KI ABSICHERUNG
Safe AI for Automated Driving

KI Absicherung – Finale Ergebnissteckbriefe TP4

Version zur Veröffentlichung

Version	1.0
Editor	Frédérik Blank / Robert Bosch GmbH Andreas-Juergen Rohatschek / Robert Bosch GmbH Dr. Stephan Scholz / Volkswagen AG PD Dr. Michael Mock / Fraunhofer IAIS
Projektkoordination	Volkswagen AG / Fraunhofer IAIS
Fälligkeit	31.12.2022
Erstellungsdatum	03.11.2022

Gefördert durch:
Bundesministerium für Wirtschaft und Klimaschutz
aufgrund eines Beschlusses des Deutschen Bundestages

Dokumenteninformation

Autoren

Maram Akila / Fraunhofer IAIS

Markus Bach / Valeo Schalter und Sensoren GmbH

Stefan Blaesius / BMW Group

Frédérik Blank / Robert Bosch GmbH

Frank Bonarens / Stellantis, Opel Automobile GmbH

Lydia Gauerhof / Robert Bosch GmbH

Christian Heinzemann / Robert Bosch GmbH

Christian Hellert / Continental AG

Martin Herrmann / Robert Bosch GmbH

Bernhard Knie / Volkswagen AG

Iwo Kurzidem / Fraunhofer IKS

Nicolas Gay / DXC Luxoft

Michael Mock / Fraunhofer IAIS

Christian Pfister / Automotive Safety Technologies GmbH

Andreas-Juergen Rohatschek / Robert Bosch GmbH

Martin Schels / Continental AG

Michael Schuldes / RWTH Aachen

Thomas Schulik / ZF Friedrichshafen AG

Gesina Schwalbe / Continental AG

Thomas Stauner / BMW Group

Reviewer

Frédérik Blank / Robert Bosch GmbH

Andreas-Juergen Rohatschek / Robert Bosch GmbH

Stephan Scholz / Volkswagen AG

Michael Mock / Fraunhofer IAIS

Maram Akila / Fraunhofer IAIS

Frédérik Blank / Robert Bosch GmbH

Ruby Moritz / Volkswagen AG

Christian Pfister / Automotive Safety Technologies GmbH

Andreas Rohatschek / Robert Bosch GmbH

Martin Schels / Continental AG

Thomas Schulik / ZF Friedrichshafen AG

Thomas Stauner / BMW Group

Christian Witt / Valeo Schalter und Sensoren GmbH

Kontakt

(in Vertretung für die Projektkoordination)

European Center for Information and Communication Technologies – EICT GmbH

EUREF-Campus Haus 13

Torgauer Straße 12-15

10829 Berlin

Germany

Email: projects@eict.de

Projektwebsite: https://www.ki-absicherung-projekt.de/

Revisionslog

Version	Datum	Kommentar	Autor	Partner
0.1	Bis 21.10.2022	Input auf Confluence	s.o.	s.o.
0.2	25.-26.10.2022	Ausspielen Word, Prüfung und Korrektur Übertrag, Strukturierung und Layout Dokument	Bert Hildebrandt	EICT
0.3	26.10.2022	Formelles Review und Layout	Frank Bauer, Dr. Nikos Papamichail	EICT
1.0	03.11.2022	Finalisierung	Dr. Nikos Papamichail	EICT

Inhaltsverzeichnis

1 AP4.1 Strukturierung und Formalisierung des Eingaberaums

1.1 E4.1.1 Final: Definition des Grundkontexts (zur Veröffentlichung)

1.1.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Report*
Group/Cluster	*C - Spezifikation; Definition; Auswahl / Übersicht / Katalog / Zuordnung; Template; Anforderung; Taxonomie*
Type of content	*Definition*
Classification level	*PU*

1.1.2 Description of the result

1.1.2.1 Motivation

In E4.1.1 we model realistic urban traffic scenarios - so called *base contexts* - suitable for pedestrian detection tasks. These models are to be used to generate synthetic data for training, testing, and assuring DNNs. In order to model base contexts it is required to define the elements of the scenery that we do not or cannot vary, like intersections, roads, lanes, traffic infrastructure, buildings, vegetation, etc.

A base context should provide enough information to allow the generation of variances of a specific set of scenarios that offer a further evaluation of the AI function. Hence, it contains e.g. all information on infrastructure elements (road geometry, road markings, buildings, traffic lights, etc.) similar to layer 1 to 3 in Pegasus.

1.1.2.2 Finding and Deriving Base Contexts

Our approach towards selecting and designing base contexts was based on multiple requirements. Based on input from Euro NCAP scenarios, AP2.2 Corner Cases and AP2.5 Data Generation we approached those contexts that include relevant features and pose no inappropriate difficulties in their generation. The main requirements were urban scenery (high density of buildings along the road), many road side objects (parked cars, bus stops, ...) for any kinds of occlusion, high variation of backgrounds (different buildings/facades/vegetation), different light and contrast variations. Not all requirements could be realized within the project, e.g. only natural lighting (sun), dry weather and no road cover (snow, leaves, ...) was supported by the data producers. However it was possible to specify the sun position, cloudiness and wet road surfaces. A detailed list of requirements is available on this page: Groundcontext requirements (summary). A very high rated requirement is that base contexts allow for occlusions of various kinds, e.g. behind parked cars, driving cars, trees, corners of buildings, poles and other road infrastructure items. While we do not define the assets (bus stops, vehicles, road side objects, ...) themselves within E4.1.1, we have to allocate sufficient space for them in the base contexts by introducing corners, intersections, wide enough streets and sidewalks and slopes which can block the view upon a pedestrian.

We gathered a list of in total 18 possible base contexts and derived relevant metrics from discussions with P1 (see also prioritization for tranche#3, tranche#4, tranche#5). These metrics were used to rank the base contexts to allow for an informed decision on which base contexts will be used for further data generation.

A data analysis on the first tranches showed that the distribution of pedestrian was anything but even in the generated images. This led to an improved base context design with more slopes along the streets of an intersection. These slopes allow pedestrian assets to be placed in the upper or lower part of the image and not just along the horizon line as in the previous data sets.

1.1.2.3 Base Contexts for KI-Absicherung

Within KI-Absicherung it was decided to use synthetic and real road networks as base context. Both have their pros and cons. Base context based on real road networks are very close to reality. Synthetic base context on the other hand can be designed and modified to represent certain aspects which are difficult to find in reality. While the data producer BIT-TS focused on synthetic maps, based on ASAM OpenDrive map standard, MackeVision concentrated its work on a real junction in Leonberg, Germany. The OpenDrive maps for BIT-TS were generated by Bosch, the Leonberg junction was photogrammetric digitized.

In the later tranches the data producers enabled to modify static scene elements sequence-by-sequence or even frame-by-frame. This feature allowed to produce frames with different road surfaces, road qualities, road markings and even different buildings while still using the same base context.

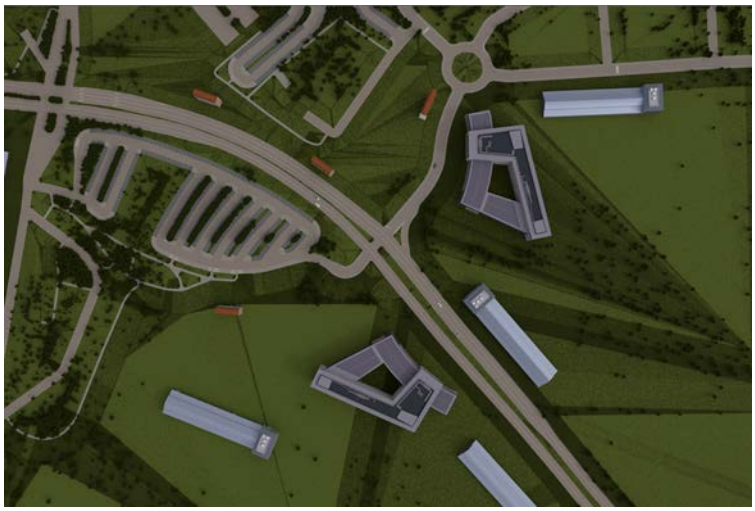
1.1.2.4 BIT-TS

1.1.2.4.1 Cooperation BIT-TS / Bosch

Bosch developed synthetic 4-way junctions in OpenDrive format, which usage started in tranche 3. BIT-TS placed buildings, vegetation and road furniture along the roads to complete the base context.

Over the project runtime several base contexts have been developed by Bosch and were provided to BIT-TS. For every tranche at least two different versions were developed: small urban junctions with two lanes and bigger junctions with four lanes. For each tranche some specialties were added, like more curves, additional pedestrian crossings and more road elevation profiles. The most recent versions of the base context are depicted below.

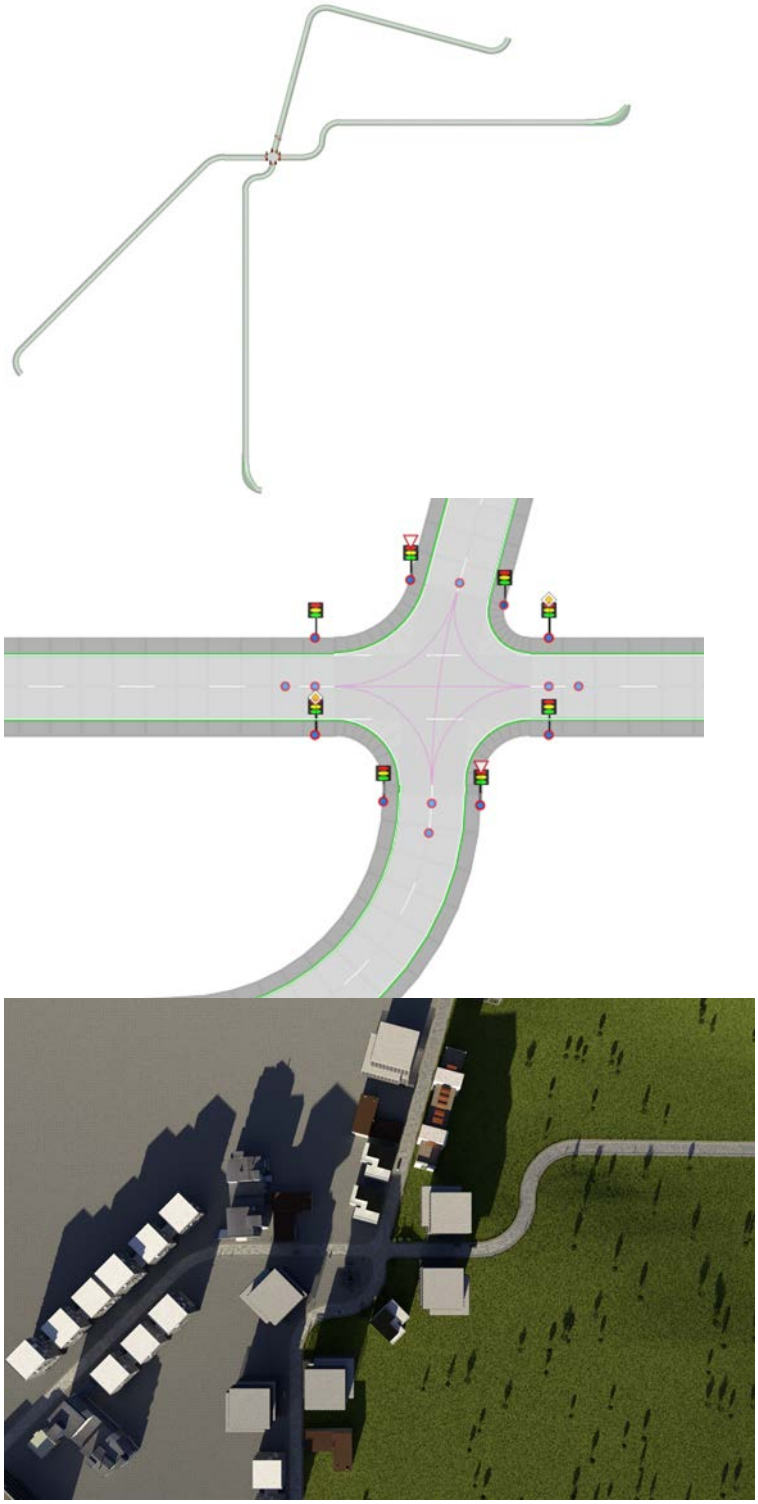
Tranche 1



Tranche 2

Tranche 3

Synthetic junction with 2 lanes in OpenDrive format generated by Bosch. Only the east road has a elevation profile (slope down). The east and south road have a s-shaped curve (2 x 90°), the west and north road are generated with a single curve. In this tranche all roads end with a sharp bend equipped with buildings to minimize the sky area and to provide backgrounds as diverse as possible.



