Automating Safety Argument Change Impact Analysis for Machine Learning Components

Carmen Cârlan*
fortiss GmbH
carmen.carlan@gmail.com

Lydia Gauerhof*
Robert Bosch GmbH
lydia.gauerhof@de.bosch.com

Barbara Gallina
Mälardalen University
barbara.gallina@mdu.se

Simon Burton
Fraunhofer IKS
simon.burton@iks.fraunhofer.de

***Abstract*—The need to make sense of complex input data within a vast variety of unpredictable scenarios has been a key driver for the use of machine learning (ML), for example in Automated Driving Systems (ADS). Such systems are usually safety-critical, and therefore they need to be safety assured. In order to consider the results of the safety assurance activities (scoping uncovering previously unknown hazardous scenarios), a continuous approach to arguing safety is required, whilst iteratively improving ML-specific safety-relevant properties, such as robustness and prediction certainty. Such a continuous safety life cycle will only be practical with an efficient and effective approach to analyzing the impact of system changes on the safety case. In this paper, we propose a semi-automated approach for accurately identifying the impact of changes on safety arguments. We focus on arguments that reason about the sufficiency of the data used for the development of ML components. The approach qualitatively and quantitatively analyses the impact of changes in the input space of the considered ML component on other artifacts created during the execution of the safety life cycle, such as datasets and performance requirements and makes recommendations to safety engineers for handling the identified impact. We implement the proposed approach in a model-based safety engineering environment called FASTEN, and we demonstrate its application for an ML-based pedestrian detection component of an ADS.**

***Index Terms*—Safety Cases, Machine Learning (ML), Operational Design Domain (ODD), Change Impact Analysis (CIA)**

I. Introduction

The application of data-driven functions, such as machine learning (ML), is finding its way into safety-critical systems. For example, Automated Driving Systems (ADS) are increasingly using ML for perception tasks, such as pedestrian detection (see Fig. 1). This imposes certain challenges when it comes to the safety assurance of the overall system. On the one hand, the use of training data in place of a detailed functional specification hinders the traceability between assurance artifacts, which is mandated by safety standards, such as ISO 26262 [16] and ISO 21448 [17] in automotive. On the other hand, prediction uncertainty, lack of robustness to subtle changes in the inputs and a lack of explainability in the decisions made by the ML-model call for an iterative development process, whereas traditional safety engineering processes consider systems operating in well known and well defined environments are non-iterative.

*First two authors contributed equally to this work.

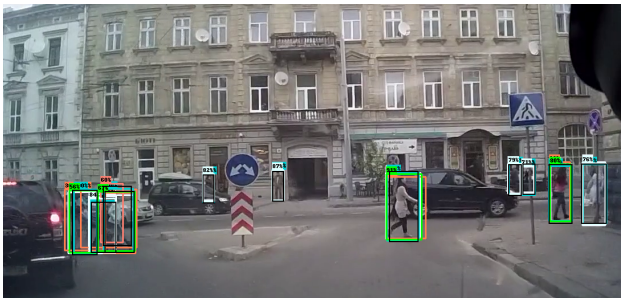

Fig. 1. JAAD data sample with prediction of three different DNNs trained to predict pedestrians (green, red, white bounding box) and the ground truth (black bounding box)

The iterative nature of ML development leads to the need to continuously adapt related assurance artifacts such as datasets and performance requirements. This is motivated by the underlying life cycle used both during model development and during operation, as previously unknown hazardous scenarios are continuously identified, leading to the need to update the model or make modifications to the context of use.

Automatically determining which assurance artifacts are impacted by such updates would be beneficial, but, however, challenging because it requires the existence of explicitly specified traces between assurance artifacts. Whereas ISO 26262 and ISO 21448 mandate traceability between safety requirements and the implementation, e.g., the code, ML based components are trained based on a given dataset, resulting in a large number of optimized model parameters that cannot be directly traced to individual requirements. This lack of direct traceability can be compensated for, e.g., by explicitly linking all the assurance artifacts within a safety argument [9]. In the literature, a set of patterns for building such arguments has been proposed [4], [12], [13], [31].

Given changes in the input space, to identify the impacted assurance artifacts, it is sufficient to execute change impact analysis (CIA) on the safety argument. Due to the frequency of such changes, the UL Standard for Safety for Evaluation of Autonomous Products UL 4600 [30] recommends the automated execution of safety case CIA. To enable automated CIA for safety cases, the checkable safety case framework [7] proposes a modeling approach for safety arguments, supporting the automated execution of safety argument CIA. Checkable safety argument models are deeply integrated with models of other assurance artifacts via direct traces, enabling

automated checks for consistency between safety arguments and the traced artifacts. Here, consistency means that there are no contradictions between the safety argument and the artifacts on which the argument is based. To detect inconsistencies, consistency rules shall be defined and the change scenarios that may violate the consistencies shall be identified. With the help of these traces, the CIA guides the safety assurance process, by supporting the safety engineer in recognizing dependencies at an early stage and initializing appropriate measures.

Paper contributions. **Contribution 1.** We further develop the checkable safety cases framework to enable accurate change impact analysis for safety arguments. To this end, we extend the state-of-the-art metamodel for safety cases to allow for the *semantically enrichment of the direct traces between argumentation elements and of the relationships between argumentation elements with change sensitivity information*. Change sensitivity information explicitly specifies how the impact of a change in a traced assurance artifact propagates throughout the argumentation structure. This also enables the identification of the impact a change in one assurance artifacts referenced by a safety argument has on other assurance artifacts referenced by the same safety argument. When they entail placeholders for direct traces to other assurance artifacts, which may be instantiated for a given system, checkable safety arguments are reusable. **Contribution 2.** We propose a *seven-step qualitative CIA* for arguments about the sufficiency of the used dataset (including training, test and validation data) given changes in the input space. The proposed seven-step CIA emphasizes the importance of monitoring changes in the ODD. We exploit the explicit traces between different artifacts specified in the safety argument to semi-automatically analyze the impact a change in the ODD has on the other artifacts. By doing so, we compensate for the missing traceability between ML specific artifacts. The more the semantics behind the traces are made explicit, the more accurate guidance the safety engineer can benefit from. **Contribution 3.** Complementary to the qualitative safety argument CIA, we also propose *a novel method to quantitatively evaluate the coverage of the input space and of the triggering conditions by the data*.

Paper structure. In Section II, we provide an overview of the concepts on which this paper is based. Then, in Section III, we discuss the state of the art of safety argument CIA and existing works towards the continuous development and assurance of ML components. In Section IV, we present an ML-based pedestrian detection component, which we use as example throughout the paper. In Section V, we present improvements of the checkable safety case framework to support the automated execution of accurate impact analysis of changes in the input space of an ML component. In Section VI, we illustrate the usage of the framework for the considered pedestrian detection component, and discuss tool support for the proposed solution. We conclude the paper in Section IX, while also providing ideas for future work.

