driven development in requirements engineering. In particular, we found the function of detecting conformity of models with design patterns to be useful. However, the type of patterns differs from our study. Their research dealt with security design patterns for model-driven development, while our study investigates machine learning patterns.

7. CONCLUSION AND FUTURE WORK

We proposed a framework and implemented a tool to support the application of Machine Learning Reliability Solution Patterns. Then we validated our approach via a case study. Specifically, we applied a plug-in in astah* System Safety to automatically implement the patterns. We expect that our pattern application will help engineers correctly apply the patterns. In conclusion, we have provided a pattern application support framework, which addresses human error when applying the Machine Learning Reliability Solution Pattern.

In the future we have three prospects. The first is to develop a pattern recommendation tool. We plan to employ graph pattern matching and GraphPrompts to realize a more effective pattern selection support tool. To improve our methodology, we plan to refine the UI and incorporate a warning notification system if the pattern is incorrectly applied. Finally, we will conduct evaluation experiments to validate the effectiveness of our framework by the efficiency and accuracy of pattern applications with and without the use of support tools.

Acknowledgment

We would like to thank Mary Tedeschi for shepherding this paper and Writer's Workshop participants. We would also like to express my gratitude to Martin Gutsche, especially for his invaluable suggestions regarding the DNN Robustness Case Verification Pattern.

REFERENCES

Christopher Alexander. 1977. *A pattern language: towns, buildings, construction.* Oxford university press.

The Assurance Case Working Group. 2021. Goal Structuring Notation Community Standard Version 3. `https://scsc.uk/r141C:1?t=1`. (2021). (Accessed on January 14, 2024).

Steffen Herbold and Tobias Haar. 2022. Smoke testing for machine learning: simple tests to discover severe bugs. *Empirical Software Engineering* 27, 2 (2022), 45.

Jati H Husen, Hironori Washizaki, Jomphon Runpakprakun, Nobukazu Yoshioka, Hnin Thandar Tun, Yoshiaki Fukazawa, and Hironori Takeuchi. 2024. Integrated multi-view modeling for reliable machine learning-intensive software engineering. *Software Quality Journal* 32, 3 (2024), 1239–1285.

Jati H Husen, Hironori Washizaki, Nobukazu Yoshioka, Hnin Thandar Tun, Yoshiaki Fukazawa, and Hironori Takeuchi. 2023. Metamodel-Based Multi-View Modeling Framework for Machine Learning Systems.. In *MODELSWARD*. 194–201.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

Richard Johnson John Gamma, Erich Helm and Ralph Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education. https://books.google.co.jp/books?id=6oHuKQe3TjQC

Maya Kabkab, Azadeh Alavi, and Rama Chellappa. 2016. Dcnns on a diet: Sampling strategies for reducing the training set size. *arXiv preprint arXiv:1606.04232* (2016).

Valliappa Lakshmanan, Sara Robinson, and Michael Munn. 2020. *Machine learning design patterns*. O'Reilly Media.

Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process. In *Proceedings of the 44th International Conference on Software Engineering*. 413–425.

International Standardization Organization. 2018. ISO 26262-1:2018(en) Road vehicles - Functional safety - Part 1: Vocabulary. https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en. (2018). (Accessed on January 14, 2024).

Yuki Shiroma, Hironori Washizaki, Yoshiaki Fukazawa, Atsuto Kubo, and Nobukazu Yoshioka. 2010. Model-driven security patterns application based on dependences among patterns. In *2010 International conference on availability, reliability and security*. IEEE, 555–559.

Jeongju Sohn, Sungmin Kang, and Shin Yoo. 2023. Arachne: Search-Based Repair of Deep Neural Networks. *ACM Transactions on Software Engineering and Methodology* 32, 4 (2023), 1–26.

Change Vision. 2024a. astah*. https://astah.net. (2024). (Accessed on January 14, 2024).

Change Vision. 2024b. astah* System Safety. https://astah.net/products/astah-system-safety/. (2024). (Accessed on January 14, 2024).

Hironori Washizaki, Foutse Khomh, Yann-Gaël Guéhéneuc, Hironori Takeuchi, Satoshi Okuda, Naotake Natori, and Naohisa Shioura. 2020. Software engineering patterns for machine learning applications (sep4mla) part 2. In *Proceedings of the 27th conference on pattern languages of programs*. 1–10.

Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. 2020. Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access* 8 (2020), 74720–74742.

Yijun Yu, Haruhiko Kaiya, Nobukazu Yoshioka, Zhenjiang Hu, Hironori Washizaki, Yingfei Xiong, and Amin Hosseinian-Far. 2017. Goal modelling for security problem matching and pattern enforcement. *International Journal of Secure Software Engineering (IJSSE)* 8, 3 (2017), 42–57.

Marwa Zeroual, Brahim Hamid, Morayo Adedjouma, and Jason Jaskolka. 2023. Security Argument Patterns for Deep Neural Network Development. https://drive.google.com/file/d/1j1FreLDMqygleJR30iVX_kV3HSJdxYh7/view. (2023). (Accessed on January 14, 2024).