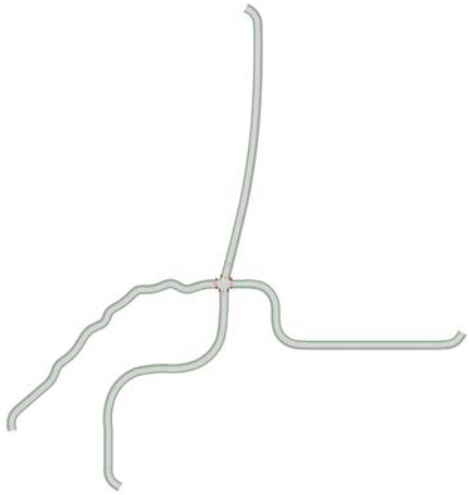
Figures: Overview, detail, and bird eye view with buildings and textures of 4-way junction with 2 lanes and traffic lights, priority road east to west.

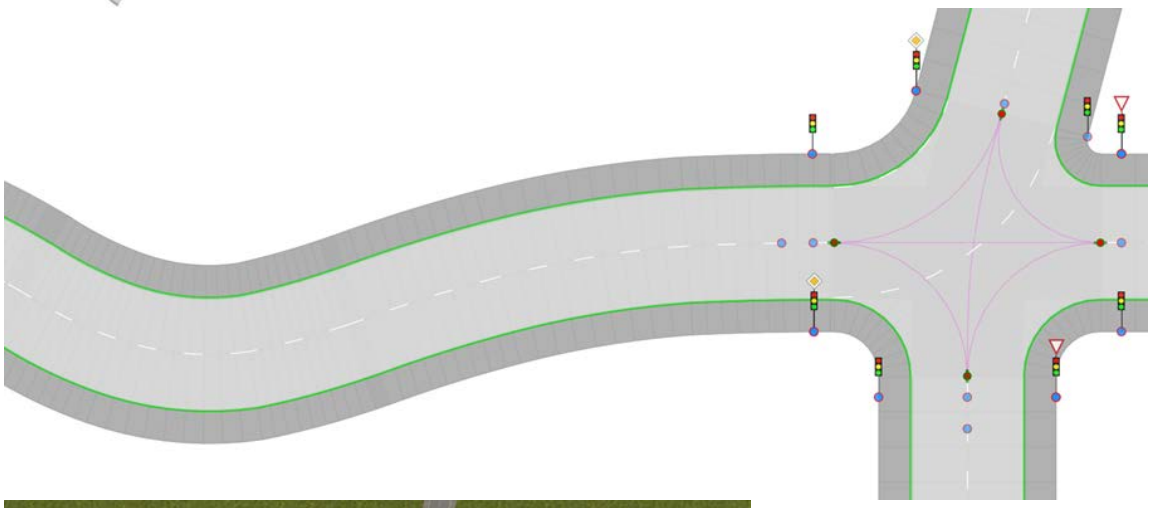
Tranche 4

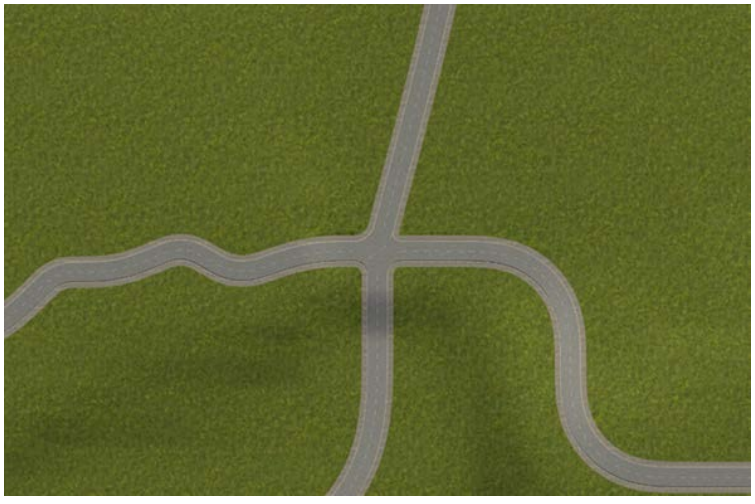
For tranche 4 BIT-TS reused the OpenDrive map from tranche 3.

Tranche 5

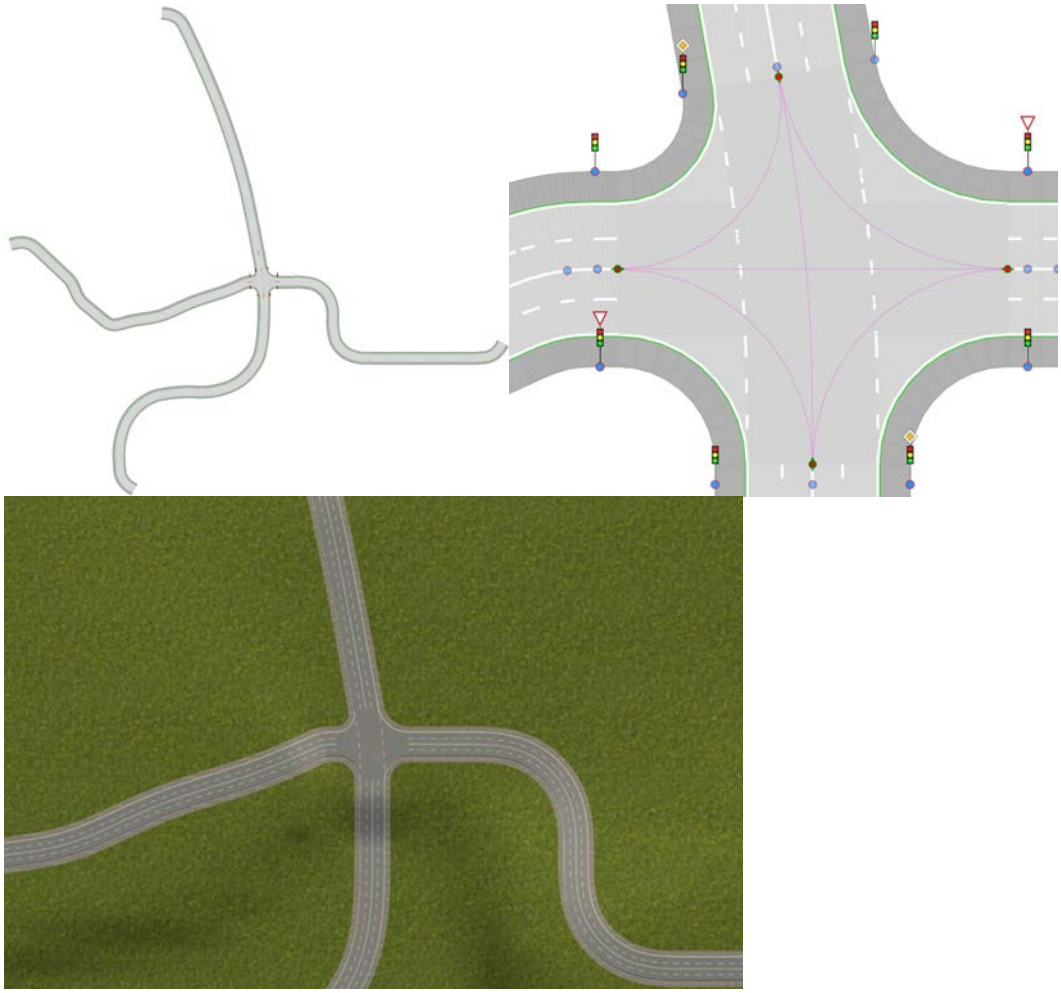
Synthetic junctions with 2 and 4 lanes in OpenDrive format generated by Bosch. The roads have a complex elevation profile with slopes up to 8%. The east and south road have a s-shaped curve (2 x 90°), the west road is generated with random curve pattern. In this tranche all roads end with a sharp bend equipped with buildings to minimize the sky area and to provide backgrounds as diverse as possible.







Figures: Overview, detail, and elevation profile of 4-way junction with 2 lanes and traffic lights, priority road north to west.



Figures: Overview, detail, and elevation profile of 4-way junction with 4 lanes and traffic lights, priority road north to south.

The OpenDrive maps are available in this Repo:
https://luxproject.luxoft.com/stash/projects/KIA/repos/tp4_ap4.1_ontology/browse/groundcontext

Based on these OpenDrive maps BIT-TS generated the following synthetic image:

Figure: Scene rendered by BIT-TS based on the 4-lane OpenDrive base context from Bosch

1.1.2.5 MackeVision

Real junction photogrammetric digitized in Leonberg, Baden-Württemberg, Germany which was used for all tranches by MackeVision.

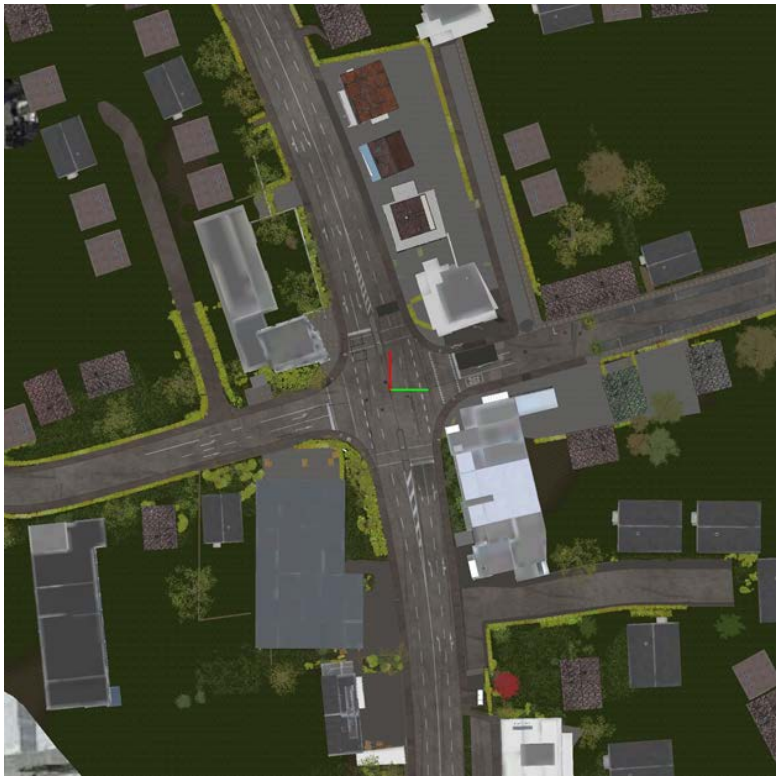


Figure: Junction in Leonberg as used by MackeVision

Based on this map MackeVision generated the following synthetic image:

Figure: Scene rendered by MackeVision based on the Leonberg junction base context

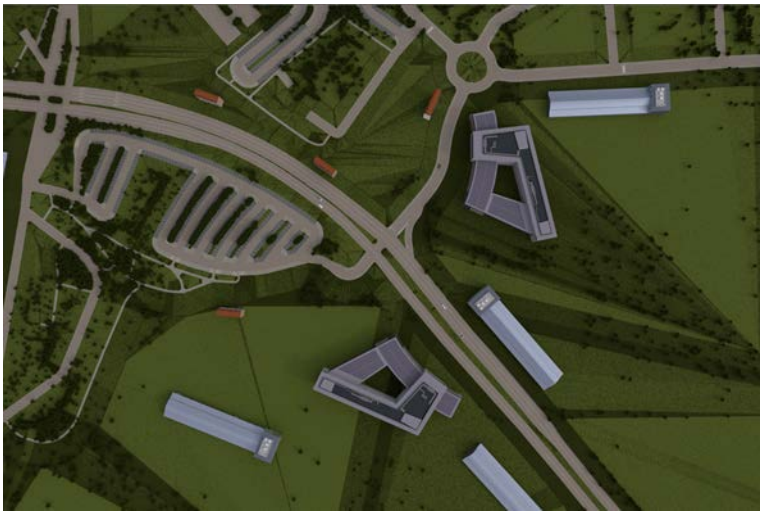
Additional information and images of different base contexts for different tranches are also available on this report: 4.1.4b Ergebnisbericht (M30) - Enriched metainfo description format