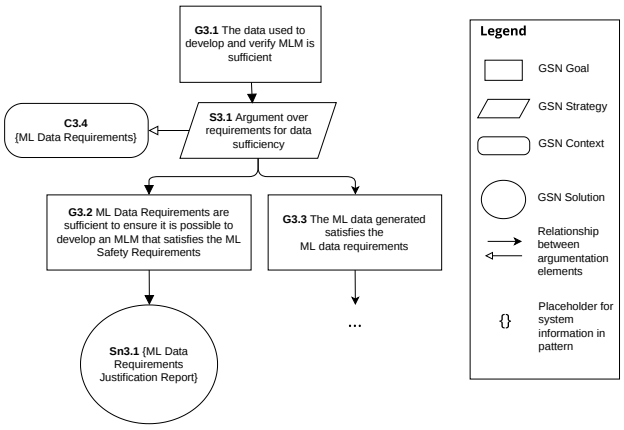

Fig. 2. A fragment of the AMLAS Pattern for Data Sufficiency [13]

.

II. Fundamentals

This section presents the basics of safety cases and checkable safety cases. In addition, insights are given into the artifacts that are referenced by the safety arguments scoping ML-based components. Further, an overview of the standards relevant for the development of safe ML components is provided and the importance of data sufficiency is discussed.

Safety Cases. A safety case is "a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment." (DS 00-56 Issue 4). To specify arguments, graphical representations such as the Goal Structuring Notation (GSN) [28] can be used. GSN is a graphical argument notation that can be used to explicitly document the individual argumentation elements (goals, strategy specifying how goals are supported by other goals, and ultimately supported by solutions, as depicted in Fig. 2), the context defined for the argument, and the relationships between the argumentation elements. The standard on GSN includes guidelines for the definition (template) and specification (explicit GSN modelling elements) of so called safety argument patterns, which are reusable, previously successful arguments, entailing parameters, i.e., placeholders for system-specific information [28]. To formally model a safety argument in compliance with GSN, one should use GSN/SACM [28] - the dedicated extension of the Structured Assurance Case Metamodel (SACM) [23].

Annotating Argumentation Elements with Status Information There are different types of impact a claim undergoes given a change in a traced artifact. Whereas Kelly and McDermid [21] differentiate between 'directly' and 'indirectly' and 'potentially' and 'actually' impacted elements, Kokaly *et al.* [22] discuss different states of safety case elements. According to Kokaly *et al.* [22], argumentation elements have a validity state, i.e., they may be either 'valid', reflecting the current state of the system, or they need 'to be rechecked'. 'Directly impacted' argumentation elements have a reference to a changed assurance artifact. A directly impacted argumentation element is the cause of initializing CIA. Argumentation elements that

do not have a direct trace to a changed artifact, but are impacted by the change are referred to as 'indirectly impacted'. Whereas potentially impacted elements have a direct or indirect trace to a changed artifact, and may or may not be impacted by the change.

Checkable Safety Cases. To enable automated CIA for safety cases, Cârlan *et al*. [7] introduced the checkable safety case framework. Checkable SCs are deeply integrated with other assurance artifacts, as the references to such artifacts in the argumentation elements are specified in a machine-readable manner, as direct traces between models.

Safety Cases for ML-based Components. Several approaches to structure the safety argument scoping an ML-based perception function applicable in different domains have been proposed [4], [12], [13], [27], [31]. All safety arguments related to ML-based perception functions have in common the fact that they refer to the following artifacts: Operational Design Domain (ODD), requirements, used data, Deep Neural Network (DNN), and triggering conditions identified during the execution of ISO 21448 activities. As mentioned in Section I, linking these artifacts and tracing them back to requirements is essential, since in a trained ML component there is no code that can be directly traced back to requirements [9]. The ODD represents the specified input space model and comprises the environment within which the system is designed to function safely, thus defining the system boundaries. The requirements steer the safety assurance activities within the data management and training phase, whereas the data is used for the development and the Verification & Validation (V&V) of the ML component. For this reason, data has a significant impact on the safe ML behavior. Usually, data comes with labels, annotations, and data analysis results.

Automotive Safety Standards and ML ISO 26262 focuses on the functional safety of electric and electronic systems used in road vehicles and has been established for many years in the automotive domain. Important shortcomings of ISO 26262 in relation to ML were highlighted in [14] and in [26]. However, ISO 26262 does not consider the assurance of the safety of the intended functionality (SOTIF), which is addressed in ISO 21448. SOTIF regards hazards caused by functional insufficiencies due to inappropriate specification of the functionality on system level or on components level, or insufficient performance of components. In the presence of triggering conditions, functional insufficiencies result in hazardous scenarios. Triggering conditions are specific conditions of a scenario that initiate a system reaction leading to hazardous behavior or the inability to handle a reasonably foreseeable indirect misuse. Appendix D.2.3 of ISO 21448 indicates that the ODD and data sufficiency have an important role in the SOTIF process. ISO/AWI PAS 8800 [18], which is about safety and artificial intelligence, focuses on the use of ML in road vehicles and is planned to cover functional safety and SOTIF with respect to artificial intelligence. However, it is still under review and continuously changing. It is expected to also consider data sufficiency to play an important role. ISO/IEC 22989 [19] provides AI concepts and terminology, whereas ISO/IEC 23053 [20] introduces a framework for describing a generic AI framework using ML technology. Although none of these standards focus on safety-critical applications, they regard the topic of risk management that includes safety as part of the AI life cycle.

Data sufficiency and data suitability As the Guidance on the Assurance of ML in Autonomous Systems (AMLAS) [13] provides for each phase of ML-lifecycle a safety argument pattern, it also presents a pattern for a arguing about the appropriate management of data. The top-level claim of this argument is that the data used during the development and verification of the ML model is sufficient [13]. Gauerhof et al. [10] define that training data is sufficient, when (1) there is no under-sampling of relevant content (e.g. data features) and when (2) there are no unintended correlations that are learnt by the ML as an inappropriate pattern. Undersampling refers to an underrepresentation of events e.g., that might be critical, but rare, whereas unintended correlations are cause of inappropriate functionality of the ML component, such as an ML based pedestrian detection learns that a pedestrian has always two legs and a head. Even though this might be valid for many cases, a person wearing a long skirt and a hat should be still be detected.

III. Related Work

Safety argument CIA. There are several model-based approaches for safety case maintenance [3], [6], [7], [22]. However, to the best of our knowledge, none of the existing approaches explicitly consider an argument about the sufficiency of the data used for the development of ML-based components, and, therefore, none of them specifically analyzes the impact of changes in the input space (i.e., ODD) on the system safety case. Given changes in the operating context referenced by a GSN context element, safety argument CIA such as [21] would identify that the entire argumentation under the respective context as potentially impacted, and, therefore, to be rechecked. The MMINT-A approach proposed by Salay*et al*. [22] and the Safety Artifact Forest Analysis (SAFA) proposed by Agrawal *et al*. [3] have as prerequisites the existence of models of each artifact referenced in the safety arguments, and the existence of traces between system models, as they analyze first the impact of a change in an artifact on the other artifacts, based on traces between the models of such artifacts. Based on the impact analysis executed at the level of system models, the impact on the safety arguments is analyzed. Such approaches do not handle the analysis of change impact for ML components, since, as we discussed in Section I, one challenge of ML components safety assurance is the deficitary or even nonexistent traceability between assurance artifacts. Our approach is, on the contrary, to analyze the impact of an artifact change on other artifacts at the level of the safety argument, based on the traces between argumentation elements and assurance artifacts and the relationships between argumentation elements.

Continuous development and assurance of ML components. Approaches for continuous development and assurance of ML

TABLE I
OVERVIEW OF POTENTIAL ODD CHANGES.

	Can be controlled in reality	Can not be controlled in reality
Add ODD category/ dimension/alternative to the ontology	Extend the ODD model e.g. include country to ODD	Concretize the ODD model e.g. add contrast to pedestrian or trees to vegetation
Delete ODD category/ dimension/alternative in the ontology	Limit the ODD model e.g. exclude country from ODD	Generalize the ODD model e.g. delete clothing color from pedestrian

components motivate the need for automated CIA for assurance artifacts of ML components. For example, SafetyOps [29] is a concept to enable a continuous and traceable safety life cycle. Here different automation frameworks (e.g., DevOps, TestOps, DataOps, MLOps) are combined. However, the work is in its early stages, as, for example, the linkage between the individual frameworks is not explicitly elaborated. Gauerhof et al. [9] provide guidance for linking explicitly artifacts (see the list of ML-specific artifacts in Section II) from different phases of the ML-life cycle. This linkage would allow artifacts from the requirements elicitation phase, e.g., the data requirements and the ODD to be updated in accordance to the testing results.

IV. USE CASE

An autonomous system, such as an ADS, requires a perception of its environment in order to move within it and conduct tasks. For example, for ADS operating in urban areas, pedestrian detection is required to drive safely without harming humans. As ML outperforms by far non-learning approaches, e.g. in perceiving objects, it is already indispensable. In this section, we briefly introduce a subset of the artifacts related to an ML-based pedestrian detection component. This subset of the artifacts was generated during the assurance of the considered ML component and referenced in the safety argument of the respective component.

ODD. The ODD of the system under consideration (SuC) refers to PAS 1883 [5], which provides requirements for the minimum hierarchical taxonomy for specifying an ODD to enable the safe deployment of an ADS. For our use case, we set the country of operation to Germany.

Requirements. The SuC shall implement the requirements presented by Gauerhof *et al.* [11].

DNN. The ML component is a DNN, which consists of a Squeezenet and Region Proposal Network (RPN), previously presented by Gauerhof *et al.* [11]. The architecture is comparably small, so that it can be implemented with low computation demand on dedicated hardware in an ADS.

Data. The DNN is trained and evaluated on the Joint Attention for Autonomous Driving (JAAD) dataset [24]. The JAAD *dataset* includes 346 video clips recorded over 240 hours of driving in America, Canada, Germany and Ukraine with circa 82000 image samples. These data cover various driving situations with a variation of all four seasons, weather conditions (e.g., rain, snow and fog), and day and night time.

Triggering conditions. In Table II, we present a subset of the identified SOTIF triggering conditions, which are derived from the DNN error categories from [10]. Particular triggering conditions are expected to be mitigated by retraining the DNN with adapted data. Thereby, data sufficiency is achieved, if data is free from under-sampling of relevant data characteristics and unintended correlations [10]. If the ODD does not include the elements that are related to the triggering conditions, it is updated. In the next section, we propose a CIA considering updates of the ODD.

V. SEMANTICALLY ENRICHED CHECKABLE SAFETY ARGUMENT PATTERNS

To support safety engineers in modeling checkable safety arguments, we introduce checkable safety argument patterns, extending the concept of safety argument patterns. Checkable patterns document a semantically-enriched, reusable argumentation structure, which facilitates the automated execution of accurate safety argument CIA given changes in traced assurance artifacts. They are reusable as the have placeholders for system-specific information, i.e., placeholders for direct traces to assurance artifacts.

As we recall in Section II, checkable safety arguments have direct trace links between safety argumentation models and models of other assurance artifacts. However, these direct trace links only support the automated identification of the potentially impacted safety argumentation elements. To enable the automated identification of the actually impacted argumentation elements, we propose to semantically enrich the direct trace links between argumentation elements and assurance artifacts and the relationships between argumentation elements by annotating them with change sensitivity information. Consequently, we extend the GSN/SACM metamodel so that it supports the specification of such annotations, by adding *ChangeSensitivityAnnotation* - a dedicated class for modeling such annotations, and by adding a new attribute of type *ChangeSensitivityAnnotation* to the classes for the direct traces and relationships. Change sensitivity annotations formally specify a set of tuples consisting of: 1) the changed traced artifact, 2) the type of change the traced artifact may undergo, 3) the impact the respective change type has on the argumentation element having a trace to the changed artifact (e.g., if the argumentation element is valid or to be rechecked), and 4) update recommendations to recover from the identified impact. The update recommendations can be both for the safety argument itself, and other assurance artifacts. Examples of change sensitivity annotations can be seen in Fig. 3. The annotation *A2* in Fig. 3 specifies that, given 1) an ODD model, which is changed by 2) including an alternative, 3) *C3.1* argumentation element is to be set as to be rechecked, and 4) the update recommendation is to analyze the dataset referenced by the context argumentation element. Such change sensitivity annotations are to be specified by the safety engineer creating the safety argument, and then reviewed by an external assessor.

After changing a traced artifact due to an update recommendation, e.g., adaptation of the dataset, it is necessary to continue the CIA, i.e., to analyze the impact of the respective change on the argumentation structure.

TABLE II
EXEMPLARY IDENTIFIED TRIGGERING CONDITIONS

Description incorrect detections	Triggering conditions	Mitigation	Element in ODD	Necessary artifact changes
FN on partially occluded pedestrian	Occlusion	Retraining (data sufficiency)	Relationship pedestrian & background	ODD and data requirements
Missing labels	Missing label	Increased labelling quality	-	labeling specification
FP on tree, tree branches	tree, tree branches	Retraining (data sufficiency)	tree, tree branches	ODD and data requirements

In Fig. 3, we show *C1.1* context, which has a direct trace to an ODD model. Via the change sensitivity annotations to the direct trace to the ODD model, we specify that given any change in the ODD, the checkable context *C1.1* is to be set as *modified*, and the update recommendation is to execute a safety argument CIA, in order to analyze the impact of the ODD change on the safety argument under the respective context. Further, in Fig. 3, one can also see the change sensitivity information annotated to the relationship between the *C3.1* context and the *G3.1* goal. The annotation specifies that, given the exclusion of an alternative from the ODD, the *G3.1* goal is not impacted, i.e., the goal is still valid. Indeed, data with an unnecessarily broad representation of elements could lead to a degraded performance e.g. due to higher variance or spurious correlations. However, this would only be identified in a later development phase as a triggering condition, such as after retraining and testing, and thus, handled appropriately. Given the inclusion of a new ODD alternative, the *G3.1* goal needs to be rechecked, i.e., the status of the data coverage with respect to the new alternative is analyzed. If necessary, the dataset shall be enhanced with data covering the new ODD alternative.

VI. A CHECKABLE ARGUMENT PATTERN FOR DATA SUFFICIENCY

In this section, we present the *Checkable Assurance Argument Pattern for Data Sufficiency*, which enables the automated execution of accurate CIA for the safety argument addressing data sufficiency. The CIA scopes the identification and handling of the impact of changes in a traced ODD model on the safety argument obtained after pattern instantiation. Changes to the ODD may be motivated by the need to handle identified triggering conditions. For example, in the context of our use case, one identified triggering condition is that, due to the lack of training data suitability, occluded pedestrians were not identified, resulting in systematic false negatives of the DNN output (see Table II). Consequently, we needed to add to the pedestrian category the dimension occlusion.