1.1.3 Final Results and Conclusions

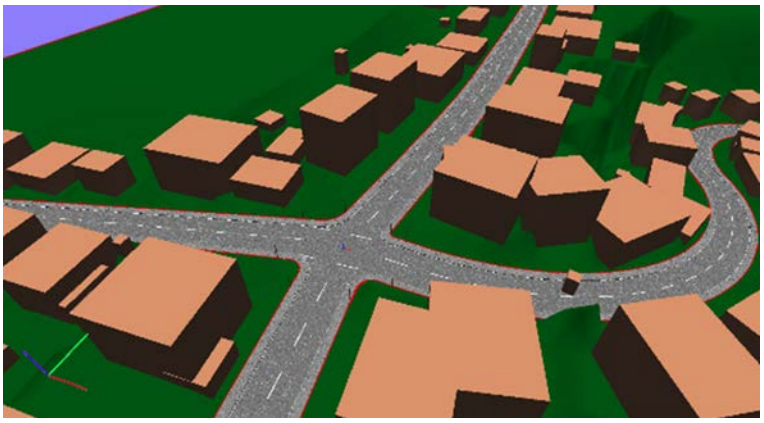
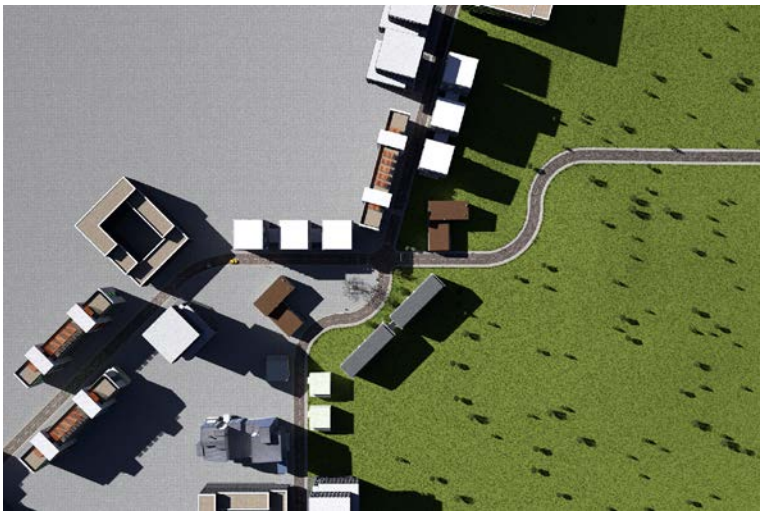
Most of the base context requirements could be fulfilled. With the results of work package 4.1.1 the data producers BIT-TS and MackeVision have been enabled to produce datasets with several 100.000 images. The datasets show a very high variation in all aspects, analysis show e.g. that all kind of pedestrian occlusions and contrast values are available.

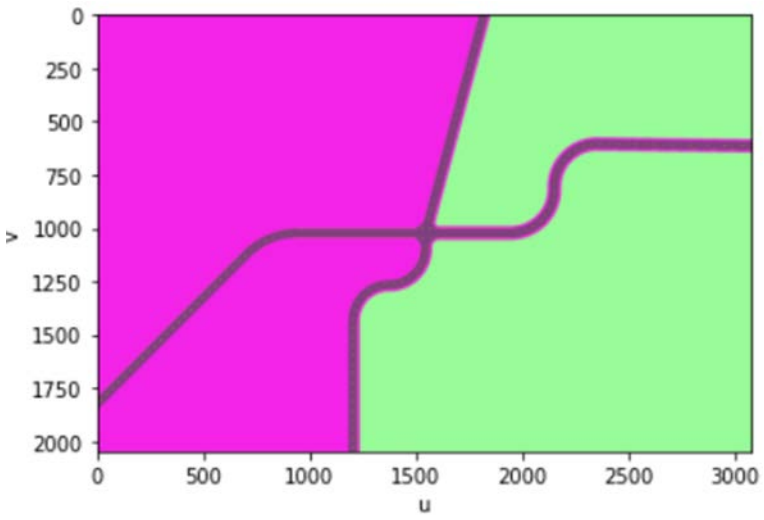
The ASAM standard OpenDrive proofed to be a very useful format to design and exchange base contexts / road network maps.

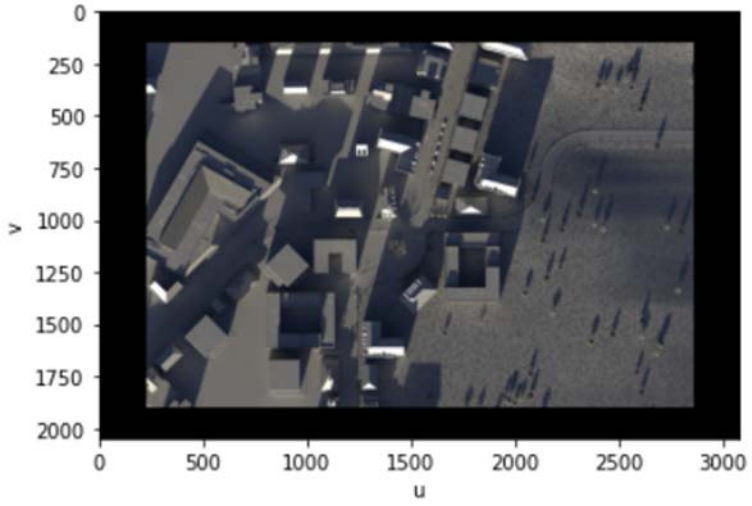
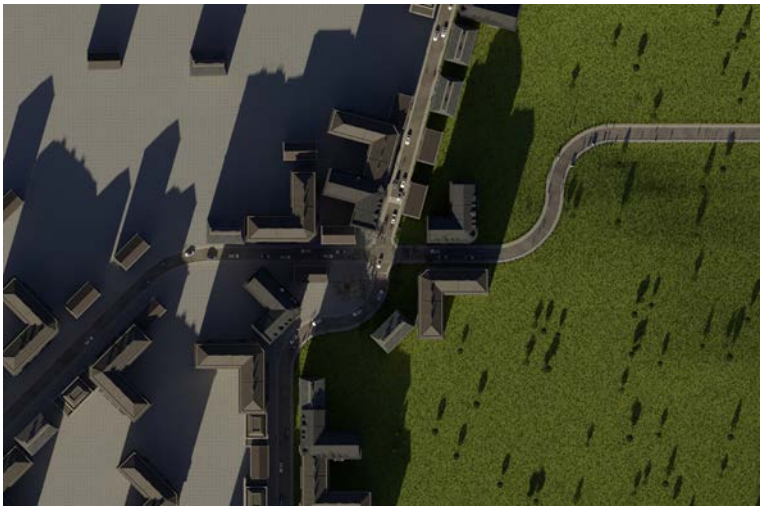
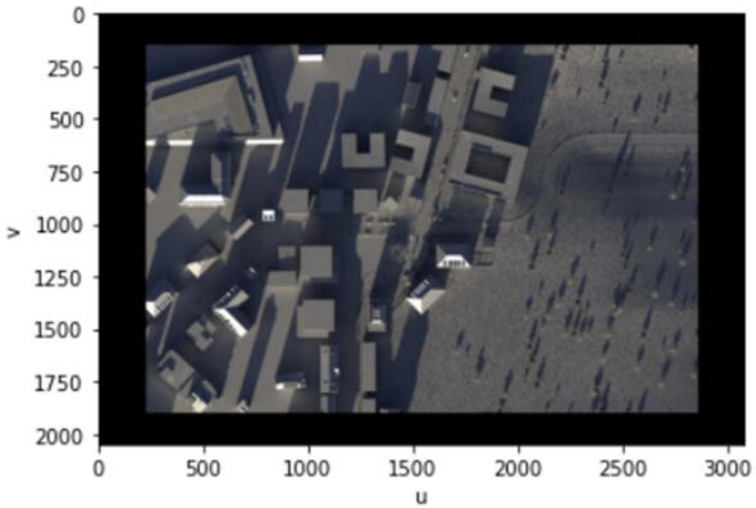
1.1.4 Example images of Base Contexts

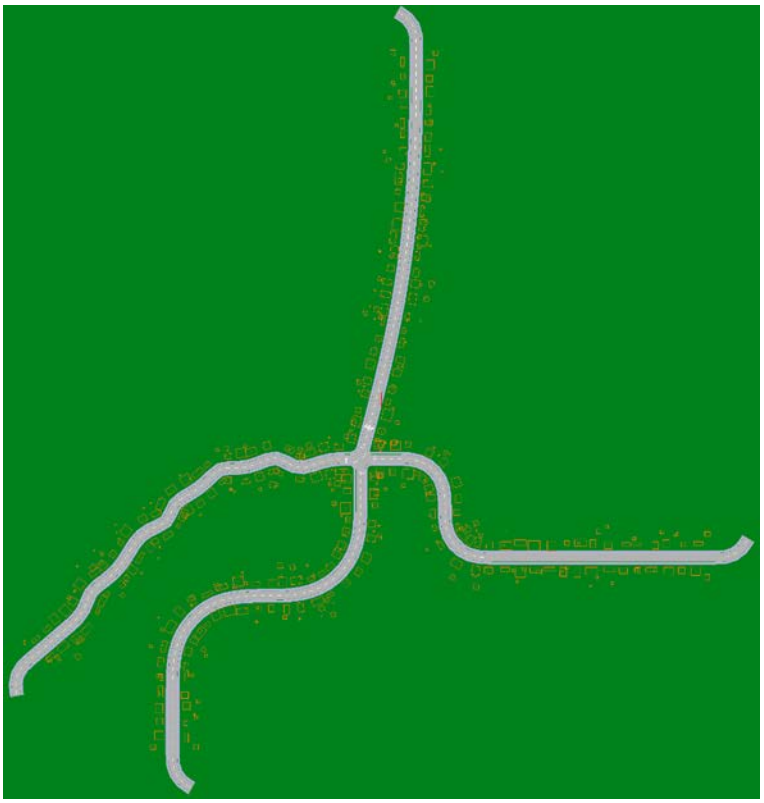
BIT-Tranche#01 - OpenDriveFile	not available
BIT-Tranche#01 - Example1	
BIT-Tranche#01 - Example2	
BIT-Tranche#01 - Example3	
BIT-Tranche#01 - Semantic Segmentation	not available

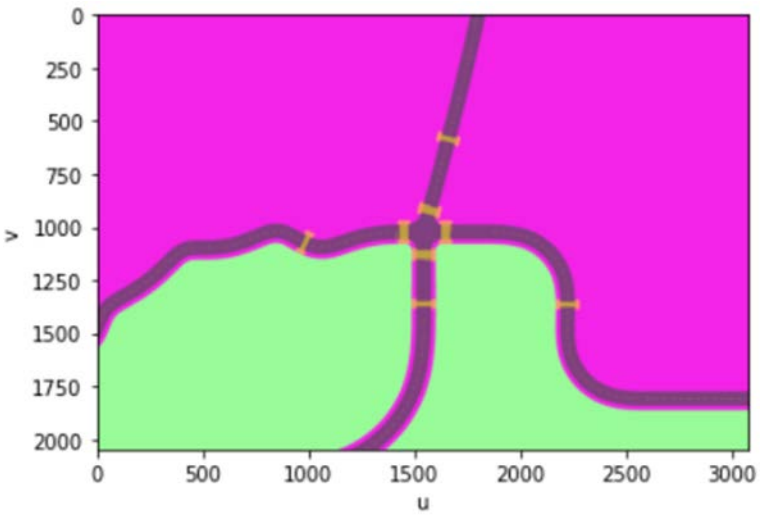
BIT-Tranche#02 (openDrive-File)	not available
BIT-Tranche#02 - Example1	
BIT-Tranche#02 - Example2	
BIT-Tranche#02 - Example3	
BIT-Tranche#02 - Semantic Segmentation	not available

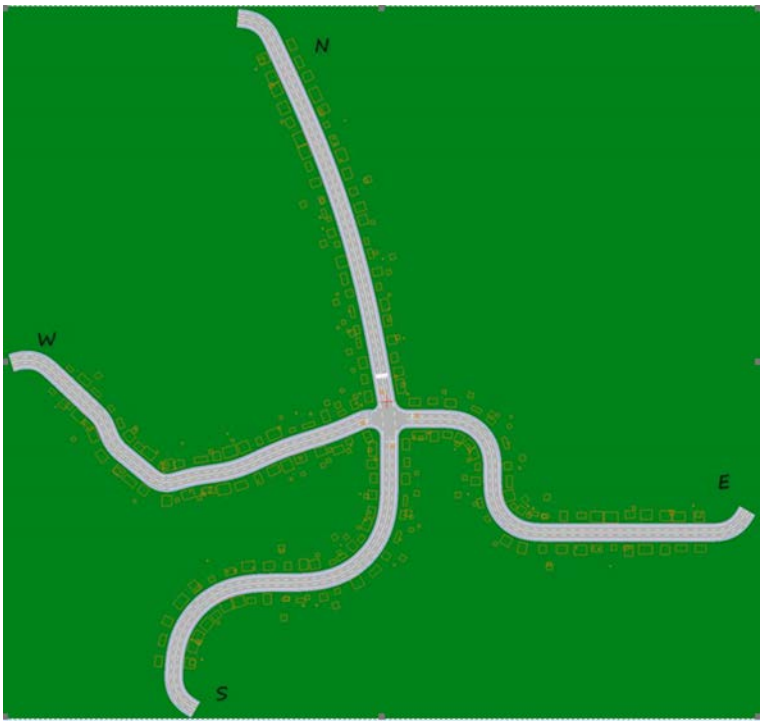
BIT-Tranche#03 (openDrive-File)	Bosch-generated OpenDriveFile as starting point  2x1L-City-Junction-TL-WE-3Ped-hill-curve-v6.xodr
BIT-Tranche#03 - Example1	
BIT-Tranche#03 - Example2	