Modeling the ODD Having an ODD model is a prerequisite for instantiating the pattern. To model the ODD, in this work, we used an ontology proposed by Herrmann *et al.* [15], which specifies the input domain of a camera-based pedestrian detection component in an ADS as a set of categories, having different relations with each other [15]. An ODD category is refined by dimensions, which, in turn, are refined by different alternatives. For example, the category 'pedestrian' is refined by several dimensions, such as 'age', and 'pose', whereas the 'pose' dimension has a set of included alternatives, such as 'standing', 'walking', 'running'. An alternative may be included or excluded from the ODD of the SuC. There are several types of changes that the ODD model can undergo. In Table I, we show the types of changes, while also discussing the impact of the change to the ODD - if it concretizes, generalizes, limits, or extends it. We differentiate between controllable and uncontrollable ODD elements. Controllable ODD elements can be controlled in the real world, meaning that during the development the ODD element can be monitored and influenced. Other ODD elements cannot be controlled. For example, one can control on which type of road (e.g., highway or city street) the ADS can be activated, but one cannot control if the ADS encounters pedestrians with camouflage clothing. While modifying controllable elements, one can either include or exclude the ODD element, whereas the ODD may be either concretized or generalized when adding or deleting uncontrollable elements. Especially the controllable elements shall be reviewed by the safety engineers, whether to add them to the ODD or exclude them, as they might heavily influence the overall functionality and the corresponding safety requirements.

Pattern intent. The intent of the AMLAS pattern for data is to specify a reusable structure for arguing about data sufficiency. As written in ISO 21448, a sufficient test dataset increases the confidence in the argument that the ML component reaches its safety goals. Also training data require no under-sampling of relevant content (e.g. data features) [10]. To enable automated safety argument CIA, we model the argumentation structure proposed by the AMLAS pattern. The intent of the aforementioned safety argument CIA is to keep the argumentation structures resulting from the instantiation of the pattern consistent with the ODD model of the system.

Pattern structure. In Fig. 4, we present the GSN-based graphical representation of the pattern. The top-level goal *G3.1* claims that the used dataset is sufficient. Instead of modeling all three *C3.1*, *C3.2*, and *C3.3* context elements from the AMLAS pattern referencing different types of dataset (i.e., training, testing and validation datasets), we model only one context, which references all these datasets. We replace the placeholders for text-based references to artifacts with placeholders for direct traces to assurance artifacts. Further, both the placeholders for direct traces to assurance artifacts and the relationships between argumentation elements are annotated with change sensitivity information. Due to space restrictions, we show in Fig. 4 only the annotation identifier, whereas in Fig. 3, we depict some of the annotations completely. Apart from considering the structure specified by the AMLAS pattern, the checkable pattern also includes further context information, i.e., the description of operational environment - **C1.1**, from an upper level argumentation structure specified by a related AMLAS pattern, namely the *Argument Pattern for*

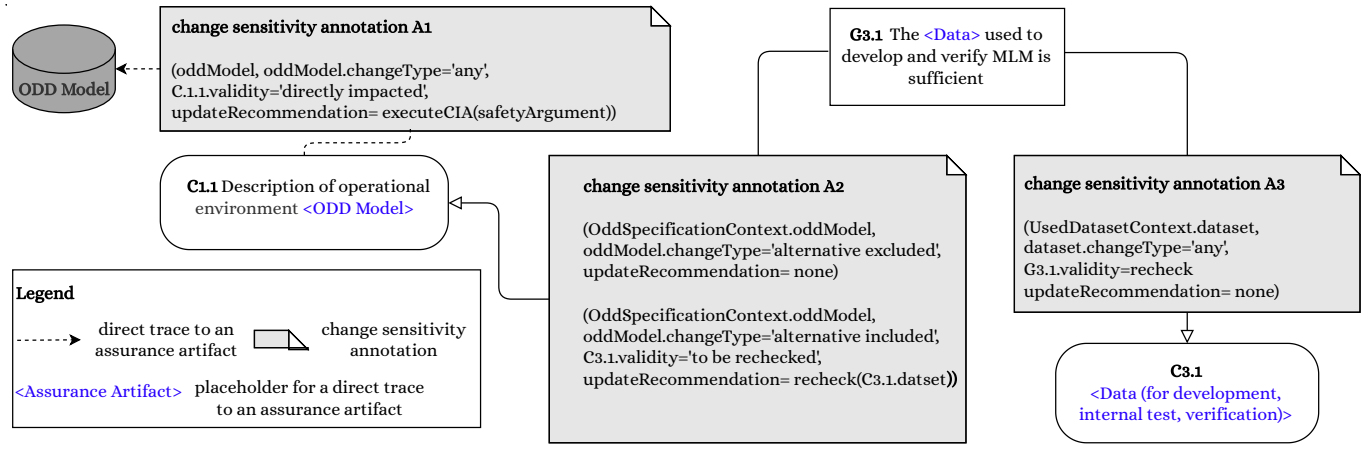

Fig. 3. Context argumentation element having a direct trace to an ODD model; trace that is annotated with change sensitivity information.

ML Safety Assurance. Where suitable, we specified references to assurance artifacts with direct traces to models of the respective assurance artifacts. For example, as seen in Fig. 4, the *C1.1* context has a placeholder for a direct trace to an ODD model.

Consistency rules considered during CIA. Given a change to the ODD (see Table I), it shall be checked whether the system still functions safely in the changed assumed operational context: **consistency rule 1:** *the used dataset shall cover all the ODD elements*. This consistency rule is taken from ISO 21448, which argues that, a sufficient coverage of the ODD by the data facilitates the component to learn how to handle hazardous scenarios and from [10] where no under-sampling of relevant content (e.g. data features) is required for training data. For simplicity reasons, we did not add further consistency rules. For example, consistency rules between the safety argument and the ML data requirements, referenced by goals *G3.4* - *G3.7*, should be also defined.

In the following, we explain how this consistency rule is checked via qualitative and quantitative CIA.

Quantitative CIA. Whenever the ODD is modified in order to address an identified triggering condition, not only the impact on the safety argument is analyzed, but also the data coverage is re-assessed [25]. To this end, in accordance with **consistency rule 1**, we propose to evaluate the coverage of the identified triggering conditions by the data, the ratio of covered triggering conditions to data, and the coverage of the ODD by the data, as depicted in Fig. 5. The required threshold for the coverage of identified, i.e., known, known triggering conditions, e.g. $n_{TC_X} >= 1$ sample for the particular triggering conditions TC_X, might be set higher than the threshold for the coverage of a particular ODD element $m_{ODDelement_X}$, as triggering conditions are considered critical. To be able to count the samples according to a particular ODD element, annotations are needed, e.g. occlusion annotated to the ground truth, or metrics to measure particular ODD elements, e.g. contrast in the bounding box measured by Mean Square Root (MSR) contrast. Having real data that sufficiently covers all identified triggering conditions might be challenging, as critical situations are dangerous to collect. Here synthetic data might ease the challenge. It is also feasible to count the uncovered TCs, as each TC is a combination of ODD elements. A particular TC is uncovered, if the required amount of samples for a particular triggering condition $TC1$ is not available in the dataset. In general, a higher coverage increases the confidence in the safety argument, as it implies that more safety critical scenarios containing identified triggering conditions are considered to ensure that SOTIF is achieved. This approach could be extended by considering development data and V&V data separately.

Qualitative CIA. The CIA steps to be taken whenever the ODD undergoes a change of the types presented in Table I are depicted in Fig. 4 and enlisted in the following. Also, in Fig. 4, we show the status of argumentation elements after each CIA step. The status of argumentation elements is specified as discussed in Section II. The traced artifacts, such as the ODD model, or the data can be annotated as 'changed', to specify they have undergone certain modifications. One CIA step may have multiple possible outputs, which, in the model is represented by the 'choice' element (see Fig. 4). The output in this case is determined by evaluating one or more conditions. The pseudo code for this seven-step procedure is also given in Algorithm 1.

[Step 1] The ODD model traced by the *OddSpecification-Context* argumentation element is continuously checked for changes. Based on the change sensitivity information annotated to the direct trace between the argumentation element and the ODD model (see annotation *A1*), given any change to the ODD model, the validity state of the tracing context argumentation element is set as directly impacted. The update recommendation is to further execute CIA, namely to identify the change impact propagation throughout the argumentation structure.

[Step 2] The fact that the validity state of context *C1.1* has been set to directly impacted potentially impacts top-level goal *G3.1*, which, in accordance to *A2*, is set as to be rechecked. Also, based on the GSN relationships, the change impact will propagate to *C3.1* [21]. By excluding an alternative, goal *G3.1*

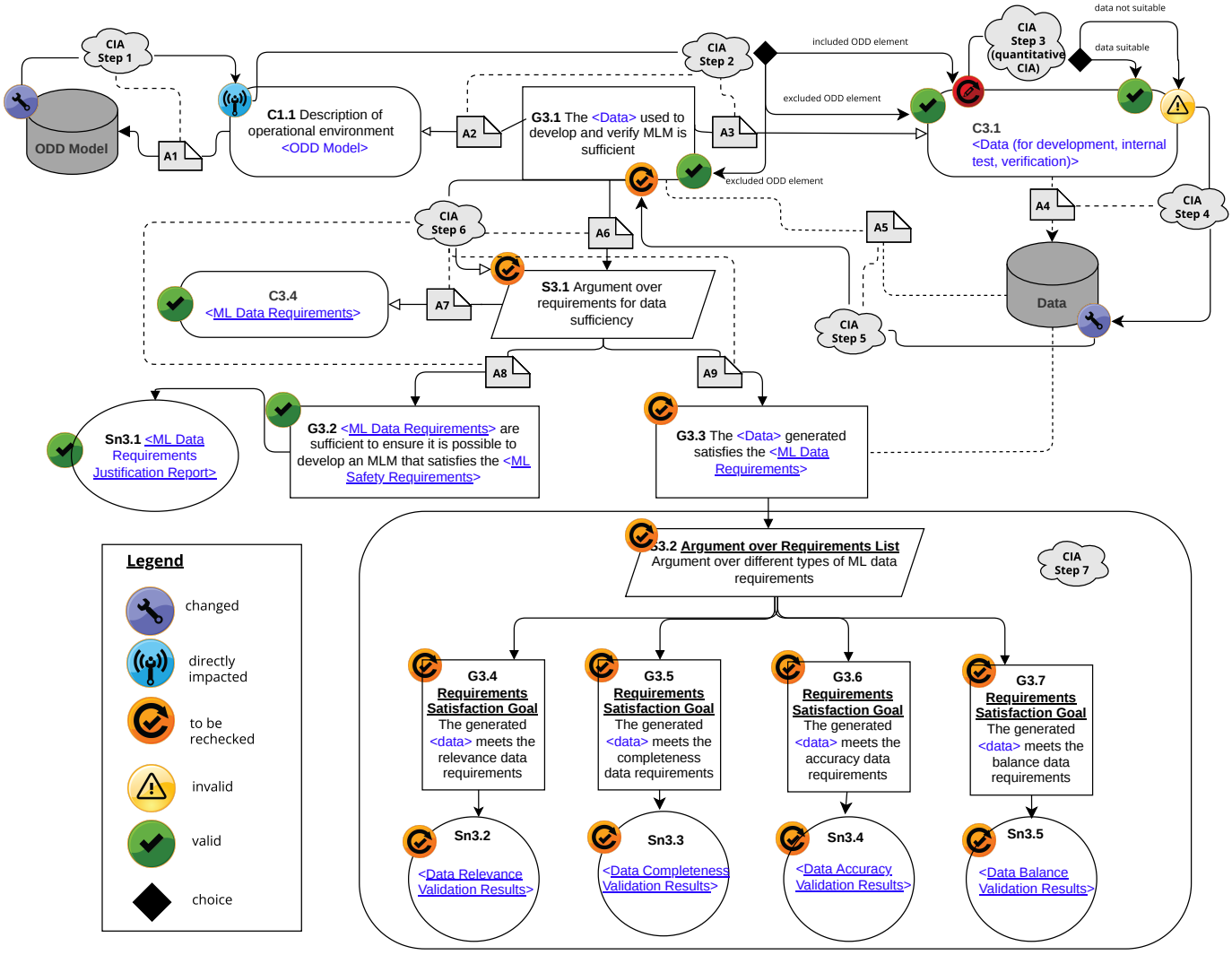

Fig. 4. The *Checkable Assurance Argument Pattern for Data Sufficiency*, with placeholders for direct traces to an ODD model and to datasets. While regular pattern parameters are depicted within curly brackets {}, placeholders for direct traces to certain types of models are depicted in angle brackets ⟨⟩. Both the traces to artifacts and the relationships between argumentation elements are semantically enriched with change sensitivity annotations.

and context *C3.1* are not impacted by the change and remain valid, and therefore the CIA stops. Whereas if the change consists of the inclusion of an alternative, *C3.1* context is annotated as to be rechecked, and the update recommendation is to analyze if the dataset referenced by *C3.1* is sufficient, i.e., if it sufficiently covers the newly added ODD element.

[Step 3] Here it is checked if the required coverage is achieved. For example, if rain constitutes a triggering condition and there is data including annotations about the rain status in the data samples, then the number of samples for this particular triggering condition can be automatically counted. Consequently, via formula given in Fig. 5, it can be determined if the given coverage exceeds the required coverage or not. This analysis could be automated, given the correct and complete annotation of data and the adequate tool support to verify data sufficiency. If the dataset is indeed insufficient, the *C1.1* context is annotated as outdated. The update recommendation for such an impact is the appropriate update of the dataset and then retraining of the ML component. Whereas if the dataset is found to be still sufficient, the *C3.1* context remains valid, and the CIA stops here.

[Step 4] If the dataset is not sufficient, then, in accordance to change sensitivity annotation *A4*, the dataset needs to be appropriately modified.

[Step 5] Given the direct trace between goal *G3.1* and the dataset, and annotation *A5*, when the dataset changes, *G3.1* is directly impacted, and needs to be rechecked.