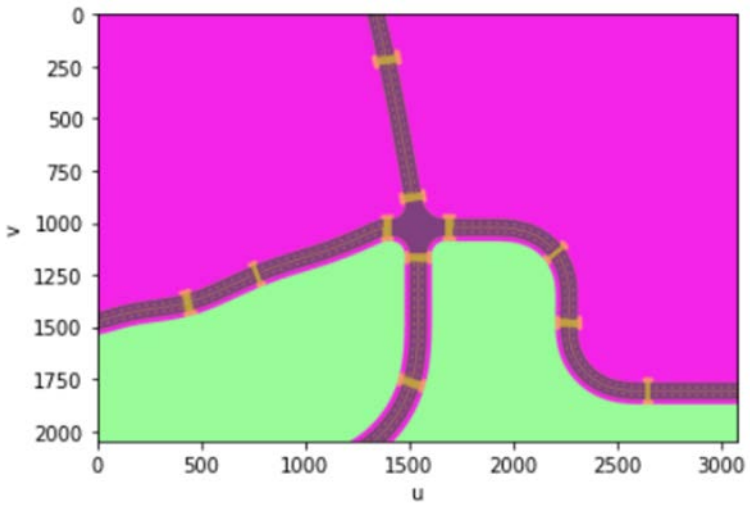
BIT-Tranche#03 - Example3	
BIT-Tranche#03 - Semantic Segmentation	

BIT-Tranche#04 (openDrive-File)	see tranche#03 openDrive file
BIT-Tranche#04 - Example1 (Seq: 0147)	
BIT-Tranche#04 - Example2	
BIT-Tranche#04 - Example3 (Seq: 0149)	
BIT-Tranche#04 - Semantic Segmentation	see tranche#03

BIT-Tranche#05 (openDrive-File1)	 KIA-Bosch-Groundcontext-Tranche5-2L-S-EndCurve.xodr
BIT-Tranche#05 - Example1	

BIT-Tranche#05 - Example2	
BIT-Tranche#05 - Example3	
BIT-Tranche#05 - Semantic Segmentation1	

BIT-Tranche#05 (openDrive-File2)	 KIA-Bosch-Groundcontext-Tranche5-4L-S-EndCurve.xodr
BIT-Tranche#05 - Example4	
BIT-Tranche#05 - Example5	

BIT-Tranche#05 - Example6	
BIT-Tranche#05 - Semantic Segmentation2	

1.2 E4.1.2a Final: Strategie zur Analyse des absicherungsrelevanten Eingaberaum

(zur Veröffentlichung)

1.2.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Report*
Group/Cluster	*E - Modell; Methodik; Konzept*
Type of content	*Methodik*
Classification level	*PU*

1.2.2 Description of the result

1.2.2.1 Motivation

A base for a safety and assurance case for an AI-based perception function is a detailed description of the functions input domain, also called its context. Such a description of the input domain shall provide a detailed specification of the environment in which the function is used

later on, i.e., it defines intended operating conditions and functional system boundaries for the assurance case. In addition, it forms a base for the design and analysis of datasets and safety related tests to ensure that all relevant domain elements are well represented. The different uses of such a description of the input domain require that it is both understandable by human (safety) engineers and amenable to computer-aided processing such that it can be used, for example, for data set analysis or systematic data generation.

Before the start of the project, there existed no established procedure on how to develop a description of the input domain and how to represent it. In AP4.1, our goal was to close this gap and to develop such an input domain model for the use case of pedestrian detection in urban environments that we pursue in KI Absicherung.

In this project result, we focus on the strategy for analyzing the input domain of the AI function that defines a structured and iterative approach for creating a domain model. The resulting input domain model for the pedestrian detection use case is described in the project result E4.1.2b and the ontology that we derived therefrom is part of E4.1.4a. Along with the strategy, we also document important design decisions that need to be made when creating an input domain model, particularly with the ultimate goal in mind of representing the input domain model as an ontology.

1.2.2.2 Input Domain

One major task of work package 4.1 is finding a common language to speak about different domains. The main scope of this work package is accurate structuring of the AI input domain (called the context) and identification of relevant context dimensions that need to be described for the domain at hand. As shown in Figure 1, there are three input domains or contexts that are important to be analyzed, which partly overlap, but not necessarily.

1. How does the data look like, that should be generated? (Input for simulation, has to allow high variance)
2. What are assurance-relevant input dimensions? (relevant for definitions of assumptions and guarantees for the assurance case and specific tests)
3. How does the generated data look like? (labels, metadata, additional information)

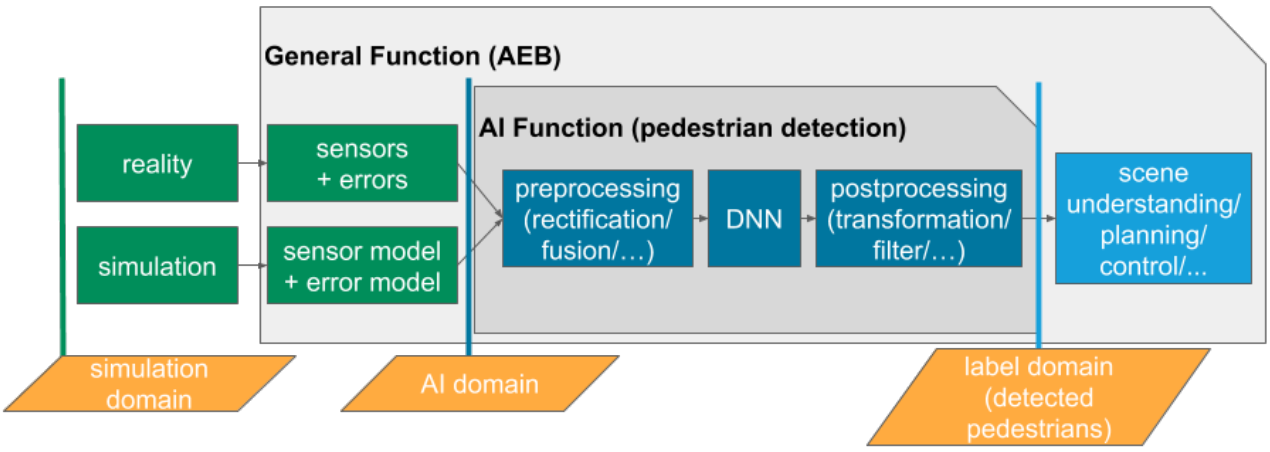


Figure 1: Overview about where different domains are located within the general function

The simulation domain is the interface to the data generation in TP2. This domain is analyzed primarily as part of the process P1 and shall enable to specify data that shall be rendered. The AI domain is relevant for the assurance case and for testing of the AI function. The results of analyzing the AI domain will be primarily used in TP4. The label domain is analyzed and described by work package 1.2. In this project result, focus is given to the analysis of the simulation domain and the AI domain.

1.2.2.3 Strategy for Analyzing the Input Domain

In the following, we present the strategy that we developed in AP4.1 for analyzing the input domain of an AI function. The strategy has been successfully applied within the project for deriving the input domain model described in project result E4.1.2b. It consists of the following seven steps:

1. Review of public data sources and existing standards
2. Brainstormings with experts from different fields of expertise
3. Structuring of initial results in context element categories
4. Expert Interviews for each context element category
5. Consolidation of context elements
6. Grounding of Context Elements and Variations in Physical / Measurable Units
7. Iterative refinement based on data analysis, assurance case, and test

In the following, we describe each of these steps in more detail.

1.2.2.3.1 Review of Public Data Sources and Existing Standards

In the first step, existing standards targeting (part of) the domains under analysis should be reviewed. The reason behind this step is that a standard usually captures long term experience about important things to consider for a particular task and model. Even though there is no standard for analysis of AI functions, there are existing standard formats targeting the description of parts of the relevant input domains. Reviewing such standards should be complemented by general (internet) search for errors of perception functions as these often give hints about what went wrong for other perception functions.

In our project use case, we aim at pedestrian detection at urban crossings with camera and lidar systems. Consequently, we reviewed standards regarding the specification of traffic scenarios like OpenDrive and OpenScenario. In addition, perception is intuitively concerned with the surface properties of the objects that need to be detected such that we also considered standards like glTF. Further documents that might be considered are normative and law documents that restrict, e.g., the appearance and positioning of traffic infrastructure like traffic lights, traffic signs, etc.

In addition, the GIDAS (German In-Depth Accident Study) database has been analyzed. The accident classes used in GIDAS for accident classification may serve as a high-level grouping of scenarios. Examples of such classes are e.g. “471 – traverse, pedestrian from right after junction” or “221 – traverse, left-turning vehicle and pedestrian in same direction”. We came to the conclusion that GIDAS currently does not help to identify corner case for AI in perception,

because the figures primarily indicate an increased likelihood of pedestrian accidents in case where the driver's attention is focused on a different driving task, like crossing a junction.

1.2.2.3.2 Brainstormings with Experts from Different Fields of Expertise

In parallel to reviewing public data sources, we recommend to conduct brainstorming sessions with experts from different fields of expertise. In particular, we queried AI experts, simulation experts, sensor experts, and safety experts to write down context dimensions that they consider important. In addition, the corner cases by work package 2.2 has been integrated in this step.

1.2.2.3.3 Structuring of Initial Results in Context Element Categories

Typically, already a high number of context elements from the input domain can be derived from the two initial steps of the analysis. Working with and maintaining such a list in a flat manner is difficult. Therefore, we recommend to derive a structuring of the obtained context elements into different categories from the target domain. Since we focus on detecting pedestrians as part of traffic situations, we used the layer-based structuring of a road-based context from the Pegasus project [2]. Since the Pegasus project had no focus on perception and, in particular, on AI-based perception functions, we extended and refined the Pegasus proposal to the following categories of context elements listed in Table 1.

Legend: Base Context | Variations | Excluded

Layer	Description	Examples of Context Elements
1	Road-Level	Junction, lanes, markers, shape x-y-z, road side objects (e.g., houses, advertising pillar, ...) -> coordinates + orientation
2	Traffic Infrastructure	signs, poles, traffic islands, bus stops, ...
3	Temporary manipulation of 1 and 2	Construction zones, stickers on traffic light / sign, ...
4	Objects	Assets
4.1	Subjects (actively and deliberately moving)	Car, Truck, Bus, Pedestrian (Group of x) including trajectory + position + animation (walking style like running, walking, drunk, ...) + full-body poses over the time
4.1.1	Pedestrians (Example for variations of subjects)	• skin color/structure • face shape • eyes/mouth/lips/nose • hair/facial hair • body (shape, posture) • ...
4.2	Objects (passively moveable):	• overridable vs. not overridable (trash, lost cargo, leaves, ...)

Layer	Description	Examples of Context Elements
		• fixed at one point, but moving in itself: vegetation (trees, plants, …), flags
5	Environment	
5.1	Weather	sun, cloud, rain, snow, wind, fog, lightning/thunder, …
5.2	Light sources	brightness, angle, …
6	Digital Information	Digital map, car-to-car communication, …
7	Materials	
7.1	Surface characteristics	• Object surfaces: reflectance, brightness, conditions for each object and sensor modality • Road surface: asphalt, sand, grit, wetness, snow, puddle
7.2	Material characteristics/properties	texture, fabrics,….
8	Sensor characteristics	resolution, distortion

Table 1: Proposed categorization of context elements.

The listed context elements for each layer are only meant as examples, a full description of context elements in each category is given as part of project result E4.1.2b.

The basic idea of the categories is that each layer adds objects and information to the lower layers. Layer 1, road-level, defines the basic road infrastructure consisting of roads with lanes and markings. Layer 2, traffic infrastructure, adds traffic related objects like traffic lights, traffic signs and so on to the road infrastructure. Layer 3, temporary manipulation, adds, for example, construction zones to the description. The result of Layers 1-3 is a scenery. Layer 4, objects, places subjects and objects[1] on the roads and defines their behavior. The result of combining the scenery with the information from Layer 4 yields a scenario description. Layer 5, environment, adds weather effects and lighting to the scenario. Lighting particularly involves a description of the properties of each light source in the scenario. Layer 6, digital information, from Pegasus is not used as it does not contribute to perception of pedestrians with help of camera and LiDAR systems. Layer 7, materials, is an addition compared to Pegasus and adds information of the surface characteristics and material properties to all subjects and objects in the scenario. This also involves the road surface. Finally, Layer 8, sensor characteristics, adds information about the sensors placed in the scenario that are used to capture the scenario during data generation.