[Step 6] Given the need to recheck goal *G3.1*, based on the relationships between argumentation elements, all argumentation elements supporting *G3.1* should be rechecked. However, our CIA, based on change sensitivity annotations *A6-A9*, identifies that the modification of the dataset only actually impacts goal **G3.3**, which has a direct trace to the data. Context *C3.4* having a direct trace to the ML data requirements

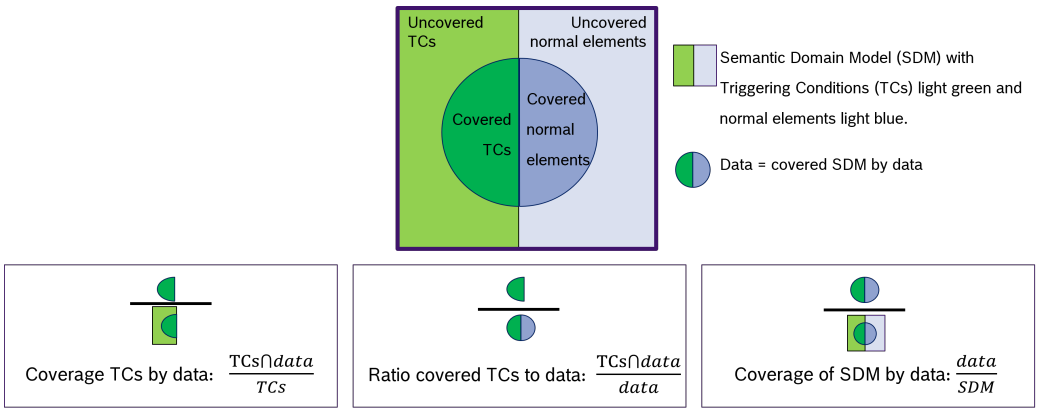

Fig. 5. Formulas computing the coverage of the TCs by the data, the ratio of covered the TCs to data, and the coverage of the ODD by the data.

Algorithm 1 Propagation of CIA throughout the safety argument depicted in Fig. 4

```
for Every time step do
    Check status of ODD Model for changes (see Table II)
    if Status of ODD Model is set to 'changed' then
        Step 1: Provide update recommendation as specified by change sensitivity annotation A1
        for Every CIA step i; i=i++ do
            Set status of the traced argumentation elements and assurance artifacts
            if Status of traced argumentation elements is set to 'directly impacted' 'to be rechecked', and of traced assurance artifacts to 'changed' then
                Provide update recommendation specified by the change sensitivity annotation(s) to the safety engineer or initialize an automated review procedure, e.g. quantitative CIA.
            else
                Stop CIA
            end if
        end for
    else
        Stop CIA
    end if
end for
```

is not impacted by the changes in the dataset, since the requirements remain the same. Further, goal *G3.2* arguing about the sufficiency of requirements is also not impacted, since the requirements are not changed, only the data set. Consequently, solution *Sn3.1* is annotated as still valid, and can be reused after a change in the ODD model.

[Step 7] Based on the relationships between argumentation elements and because of their direct traces to the dataset, the subgoals and solutions supporting goal *G3.3* are annotated as to be rechecked.

Pattern applicability. The pattern may be used for any ML component implementing a safety-critical function, given the existence of all the artifacts referenced by the argument (i.e., ODD, triggering conditions, safety requirements).

Pattern consequences. Whenever CIA scoping the argument resulted from the instantiation of the *Checkable Assurance Argument Pattern for Data Sufficiency* is executed, the propagation of the change impact to rest of the safety case shall be also analyzed. However, so far, we do not provide support for this analysis and, consequently, it needs to be done manually.

Pattern implementation. The checkable pattern may be implemented only when the considered dataset and models of the ODD, and ML data requirements are available. Further, the type of evidence required by the pattern, such as data completeness validation results shall also be available.

VII. Tool Support and Experience with Instantiating the Pattern

Next, we present our experience with instantiating the *Checkable Assurance Argument Pattern for Data Sufficiency* proposed in the previous subsection (see Fig. 4) for the pedestrian detection component presented in Section IV.

We executed the CIA steps, assuming that we concretize the 'pose' dimension of pedestrian by adding and including new alternatives, namely 'sitting', and 'lying'. This change was motivated by the refinement of the "Unusual pose of pedestrian" triggering condition leading to a misdetection of the ML component. Given this change, the execution of **Step 1** of the CIA, where the ODD model is continuously checked for changes, will result in annotating *C1.1* context as directly impacted, which will then initiate **Step 2**. In **Step 2**, given the considered change, and based on the update recommendation specified by change sensitivity annotation *A3*, the *C3.1* context argumentation element is annotated as to be rechecked, which leads to **Step 3** of the CIA, where the sufficient coverage of the ODD by the training data is checked. In **Step 3**, the training data is found to not sufficiently cover the new pedestrian poses. *C3.1* context is annotated as outdated, whereas in **Step 4**, the dataset is to be updated by the safety engineer, so that it entails sufficient samples of each pedestrian pose included in

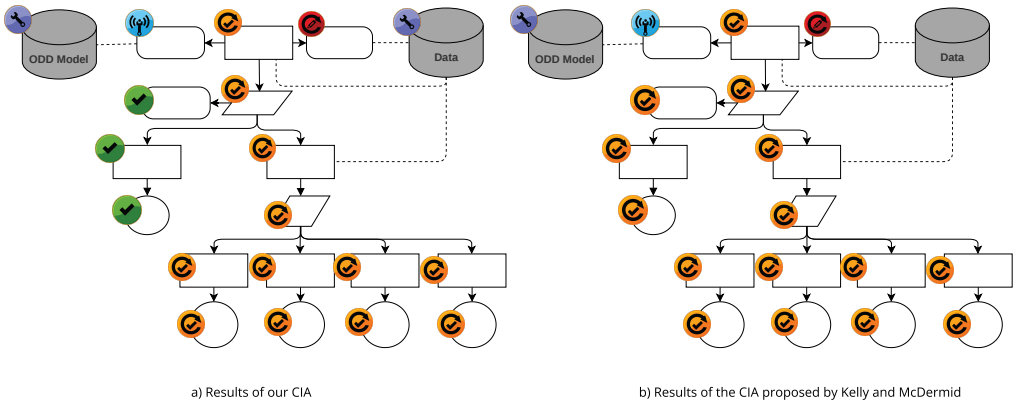

Fig. 6. CIA results for the use case.

the ODD. Given the modification of the dataset, in accordance with change sensitivity annotation *A5*, in **Step 5** *G3.1* is annotated as to be rechecked. After executing Step 6 and **Step 7**, while *C3.4*, *G3.2*, *Sn3.1* are annotated as to valid, and can be reused, *G3.3* and its supporting evidence need to be rechecked.

In Fig. 6-a, we show both the impacted area within the safety argument, and the area that can be reused. In addition, we discuss about how our CIA compares to a state-of-the-art CIA. We compare the impact area identified by the approach proposed by Kelly and McDermid [21], while considering the instantiated AMLAS pattern for the SuC and the aforementioned concretization of the 'pose' dimension with ours. The approach proposed by Kelly and McDermid [21] cannot identify automatically a change in a referenced model, the CIA being triggered manually by the safety engineer. This is because the approach do not support the specification of machine readable, direct traces to the referenced assurance artifacts, but only of text-based references. Given any type of change in the referenced ODD, following the CIA approach of Kelly and McDermid [21], the safety engineer manually annotates the *C1.1* argumentation element as directly impact. Then, following the relationships between argumentation elements, their CIA can automatically annotate all the elements in the argumentation structure as potentially impacts. Next, the safety engineer needs to manually review the actual impact of the change on each element in the argumentation structure, covering the entire argumentation. The approach does not provide any update recommendations. For example, in contrast to our approach, it does not recommend when and how the data needs to be updated, as a reaction to changes in ODD. We observethat our approach allows for a more accurate CIA than existing approaches [21].

We modeled the checkable pattern in FASTEN [1], [8]. The pattern instantiation implies the specification of traces from the safety argument model to models of different assurance artifacts (e.g., to ML data requirements, ML safety requirements, ODD and triggering conditions). To this end, we extended FASTEN with new Domain specific Languages (DSLs) to enable the modeling of the ODD, and of the identified SOTIF triggering conditions. However, the tool does not yet support the automated identification of the type of change the ODD model undergoes. Therefore, whenever a change in the ODD model is detected, the CIA in FASTEN will annotate the *UsedDatasetContext* as to be rechecked without differentiating an alternative is included or eliminated, and the update recommendation will be to recheck the dataset. As such, the CIA supported in FASTEN is more conservative and less accurate than the CIA presented in the previous section. Also, the quantitative CIA is still yet to be implemented in CIA. Currently, the new functionality in FASTEN is at Technology Readiness Level (TRLs) TRL4 (validated in a lab). To be used in industrial projects, so that a safety manager or safety engineer relies on the automation, tool qualification would be required.

In Fig. 7, one can see a screenshot of a part of the model of the instantiation of the checkable pattern for the pedestrian detection ML-based component described in Section IV. While **C1.1** has a direct trace to the ODD we modeled for the use case, **C3.1** references external documents with the used datasets. Fig. 7 shows a recommendation message output by the implemented consistency checks, given the occurrence of a change in the traced ODD model.

The checkable pattern can be downloaded from the git repository [2]. The pattern will be also included in the upcoming release of FASTEN.

VIII. Discussion

Modeling checkable safety cases implies more effort than modeling regular safety cases because one needs also to specify the change sensitivity information for all traces. However, this is a one time effort, and it is compensated by the reduced effort for executing CIA every time a system change occurs. The more iterations on the safety argument are conducted, the higher the benefit of the checkable safety argument is.

The accuracy of the proposed CIA highly depends on the correct linkage between safety arguments and other assurance artifacts. However, the advantage here is that the semantics behind the traces are made explicit, enabling the automated execution of consistency checks. As we discussed in the introduction, this traceability is not trivial for Machine Learning components. The more traces are made explicit, the more trust can be laid in the safety argument. The accuracy of our CIA

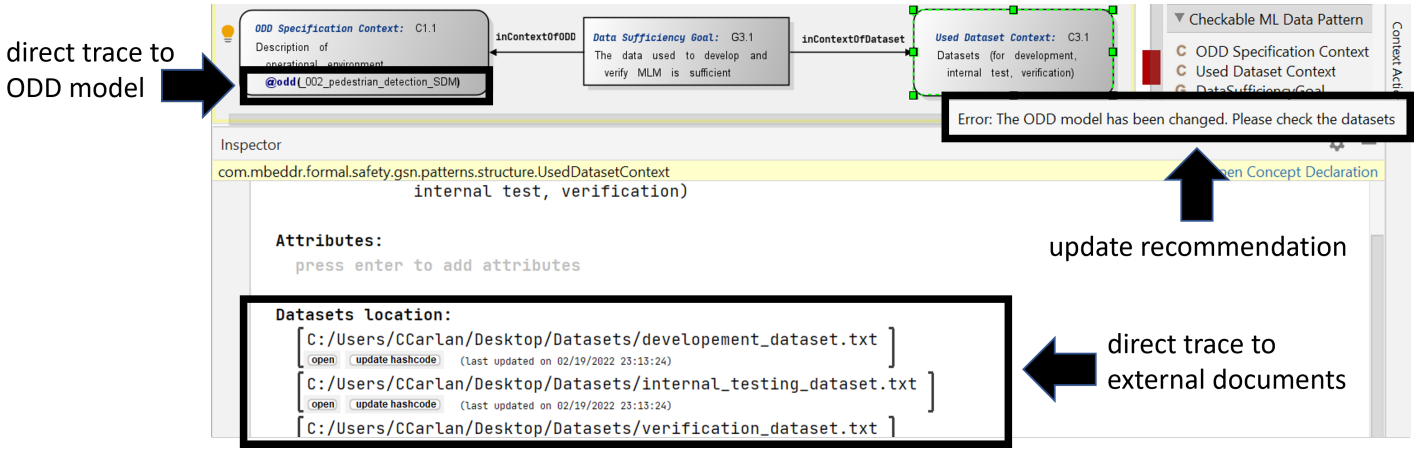

Fig. 7. FASTEN screenshot with the *Checkable Assurance Argument Pattern for Data Sufficiency* instantiated for the pedestrian detection use case

also depends on the correct and complete specification of the change sensitivity information.

However, our solution does not replace the need of a safety engineer to conduct CIA, but automates parts of the CIA steps. As the safety engineer is the one who specifies the change sensitivity information, the correctness and completeness of the automated CIA depends on his/her qualification.

The proposed CIA steps for the selected use case depend, among other things, on the considered safety argument pattern and change scenario. In this work, we consider the adaptation of the ODD as change scenario and the a safety argument pattern from the literature. This change scenario is an important use case, since, as also specified in ISO 21448, knowledge about previously unknown conditions must be incorporated during development. Namely, whenever hazardous scenarios are revealed either during V&V or during system operation, the description of the ODD needs to be updated so that it enables the specification of the respective scenario. It is recommended that the data used follows the description of the ODD and not the other way around, since the ODD is the basis of all safety requirements, including the data requirements. If triggering conditions are noticed that could lead to a failure, then these must be documented in the ODD and not just mitigated by further measures, changing the dataset.

Individual update recommendations have the potential to be automated, such as the proposed quantitative CIA. However, some update recommendations will remain impossible to automate in the near future. For example, in our use case a retraining on extended data might be not sufficient for particular reflection during sunset, as the camera is blinded. Then a mitigation on system level might be more efficient. This hazard might be handled by another sensor, e.g. radar, that is not affected by this physical effect.

Whereas in this work we focused on an argument for the sufficiency of the dataset used for the development of an ML component, our solution for modeling safety arguments and safety argument patterns in order to increase the accuracy of automated safety argument CIA can be applied to various safety arguments for different applications.

IX. Conclusion and Future Work

In this paper, we proposed to increase the accuracy of safety argument CIA, by enhancing the traces between argumentation elements and models of assurance artifacts with annotations specifying the impact a change in the traced artifacts has on the tracing argumentation elements. Also, to identify the propagation of that impact throughout the entire safety argument structure, we annotated traces between argumentation elements with change sensitive information about how a change in a model traced by one argumentation element impacts the other argumentation elements. We implemented the solution in a model-based environment called FASTEN, and we exemplified it for an ML-based pedestrian detection component of an automated driving system.