A key benefit of the layer separation is that the same scenery (e.g., a crossing) can be used to realize different scenarios (e.g., a left turn, a right turn, with and without pedestrians, …). Similarly, the same scenario can be evaluated under different weather and lighting conditions.

1.2.2.3.3.1 Base Context & Variations

As the legend of Table 1 shows, we distinguish between *base context* elements and *variations*. This distinction is specific for the setting in KI Absicherung and it needs to be revisited for applications of this strategy to other use cases. With base context, we refer to elements of the scenery that we do not or cannot vary, like the intersections that we use for data production. These are collected in project result E4.1.1. Basically all elements that can be described based on a map and which are fixed for multiple data sets are part of the *base context*. All other elements can be varied and are therefore *variations* of or inside a *base context.*

1.2.2.3.4 Expert Interviews for each Context Element Category

The initial set of context element categories and context elements therein needs to be refined in expert interviews with AI experts, sensor experts, and simulation experts. The goals of this step are 1) to identify further important context dimensions, 2) to derive a number of variations for each context dimension, and 3) to get a first expert opinion on importance of the different context element categories.

For each of the context element categories, Bosch conducted a number of expert interviews for extending the list of important context dimensions. We found it to be easier to first collect different context element categories from examples without yet discussing their possible variations. In a second step, possible variations were defined for each context element. While we generally aim at finding a human understandable description as well as a machine-readable formalization, we recommend to restrict yourselves to human understandable description of context elements and their variations, i.e., machine-readable formalization should be carried out later.

At this stage, a first hypothesis or voting on which context elements are relevant from a safety point of view can be made and used for initial experiments. This can be handy because usually the dimensionality of the input domain is vast such that prioritization and explicit handling of high complexity are required. In our understanding, important context elements are those where variations have caused malfunctioning of AI functions in the past. While prioritization is based on past experiences and expert guesses in this step, information on important context elements needs to be complemented later on with experiments on actual data for the currently addressed AI function as we describe in the Section "Iterative Refinement based on Data Analysis, Assurance Case, and Test" below.

Due to the special setting in the project that we have to rely on synthetic data production that cannot yet handle all context elements that were collected, prioritization of context elements has been carried out as a two-step approach as part of the P1 process. First, context dimensions were assessed regarding the ability of data production to account for these context dimension in their rendering pipelines and asset libraries, either now or in near future. In a second step, this information was used to define an operational design domain (ODD) for the project and a prioritization of context elements within the ODD was performed. Even though prioritization was performed in this way, all identified context elements have been documented and made available for later investigation.

1.2.2.3.5 Consolidation of context elements

The consolidation of context elements is performed iteratively and alternating with expert interviews for different context element categories. The consolidation has two major goals: (1)

identifying shared context dimensions and (2) grouping coherent information while factoring out shared information. Performing such a consolidation is key for obtaining a usable and well-structured input domain model.

A prime example for a shared context dimension is color as color is used at many different locations in the input domain model. Examples include hair color of a human, color of object surfaces, colors of clothes, color of emitted light of a light source, etc.. For this reason, the description of color has been factored out into a separate model part.

Along with the consolidation, also relationships between different model parts resulting from the above mentioned context element categories have been documented and visualized in an overview figure. We provide an example of such a figure as part of the E4.1.2b report. Such a figure proved to be highly useful for providing an overview of the contents of the created input domain model.

1.2.2.3.6 Grounding of Context Elements and Variations in Physical / Measurable Units

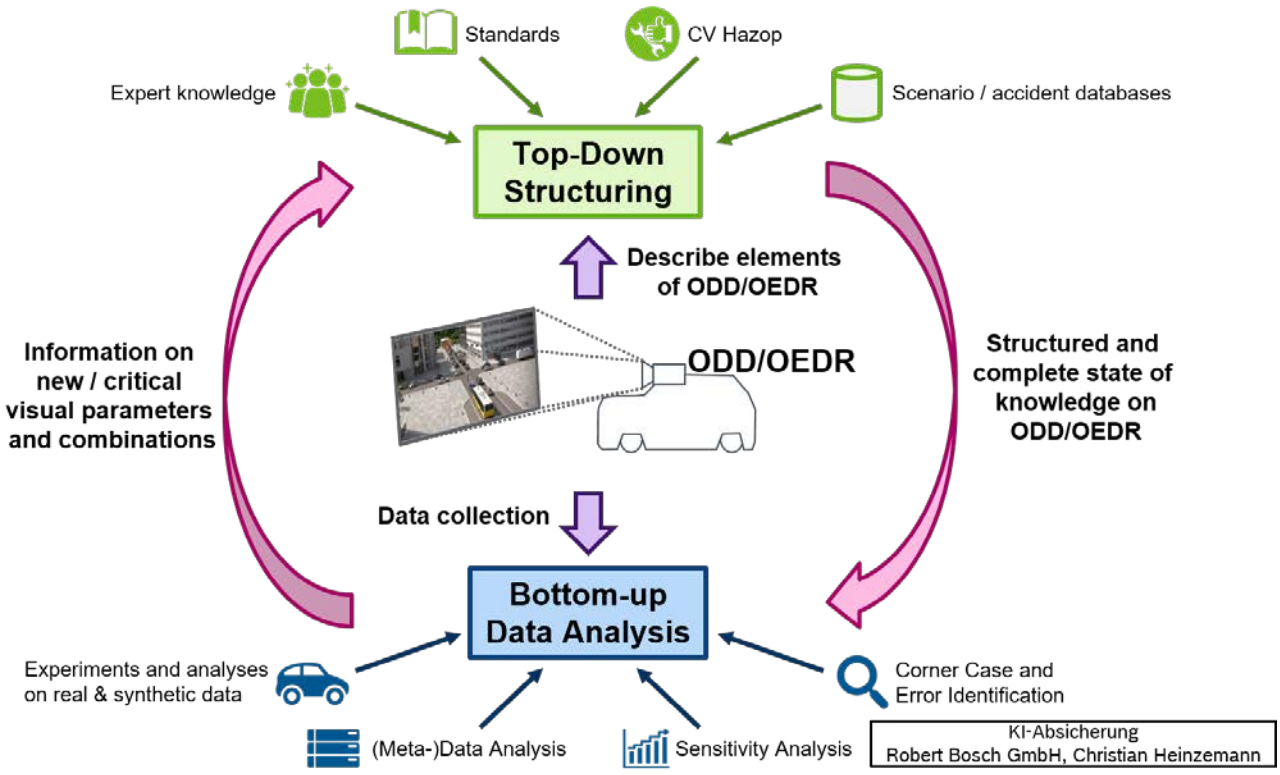
The input domain model shall also be machine-readable, for example, to support specifying data request for production of synthetic data or to label existing data. This requires a clear understanding of the meaning of each context dimension (in literature also referred to as *semantics*). Therefore, we performed a grounding of context dimensions in a physical and/or measurable quantity with a defined unit. The physical quantities and units should always be defined on context dimension in the first place.

An example is given by the context dimension "rain" in the weather conditions. For this context dimension, we chose the amount of water falling from the sky measured in the unit "mm/h" for grounding the context dimension. It is generally possible to additionally specify a probability distribution over the possible values for the context dimension if it is known. As stated above, we found it helpful to define a number of variations for each context dimension that capture in human understandable terms the possibilities how this context dimension may manifest in reality. Then, the different variations of the context dimension need to be associated to mutually exclusive values or intervals over the quantity. Wherever possible and useful, the grounding should be based on existing external definitions. In our example, the definition for light rain (ranging from 0.1 to 5 mm/h) was obtained from a definition of the German Weather Service (Deutscher Wetterdienst, DWD). The grounding should support objective measurement, either within a vehicle or by reference sensors for being able to associate concrete images to variations.

When using external definitions and/or human judgement for defining and grounding variations over intervals of physical quantities, one has to be aware that these equivalence classes might not represent equivalence classes for the performance of the neural network. While we believe that such equivalence classes are useful, e.g., for defining input coverage, it might be more suitable to resort to physical quantities or distributions for defining the assurance case.

1.2.2.3.7 Iterative Refinement based on Data Analysis, Assurance Case, and Test

After performing the above mentioned steps, a first version of the input domain model has been derived. At this stage, the creation of an input domain model needs to enter an iterative process as shown in the figure below.



The initial creation of the domain model provides a top-down structuring of available knowledge on important context element categories and their variations. As described in the previous sections, this knowledge originates from expert knowledge and other available data sources. This knowledge needs to be challenged and substantiated by a bottom-up data analyses that either confirm or refute the importance of context dimensions and that may also identify additional context dimensions that need to be taken into consideration.

Within the project, such information has been obtained based on different results, for example of work packages 2.2 (corner cases and data difficulty estimation), 2.4 (data quality) and 4.4 (test methods, mainly as part of the Evidence Workstream 04 - Performance Limiting Factors). Based on the results, some expert opinions have been confirmed and additional important aspects to consider have been identified.

1.2.2.4 Choosing Suitable Knowledge Representation Formats

So far, our strategy for analyzing the input space of the AI function was centered around identifying the important context dimensions and their possible variations and their grounding in physical units of measurement. Of course, this gathered knowledge needs to be represented in a meaningful way that supports further downstream tasks. To this end, there exist multiple possibilities for representing the gathered context dimensions where a general recommendation which is "the best" is not possible.

As always, the choice for a representation format and/or tooling significantly depends on what shall be done with context dimensions. Depending on the use cases, it may also make sense to support more than one representation. If more than one representation is used, these should be linked and, ideally, one representation should be the central artifact while all other representations are derived from the central one (preferably in an automated fashion). In the following, we provide some examples for possible representation formats including use cases that they could support.

- Combinatorial Models
    - Combinatorial Models represent the context dimensions and their discretized variations as a Cartesian space. They typically allow to exclude meaningless combinations, but do not provide further links or references between the context dimensions. An example in this regard is SCODE. Combinatorial models are suitable, for example, for combinatorial input coverage as detailed in result E4.4.3a and the identification of missing test data based on currently not covered combinations.
- Ontologies
    - Ontologies model the concepts in a domain and semantic relationships between these concepts. They are flexible and can be used for storing different kinds of knowledge. As such, they are particularly suitable for describing the semantic contents of (image) data including the semantic interplay of the contents (e.g., that a particular pedestrian holds a specific object thereby interacting with another pedestrian in a specific way). Ontologies can be complemented with reasoners for explicating knowledge that is implicitly contained in the ontology. In addition, query languages such as SPARQL can be used to perform semantic queries using the vocabulary from the ontology.
- JSON and JSON Schema
    - A JSON Schema defines the allowed structure of JSON files by defining allowed keys, how these keys may be nested, and data types that required values need to have. JSON is a preferred format for ground truth and meta-annotations for machine learning tasks due to its simplicity and good tool support for reading and writing JSON files in different languages including Python. Thereby, JSON supports a limited number of data types (booleans, numbers, strings), arrays, and hierarchical structuring (called objects in the specification), but does not directly support links or references between different objects.
- ...

In KI-Absicherung, we currently pursue combinatorial models, ontologies and JSON, each of which for a particular use case.

1.2.2.5 Derivation of an Ontology

An ontology is a machine-processable representation of concepts in a domain and of the semantic relationships of these concepts. The power of an ontology lies in the fact that it is based on formal logic and allows to apply reasoning for automatically inferring knowledge that is only implicitly contained in the currently defined knowledge (i.e., in the specified instances) about the domain. An ontology is one way to represent the knowledge gathered in terms of context dimensions and their variations. In this section, we will dig deeper into two possible ways for deriving an ontology given knowledge about the important context dimensions and their variations plus grounding in physical units of measurement. We coined these two ways as the *top-down approach* and the *bottom-up approach*, but these are no standard terms and there presumably are additional ways for creating an ontology. We provide more details on the general

pros and cons of both approaches in the following and refer to project part E4.1.2b for a detailed description on how the ontology was derived as part of the KI-A project.

Irrespective of the concrete approach that is chosen, we strongly recommend to first gather (at least a preliminary state of) knowledge about context dimensions and their variations that should be contained in the ontology.

1.2.2.5.1 Top-Down Approach

In the top-down approach, development starts from the most abstract concept *Thing* and the ontology is broken down from this level, i.e., a structuring into high-level concepts and subontologies is performed manually by the engineers. Ideally, also semantic relationships are already defined on the level of the superclasses corresponding to the high-level concepts. Here, the abstract concepts are created first and the concrete context dimensions including their variations are inserted at the end. Such an approach is always a manual one and it enables to specify a clean frame for the ontology from the start. However, the approach requires also to manually add all context dimensions and variations and to check whether they have been realized completely. The manual work bears the risk that context dimensions that are structurally similar are realized in different ways if they are modeled by different engineers. One particular example for such a case is the representation of domain elements that can also be not present in an image, which needs to be encoded explicitly due to the open-world assumption of ontologies. An example for this case are dimensions like "rain", "clouds", "scarf", etc., where the domain model uses an explicit alternative "No" indicating that no rain, clouds, scarf etc. are present. Due to the open-world assumption of an ontology not adding a scarf would not represent that the scarf is not existing. I would just represent that the existence (or non-existence) of the scarf is not known. Thus, a "No" needs to be represented in the ontology explicitly and there exist different possible realizations.