We illustrated how to model a state-of-the-art reusable safety argument pattern as checkable. The chosen argument pattern is one proposed by AMLAS and it reasons about the sufficiency of data for ensuring that an ML-based component satisfies its safety requirements. Making the pattern checkable allows for the automated execution of accurate qualitative and quantitative safety argument CIA given changes in the input space, which is specified as an ODD model. The proposed CIA is accompanied by update recommendations for impacted assurance artifacts to guide safety engineers through the process of handling the identified impact.

Next, we intend to further validate our approach by applying it to other use cases and also by exercising it for more change scenarios. Also, our work opens up new potential lines of research. For example, we intend to analyze the impact of other types of changes (e.g., changes to the system architecture, or the safety requirements) on the argumentation structure of the AMLAS pattern addressed in this paper.

Acknowledgement. The research leading to the results presented above are funded by the German Federal Ministry for Economic Affairs and Energy within the project *KI Absicherung – Safe AI for automated driving*. B. Gallina is partially supported by the Sweden's Knowledge Foundation via the Safe and Secure Adaptive Collaborative Systems project.

References

[1] https://sites.google.com/site/fastenroot/, accessed on 02.08.2022

[2] https://github.com/mbeddr/mbeddr.formal/actions/runs/2024913071, accessed on 02.08.2022

[3] Agrawal, A., Khoshmanesh, S., Vierhauser, M., Rahimi, M., Cleland-Huang, J., Lutz, R.R.: Leveraging artifact trees to evolve and reuse safety cases. In: Proceedings of the 41st International Conference on Software Engineering - ICSE. pp. 1222–1233. IEEE / ACM (2019)

[4] Bloomfield, R., Fletcher, G., Khlaaf, H., Hinde, L., Ryan, P.: Safety case templates for autonomous systems (2021)

[5] BSI org: BSI PAS 1883 Taxonomy for specifying an Operational Design Domain, https://www.bsigroup.com/en-GB/CAV/pas-1883/

[6] Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T.: Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Transactions on Software Engineering **44**(11), 1039–1069 (2018)

[7] Cârlan, C., Petrisor, D., Gallina, B., Schoenhaar, H.: Checkable safety cases: Enabling automated consistency checks between safety work products. In: Proceedings of the 31st International Symposium on Software Reliability Engineering - ISSRE Workshops. pp. 295–302. IEEE (2020)

[8] Cârlan, C., Ratiu, D.: Fasten.safe: A model-driven engineering tool to experiment with checkable assurance cases. vol. 12234, pp. 298–306. Springer (2020)

[9] Gauerhof, L., Gansch, R., Heinzemann, C., Woehrle, M., Heyl, A.: On the necessity of explicit artifact links in safety assurance cases for machine learning. In: Proceedings of the International Symposium on Software Reliability Engineering - Workshops (ISSREW). IEEE (2021)

[10] Gauerhof, L., Hagiwara, Y., Schorn, C., Trapp, M.: Considering reliability of deep learning function to boost data suitability and anomaly detection. In: Proceedings of the International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 249–254. IEEE (2020)

[11] Gauerhof, L., Hawkins, R., Picardi, C., Paterson, C., Hagiwara, Y., Habli, I.: Assuring the safety of machine learning for pedestrian detection at crossings. In: Proceedings of the 39th International Conference on Computer Safety, Reliability, and Security - SAFECOMP. Springer (2020)

[12] Gauerhof, L., Munk, P., Burton, S.: Structuring validation targets of a machine learning function applied to automated driving. In: Proceedings of the 37th International Conference on Computer Safety, Reliability, and Security - SAFECOMP. Springer (2018)

[13] Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., Habli, I.: Guidance on the assurance of machine learning in autonomous systems (AMLAS). arXiv preprint arXiv:2102.01564 (2021)

[14] Henriksson, J., Borg, M., Englund, C.: Automotive safety and machine learning: Initial results from a study on how to adapt the iso 26262 safety standard. In: 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS). pp. 47–49 (2018)

[15] Herrmann, M., Witt, C., Lake, L., Guneshka, S., Heinzemann, C., Bonarens, F., Feifel, P., Funke, S.: Using ontologies for data set engineering in automotive ai applications. In: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE (2022), accepted

[16] ISO: Road vehicles – functional safety. ISO ISO 26262-(1-12):2018, International Organization for Standardization, Geneva, Switzerland (2018)

[17] ISO Org: ISO 21448 Road vehicles — Safety of the intended functionality, https://www.iso.org/standard/77490.html

[18] ISO Org: ISO/AWI 8800 Road vehicles — Safety and artificial intelligence, https://www.iso.org/standard/83303.html

[19] ISO Org: ISO/IEC 22989:2022 Information technology — Artificial intelligence — Artificial intelligence concepts and terminology, https://www.iso.org/standard/74296.html

[20] ISO Org: ISO/IEC 23053:2022 Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML), https://www.iso.org/standard/74438.html

[21] Kelly, T., McDermid, J.: Systematic approach to safety case maintenance. Reliability Engineering and System Safety **71**, 271–284 (2001)

[22] Kokaly, S., Salay, R., Chechik, M., Lawford, M., Maibaum, T.: Safety case impact assessment in automotive software systems: An improved model-based approach. In: Proceedings of the 36th International Conference on Computer Safety, Reliability, and Security – SAFECOMP. pp. 69–85

[23] Object Managment Group: Structured Assurance Case Metamodel - SACM, version 2.1. Tech. rep. (2020), https://www.omg.org/spec/SACM/About-SACM/

[24] Rasouli, A., Kotseruba, I., Tsotsos, J.K.: Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior. Proceedings of the International Conference on Computer Vision Workshops (ICCVW) pp. 206–213 (2017)

[25] Salay, R., Queiroz, R., Czarnecki, K.: An analysis of iso 26262: Using machine learning safely in automotive software (2017), https://arxiv.org/abs/1709.02435

[26] Salay, R., Queiroz, R., Czarnecki, K.: An analysis of iso 26262: Machine learning and safety in automotive software. SAE Technical Papers **2018** (2018), https://doi.org/10.4271/2018-01-1075

[27] Schwalbe, G., Knie, B., Sämann, T., Dobberphul, T., Gauerhof, L., Raafatnia, S., Rocco, V.: Structuring the safety argumentation for deep neural network based perception in automotive applications. In: Proceedings of the International Conference on Computer Safety, Reliability and Security Workshops (SafeComp). p. 383–394. Springer-Verlag (2020)

[28] SCSC: Goal structuring notation, https://scsc.uk/gsn

[29] Siddique, U.: Safetyops. arXiv preprint arXiv:2008.04461 (2020)

[30] UL Org: Presenting the Standard for Safety for the Evaluation of Autonomous Vehicles and Other Products, https://ul.org/UL4600

[31] Zhao, X., Huang, W., Bharti, V., Dong, Y., Cox, V., Banks, A., Wang, S., Schewe, S., Huang, X.: Reliability assessment and safety arguments for machine learning components in assuring learning-enabled autonomous systems. CoRR **abs/2112.00646** (2021), https://arxiv.org/abs/2112.00646