1.2.2.5.2 Bottom-Up Approach

In the bottom-up approach, the development starts from the concrete context dimensions, i.e., these are added first to the ontology. Then, the development proceeds by identifying suitable superclasses and relationships between the context dimensions. This approach is beneficial if the context dimensions have been gathered in a machine processable form such that the context dimensions can be added to the ontology using a generator approach. In this case, it can be ensured that context dimensions with a similar structure are all modeled in the same way and that the ontology is complete with respect to the gathered context dimensions. On the down-side, manually introducing superclasses in later stages might lead to increased effort (as all existing concepts need to be understood and taken into account) and lead to sub-optimal higher-level structures.

1.2.2.6 Dealing with Incremental Changes

The domain model is a so-called living artifact, i.e., it is subject to frequent changes as new information may be gathered during development and testing of an AI function, which is reflected by the iterative refinement step shown above. As described above, there are many different representation formats for the domain model, e.g., in SCODE, in JSON, in the ontology, etc. If we choose to use more than one representation format, than it is strongly advisable to mark one of these formats as the "master" format and to use the other representations only as derived formats. That means that all changes to the domain model are solely performed in the

master format and that consistent states of the master format are exported to the derived formats. If possible, these exports should be automated as much as possible by using generators.

1.2.3 Final Results and Conclusions

In AP4.1, we successfully defined and executed the strategy for analyzing the input domain of an AI-based function. The results of executing the process are the input domain model described as part of project result E4.1.2b and the ontology described as part of project result E4.1.4a. In this report, we summarized the main steps of the process and highlight relevant decisions that need to be taken for creating an input domain model. The strategy has been published along with a description of the ontology in a peer-reviewed conference paper *"Using ontologies for dataset engineering in automotive AI applications"* to appear at the DATE ADS 2022 conference.

1.3 E4.1.2b Final: Strukturierung des Eingaberaums (zur Veröffentlichung)

1.3.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Bericht*
Group/Cluster	*C - Spezifikation; Definition; Auswahl / Übersicht / Katalog / Zuordnung; Template; Anforderung; Taxonomie*
Type of content	*Katalog*
Classification level	*PU*

1.3.2 Description of the result

1.3.2.1 Motivation

A basis for a safety and assurance case for an AI-based perception function is a detailed description of the functions input domain, also called its context. Such a description of the input domain shall provide a detailed specification of the environment in which the function is used later on, i.e. it defines intended operating conditions and functional system boundaries for the assurance case. In addition, it forms a base for the design and analysis of datasets and safety related tests to ensure that all relevant domain elements are well represented. The different uses of such a description of the input domain require that it is both understandable by human (safety) engineers and amenable to computer-aided processing such that it can be used, for example, for data set analysis or systematic data generation.

Based on the strategy for analyzing the input domain developed in project result E4.1.2a, we conducted an in-depth analysis of the input domain of a pedestrian detection function for a vehicle driving in urban environment. Thus, this project result focuses on the actual application of the strategy and the development of the corresponding input domain model. We chose to structure the input domain model along the Pegasus layers and represent the model as a combinatorial model based on SCODE. This model then served as a basis for deriving an initial ontology for the input domain that was further refined as part of the project result E4.1.4a. In this report, we will provide further insights how we implemented the strategy and we will

provide a description of the resulting input domain model. The actual model with exports in different formats such as HTML and YAML is provided as part of project result E4.1.2c.

In the following, we first provide an overview how we implemented the strategy for analyzing the input domain. This includes, in particular, a description how we implemented the automated translation from SCODE Zwicky boxes to an ontology. Thereafter, we present the final input domain model that we collected in SCODE and outline the intent and the dependencies of the different Zwicky boxes. Finally, we describe where and how the input domain model (or parts of it) have been used in KI Absicherung.

1.3.2.2 Implementing the Strategy for Analyzing the Input Domain

For deriving the input domain model, we applied the strategy described as part of project result E4.1.2a. In this section, we will explain some design decisions and report on some lessons that we learned during our work. Figure 1 shows an overview of the artifacts that we used and created for the input domain model and the ontology derived thereof.

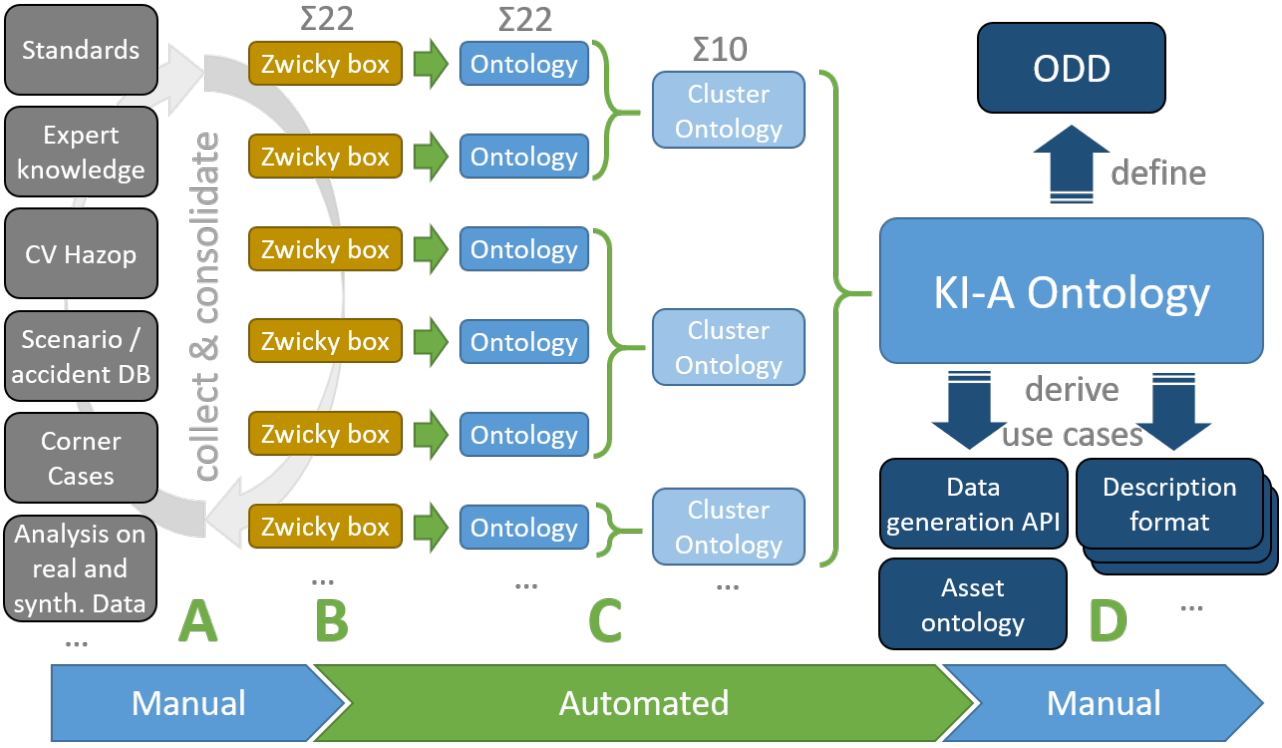


Figure 1: Overview of the Process of Developing the Input Domain Model and Deriving the Ontology

We executed the strategy in four phases named by the letters **A - D** in Figure 1. In the following sections, we will related these phases to the different steps of the strategy and describe the outcomes.

1.3.2.2.1 Collecting Context Elements

During phase **A**, we collected the context dimensions and their variations that should go into the input domain model. Thus, phase **A** spans the steps "Review of public data sources and existing standards", "brainstormings with experts from different fields of expertise", "structuring of initial results in context element categories", and "expert interviews for each context element category". As expected, the analysis of the standards did not provide too many insights as the

existing standards and databases are yet non perception-oriented. The initial brainstormings, however, were considered to be quite useful for creating an initial list of interesting elements. This list, in turn, allowed for identifying the relevant domain experts for specific topics. During this step, using the Pegasus layers for structuring the brainstorming and the initially collected elements proved to the quite useful. While it might be tempting to start the brainstorming with the Pegasus layers as a basis for the discussion, it might narrow the focus too much to the layers presented therein. Given that we identified two additional layers to be added to the Pegasus model, we believe that such structure should be better added after the initial brainstormings.

During the collection phase, we made an early decision to represent the collected context dimensions in morphological boxes, also called *Zwicky boxes* in the SCODE method. Zwicky boxes provide a very concise representation of combinatorial spaces and naturally allow to represent context elements and their variations. We used the Zwicky boxes intensively during the expert interviews to show the experts what has already been collected and to refine the gathered knowledge. Due to the compact representation, we were able to keep the overview about what has already been specified. At this stage, we only collected the context elements and variations in natural language without yet caring about their formalization, which allowed to progress faster with the expert interviews.

As an intermediate result, we yield a high number of Zwicky boxes with significant conceptual overlaps in between them that provided the input for the consolidation phase.

1.3.2.2.2 Consolidating Context Elements

During phase **B**, we consolidated the context dimensions and their variations. Thus, phase **B** spans the steps "Consolidation of context elements" and "grounding of context elements and variations in physical / measurable units". During the expert interviews, the initial Zwicky boxes have been expanded significantly. As a result, the intermediate Zwicky boxes often featured elements from different Pegasus layers and there was significant overlap between the Zwicky boxes as repeatedly occurring concepts like surfaces or colors appeared in many different Zwicky boxes. During this phase, the Zwicky boxes were rearranged and reworked such that they fitted again to the Pegasus layers and such that repeatedly occurring elements have been factored out into separate Zwicky boxes. Figure 2 provides an overview of the resulting 22 Zwicky boxes that built the resulting input domain model. We provide a more detailed description of the individual Zwicky boxes below in the Section "Final Input Domain Model". Along with the Zwicky boxes, Figure 2 also illustrated the conceptual relationships between the Zwicky boxes. Creating such a highlevel overview and formally defining the relationships on this coarse level first proved to be highly useful for the discussions and the subsequent work. Identifying physical quantities for grounding the context dimensions has been a major step for increasing acceptance of the input domain model by other work packages as it provided a clear understanding of what was meant by the individual terms in the model.

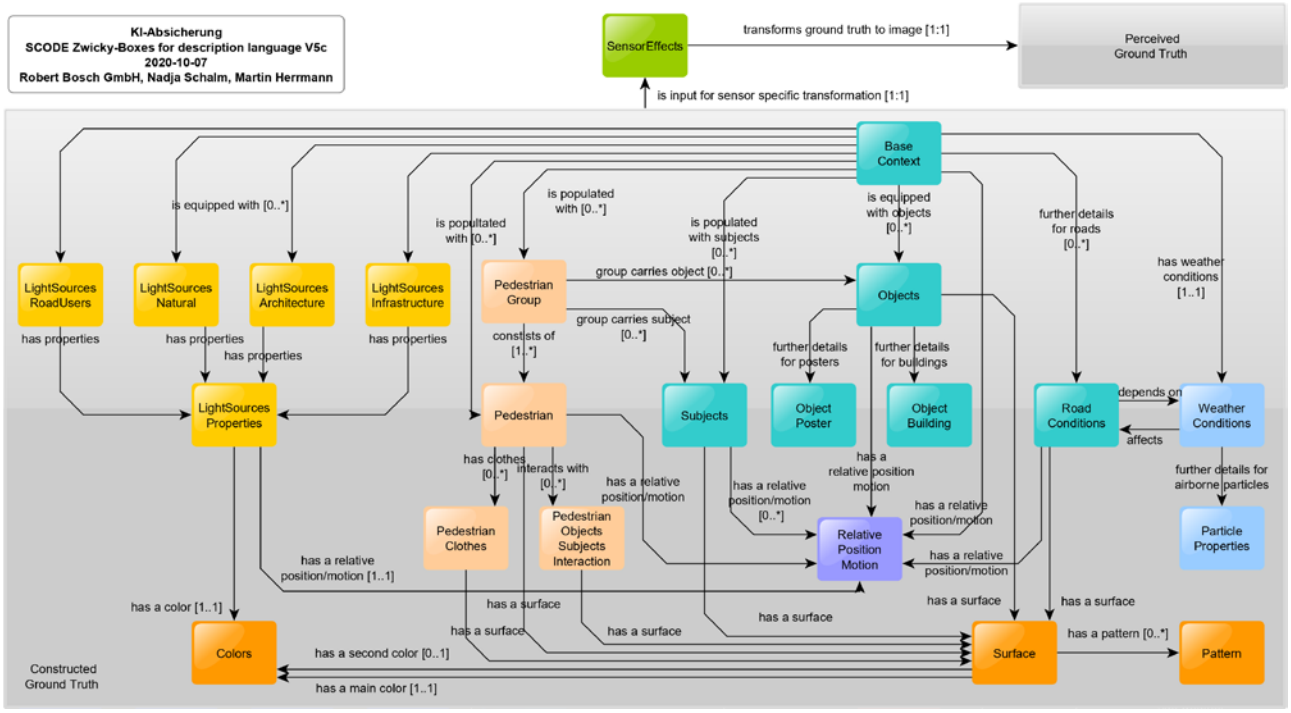


Figure 2: Overview Figure showing the relationships between the different Zwicky boxes

Please note the the process of collecting and consolidating context elements should be performed iteratively as also described in the strategy and as highlighted in Figure 1.

1.3.2.2.3 Generating an Ontology from the SCODE Model

In AP4.1, we decided to follow the bottom-up approach for deriving an ontology from the context dimensions collected in the SCODE model and the relations between the Zwicky boxes as shown in Figure 2 above.

The reasons for that decision were:

- The content of the Zwicky boxes was already used in other parts of the project (e.g. to define the ODD, annotate the images, combinatorial testing), so the goal was to preserve its content and structure in the ontology
- The effort to develop an ontology generator were estimated to be lower than to manually create an ontology based on 250 dimensions and 1000 alternatives
- We wanted to avoid double maintenance and thus to have only one master (the SCODE model in the first phase, and then later on the ontology)

As a consequence, we developed and implemented a generator during phase **C** for deriving the ontology in an automated fashion.

1.3.2.2.3.1 Design Considerations

The primary difference between an ontology and the SCODE model is the kind of information representation. The SCODE model is structured into *Zwicky boxes*, *Dimensions* and *Alternatives* and can be seen as a tree-alike structure with three different levels. This limited complexity allows the application of coverage analysis or combinatorial testing. The disadvantage of this structure is the limitation of knowledge representation, since more detailed structures and links

between dimensions or alternatives cannot directly be integrated. Ontologies on the other hand allow a much more complex graph-alike structure. It basically consists of *classes* that are in relation with each other (*object properties*) or store additional facts (*data properties*). A tree-alike structure can be realized as well as more detailed representations or links between different cluster of classes.

Apart from some exceptions (see annotations chapter below) we decided to map each dimension of a Zwicky box, Dimension and Alternative as classes in the ontology. The Zwicky boxes are stored as sub classes of http://www.w3.org/2002/07/owl#Thing. Dimensions are stored the same way, but are also linked to the original Zwicky boxes. Each alternative of a dimension is mapped as a subclass of its dimension class. All alternatives of such a dimension are defined as disjoint classes. In case that the alternatives can be described using specific value ranges, those ranges were specified using data properties and value restrictions within the classes.

For realizing the generation, we extended the Zwicky boxes by several annotations that control how the single dimensions are defined in the resulting ontology. Then, the actual ontology is generated in a 3-step approach (see also image below) in RDF format, which is a standard format for storing ontologies. We explain both of these aspects in more detail in the subsections below. The resulting ontology is discussed as part of the project result E4.1.4a.

1.3.2.2.3.2 Annotations in SCODE

The main structuring elements of the SCODE method are Zwicky boxes, dimensions and alternatives, they build a hierarchy with three levels. Ontologies on the other hand offer many more possibilities to structure a domain. In order to leverage these we enriched our SCODE files with annotations (aka control codes) to influence the ontology exporter (ZwickyExport).

Most dimensions are mapped to ontology classes (e.g. "Hair length" is a sub class of Pedestrian), but some dimensions are relations (e.g. the dimension "Interaction with person"). They are thus annotated with **{RELATION:Domain:Range}** which triggers an export of this dimension as object property between the given domain and range.

Another important annotation is **{Range:min:max:unit}**. It is used to define physical value ranges for alternatives (e.g. for pedestrian age - teenager: {RANGE:10:17:year}). The exporter adds these ranges to the classes in the ontology and enables to use reasoning: A reasoner assigns a pedestrian instance with data property age 15 years automatically to the type age_teenager.

To enable deeper hierarchies the **{PARENT:class_name}** annotation is used. It will create an additional class class_name as super class of the current dimension. It is used for example to group rain, snow and hail dimensions to the new super class precipitation.

Other annotations are **{MULTIINSTANCE}** for dimensions with alternatives which are not mutual exclusive, **{LABEL:lang:label_name}** to add comments in different languages, and **{CLASS:class_name}** to add additional classes.

More details about control codes are available on this page SCODE RDF (Ontology) Export.

1.3.2.2.3.3 The Generation Process

To generate the KI-A ontology from the 22 Zwicky boxes in phase **C** we introduced a tool based process, see also figure #GenerationProcess. The three individual steps are explained below.

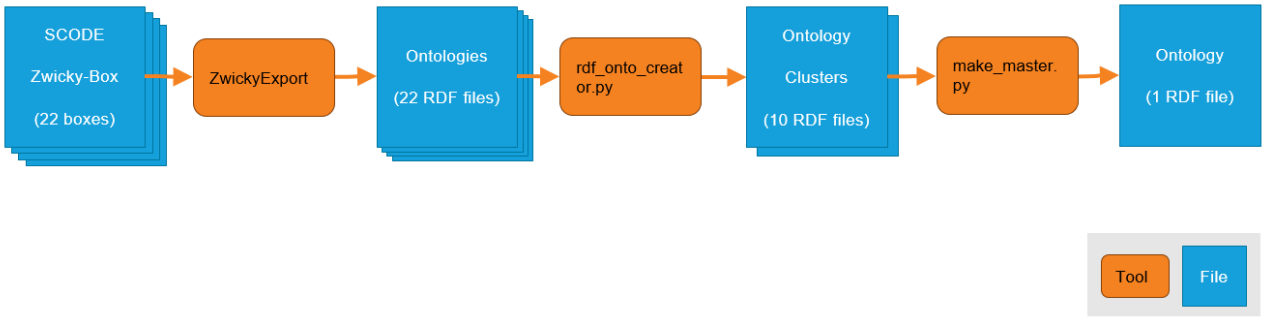


Figure 3: Generation Process - from SCODE Zwicky boxes to overall KI-A Ontology

1.3.2.2.3.3.1 SCODE to RDF (Phase C Step1)

The first step of the generation process is the generation of a RDF file for each of the 22 SCODE Zwicky boxes. Therefore an existing tool from Bosch called **ZwickyExport** has been extended to support RDF files. ZwickyExport is a GUI based tool to export SCODE files in many different formats, in KI-Absicherung it is also used to provide the HTML and YAML export, see also GitLab repo. ZwickyExport writes the RDF header, maps dimensions to classes and alternatives to disjoint sub classes, and adds ranges and labels where needed, All annotations as described above are considered during the export. The tool also offers several options to influence the RDF export, see also image below.

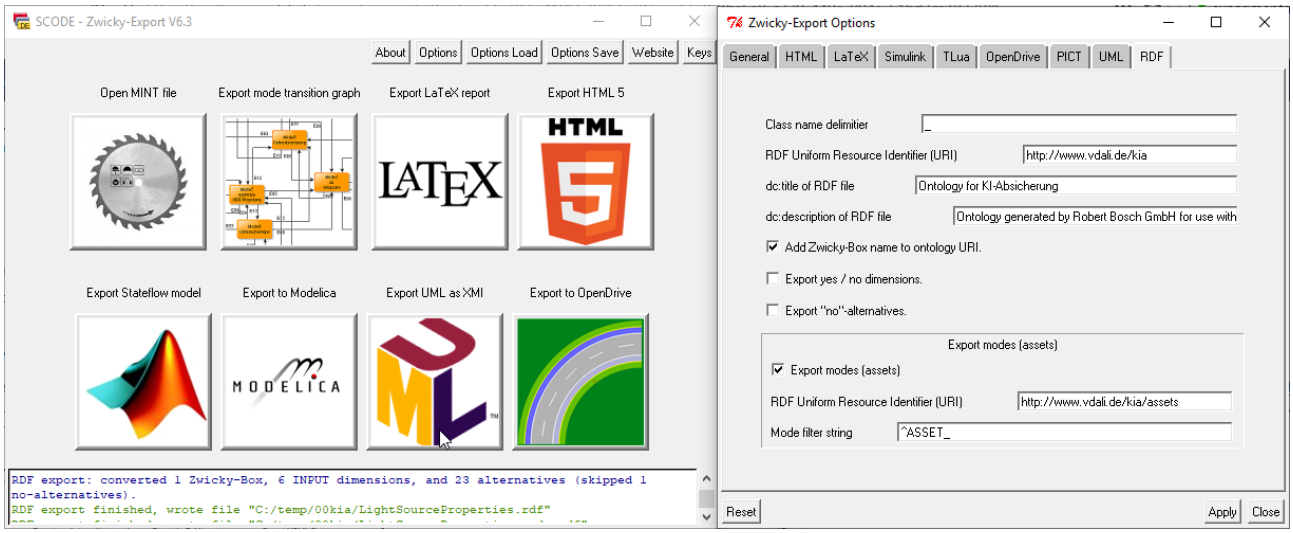


Figure 4: Overview export options in the SCODE Tool

More details about the generation process are available on this page SCODE RDF (Ontology) Export.

1.3.2.2.3.3.2 RDF to Clusters (Phase C Step 2)

Having the 22 SCODE Zwicky boxes exported into individual RDF files as described above, we decided to generate 10 cluster ontologies out of them. This on the one hand allows for a better collaboration across partners since partners can work in parallel on different cluster ontologies. On the other hand, it still ensures that future structural changes within one Zwicky box Ontology that imply changes in other topic-related Zwicky boxes can be handled in one cluster ontology. Thus, most cluster ontologies merge those single Zwicky box RDF files that are closely related topicwise. One exception is the VVM Cluster Ontology that merges the Zwicky box RDF files that will most probably have a high overlap with the VVM Ontology. It represents the main interface

for the exchange and harmonization with the VVM partners with regard to the ontology development. The clustering of the single Zwicky box RDF files into the Cluster Ontologies is depicted in Figure 5 below. As can be seen from the overall Generation Process the clustering is done automatically using a Python script called **rdf_onto_creator.py**. Given a list of single Zwicky box RDF files that belong to one cluster, the script merges the content contained in each of these files into one single cluster RDF file. In a second step, it adds the relations between the Zwicky boxes belonging to a cluster ontology (represented as labelled arrow connections in Figure 5) as so-called Object Properties and introduces some hierarchy where applicable. The resulting Cluster Ontologies are saved to RDF files and uploaded into the GitLab Repository.

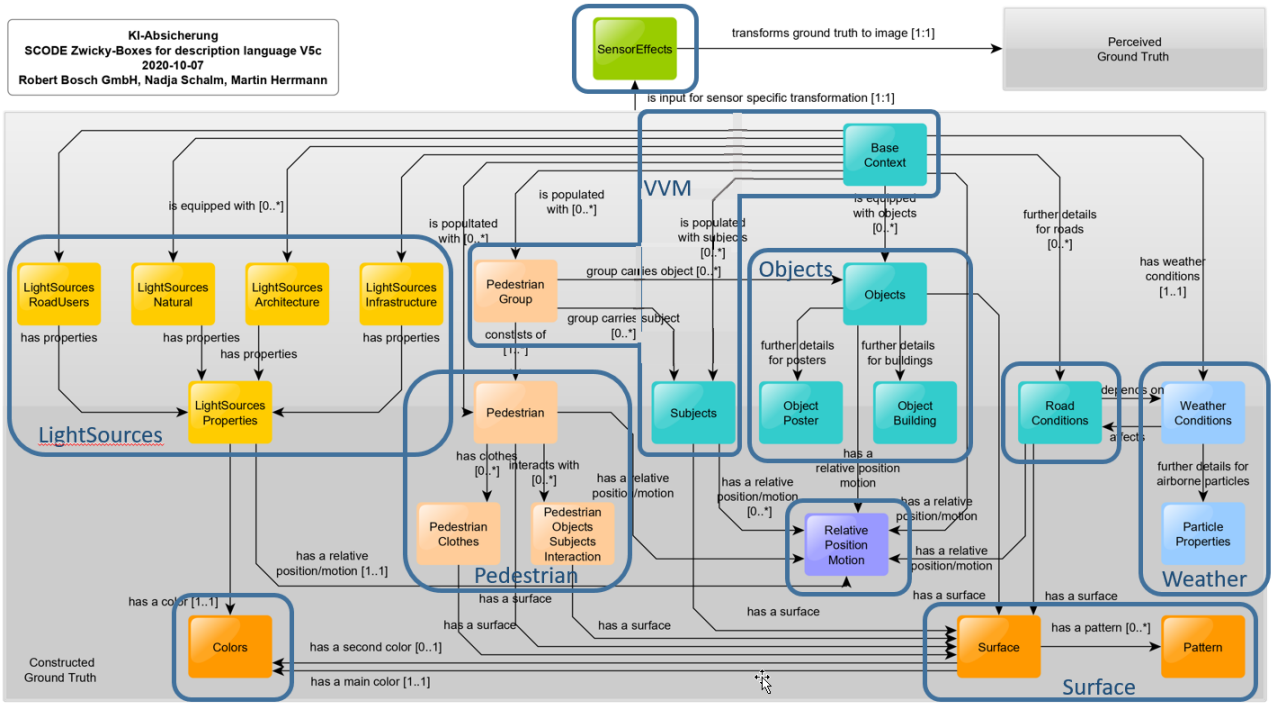


Figure 5: Clustering of Zwicky box RDF files into Cluster Ontologies

1.3.2.2.3.3.3 Clusters to Ontology (Phase C Step 3)

For keeping the ontologies light-weighted and allow concurrent modifications of different clusters, it was decided to keep the cluster ontologies as individual OWL files. Instead of merging them, all ontologies were imported into a so called *master ontology*, which includes all classes and properties and can be used for references. The generation step of the master ontology was realized using another python script that is also included in the GitLab Repository. Besides the generation of the ontology, it also generates a catalog file specific for the usage of Protégé, where all URIs are mapped to file references. In another script, all sub-ontologies are modified to allow a simpler usage in other applications. The modifications include:

- outsourcing all "has_integer_value" and "has_float_value" properties into a common utility ontology and referencing to those instead of redefining those properties in the sub-ontologies
- adding object-super-properties to indicate special properties
    - has_value: if a data property with the same name and the suffix "_value" is present

- has_alternative: if all subclasses of the property's range have to be interpreted as alternatives
- has_predefined_alternatives: *not used yet*
- has_multiple_instances: if "{MULTIINSTANCE}" annotation is present
- has_relation: if "{RELATION:..}" annotation is present
- has_yes_no_alternatives: if "Yes and No" is part of the comment, which indicates it has alternatives in the Zwicky that were not exported

Those steps allow an easier interpretation of the different types of properties and therefore allow an easier implementation of automatic generated user interfaces, e.g. for asset descriptions.

1.3.2.2.4 Manually Extending the Ontology

The resulting ontology contains all the information the was contained in the Zwicky boxes. However, as noted above, Zwicky boxes do not provide means to express explicit relationships between different context elements. Since ontologies provide such means, these relationships should be added manually in phase **D** of the development. Since the development of the actual ontology is subject to the project result E4.1.4a, we refer to this result for a more in-depth description of the ontology.

1.3.2.3 Final Input Domain Model

In this section, we present the contents of the final input domain model that we created in SCODE. The model presented here is available in various formats as part of the project result E4.1.2c and it served as the basis for generating the ontology as described in the previous section. In the following, we provide a short description of the Zwicky boxes and their intent. Thereby, we cluster the individual Zwicky boxes according to the cluster ontology structure hightlighted in Figure 5. The large grey box "constructed ground truth" in Figure 5 corresponds to the scene and scenario description and is subject to the Pegasus layer model. The sensors of the ego vehicle are used to perceive this ground truth, which yields a "perceived ground truth" in KI-A as the sensor effects are considered in the rendering process for obtaining the synthetic images and their ground truth.

It should be noted that the resulting SCODE model contains a significant number of context dimensions and variations that span a huge state space. The state space corresponds to the Cartesian product of all dimensions each represented by their discrete sets of alternatives (example: The "Relative Position" Zwicky box has 6 dimensions with 3, 5, 3, 3, 3, and 4 alternatives. 3x5x3x3x3x4 = 1620 possible combinations.). Table 1 provides an overview of the sizes of the individual Zwicky boxes. The full Cartesian product has more than 10^{120} combinations. For applications to real data where the corresponding meta-annotations are not available at zero cost, it should be tailored using the iterative refinement step of our strategy to those dimensions that could be shown to be safety critical.

Category	# combinations
Base Context	4*107
Color	126

Category	# combinations
Light Source Properties	1500
Light Source Architecture	384
Light Source Infrastructure	4.4*105
Light Source Natural	12096
Light Source Road User	2.8*1017
Object Building	540
Object Poster	48
Objects	6.2*109
Particle Properties	3888
Pattern	5
Pedestrian	1.5*1011
Pedestrian Clothes	2.4*107
Pedestrian Group	11250
Pedestrian Objects Subjects Interaction	3.2*1011
Relative Position	1620
Road Conditions	4.4*105
Sensor Effects	6.2*105
Subjects	2.1*109
Surface	384
Weather Conditions	2.3*1011

Table 1: Statistics of the Single Zwicky boxes.

1.3.2.3.1 BaseContext / VVM

The base context / VVM cluster contains the Zwicky boxes for the Base Context, Subjects, and Pedestrian Groups. We grouped these Zwicky boxes in one cluster as they relate to an overall description of the constituent elements of a traffic situation. A detailed specification in terms of an ontology is pursued in the sister project V&V Methoden such that the description of these aspects has been reduced to what is needed in KI Absicherung without putting a focus on completeness.

The Zwicky box for the base context shown in Figure 6 provides a coarse representation of intersections with their traffic infrastructure (Pegasus Layers 1 + 2). We use a coarse description here as the project will only use a small number of base contexts provided as part of the project result E4.1.1. Due to the restriction of KI Absicherung to pedestrian detection at urban intersections, we restricted the Zwicky box for the base context to the description of 4-way intersections. This Zwicky box has been successfully used to configure and to automatically generate base context proposals for the project result E4.1.1.

Junction Type	4way intersection				
Pedestrian Crossing S	no		yes		
Pedestrian Crossing W	no		yes		
Pedestrian Crossing N	no		yes		
Pedestrian Crossing E	no		yes		
Road Type S	town 2+3	town 1+1	town 2+2	town 2+1	...
Road Type W	town 2+3	town 1+1	town 2+2	town 2+1	...
Road Type N	town 2+3	town 1+1	town 2+2	town 2+1	...
Road Type E	town 2+3	town 1+1	town 2+2	town 2+1	...
Traffic Lights	no		yes		
Traffic Signs	no		yes		
Traffic Mirrow	no		yes		
Turninglane S E	no		yes		
Bus Stop S	none	inbound	outbound	both-way	
Bus Stop W	none	inbound	outbound	both-way	
Bus Stop N	none	inbound	outbound	both-way	
Bus Stop E	none	inbound	outbound	both-way	

Figure 6: Zwicky box for the Base Context

The Zwicky box for the subjects shown in Figure 7 provides a description of different actively moving objects that may appear in an image (Pegasus Layer 4.1). These are mostly other vehicles, but we also considered airborne subjects and robots. Since detecting other vehicles is not in scope of KI Absicherung, we restricted ourselves to the existence of these subjects without going into more details of their visual appearance besides specifying their surface.

Private transport motorized	no	car	motorboat	electric bicycle	electric skateboard	hovercraft	moped	motorcycle	private jet	boat	mobile home	caravan	trike	electric kick-scooter	tuk tuk	golf cart	
Public transport	no	bus	tram	train	ship	plane	suspension railway			cableway		auto rickshaw			taxi		
Private transport non-motorized	no	horse	horse-drawn vehicle		hot air balloon	scooter	skateboard	bicycle	dicycle	snowmobile		cargo bike		trike	slide		
Service vehicles	no	police	ambulance		fire-fighting vehicle		road cleening		waste management			civil protection		automobile services			
Commercial vehicle	no	truck	heavy transport		forklift truck		excavator	road roller		tar machine		food truck		post vehicle		tractor	
Combined with	nothing		trailer		towing car with rope					towing car with pole							
Persons visible	no						yes										
Is carring	nothing	cars	truck	bikes	boat	bus	trailer	motorcycle		heavy cargo		mobile home	caravan	plain	carriage	people	
Airborne subject	no	plane		hot air balloon			flying animal(s)			drone		helicopter		zeppelin			
Animal	no		pet		wild native animal					wild exotic animal							
Robot	no		small		medium					big							

Figure 7: Zwicky box for Subjects

The Zwicky box for pedestrian groups shown in Figure 8 provides a description of a group of pedestrians, i.e., how many pedestrians of which type gather as a group in an image (Pegasus Layer 4). A description of each individual pedestrian of that group is subject to the Pedestrian Zwicky boxes below. A description of pedestrian groups is relevant to characterize, for example, crowded scenes.

Old Persons	0	1	2-5	5-10	>10
Adults	0	1	2-5	5-10	>10
Teenagers	0	1	2-5	5-10	>10
Children	0	1	2-5	5-10	>10
Group Density	low		normal		high
Carrying object	no		yes		
Carrying subject	no		person		animal

Figure 8: Zwicky box for Pedestrian Groups

1.3.2.3.2 Objects

Objects are stationary or passively movable elements of a traffic situation that may be contained in an image of a traffic situation (Pegasus Layer 4.2). To this end, we also modeled vegetation as part of the objects. Please note that standards like OpenDrive and OpenScenario use a different structure where only on-road objects are part of the scenario, where as off-road objects belong to the road layer. The colors and surfaces of the objects are described by the color and surface Zwicky boxes below.

The objects Zwicky box shown in Figure 9 focusses on small passively moveable objects and vegetation that may appear on or along the road. Since the number of different types of objects that may exist in the world is vast and constantly evolving, the list is, of course, incomplete. Focus has been put on selecting exemplary objects that have shown to be interesting in the past like advertising pillars (causing occlusion), hydrants and lanterns (could be mistaken for humans), or phone booths (people inside).

Decoration	no	Christmas	Easter	Halloween	other
Poster	no		yes		
Vegetation bush density	no	low	medium	high	
Vegetation tree density	no	low	medium	high	
Vegetation flower density	no	low	medium	high	
Vegetation grass density	no	low	medium	high	
Mountains height	no	low	medium	high	
Sky visible in FOV	no		yes		
Building density	no	low	medium	high	
Advertising Pillar	no		yes		
Hydrant	no		yes		
Trash can	no		yes		
Container	no		yes		
Furniture	no		yes		
Bicycle stands	no		yes		
Postbox	no		yes		
Phone booth	no		yes		
Lantern	no		yes		
Column	no		yes		
Sculpture	no	humanoid	other		
Airborne object	no	balloon	plastic bag	paper	light material
Land border	no	wood fence	metal fence	concrete wall	vegetation (hedge)

Figure 9: Zwicky box for Objects

In addition to the moveable objects, we created a dedicated Zwicky box for buildings shown in Figure 10 as one of the main classes of stationary objects. To this end, we focussed on the materials and rough shape for characterizing the buildings in order to provide a coarse description of the scene backgrounds and to account for material effects (e.g., for glas buildings that may cause reflections).

Building material	glas	concrete	stone	wood	plaster				
Building height	low	medium	high						
Building shape	quboid	cylinder	pyramid	complex					
Roof type	no	flat	mono pitched	gabled	hipped	butterfly	arched	domed	other

Figure 10: Zwicky box for Buildings

Poster have been discussed in the AP4.1 as being an interesting kind of object as posters often depict images of the real world such that a computer vision function may consider the objects on a poster as being part of the real world. We created only a small characterization of posters as shown in Figure 11 as part of the result documentation, however, depending on the content of the poster, the contents of the domain model may be recursively applied. As an example, if the poster shows a person and if it is important how this person looks like, the pedestrian description below may be also used to describe the contents of the poster.

Poster size	S	M	L	XL		
Poster main subject	person	nature	vehicle	object	text	road scene
Poster main color	single-color	colorful				

Figure 11: Zwicky box for Posters

1.3.2.3.3 Pedestrian

Since pedestrian detection is the main use case of KI Absicherung, the pedestrian description is the most detailed among all Zwicky boxes. In particular, we provide descriptions of the pedestrians themselves, their clothes, and the objects they carry with them (Pegasus Layer 4.1).

The pedestrian Zwicky box shown in Figure 12 enables to provide a detailed description of the visual appearance of a single pedestrian. It was one of the main sources for describing pedestrian assets and for deciding on additional pedestrian assets to purchase. If multiple pedestrians are part of a scene, the description needs to be applied to each of them.

Age	child	teenager	adult	old person										
Gender	male	female	other											
Body shape	thin	normal	muscular	obese										
Body type	normal	apple	pear	hourglass	chilli	triangle								
Body height	<80cm	80cm-120cm	120cm-160cm	160cm-200cm	>200cm									
Pigmentation	high	medium	low											
Skin modification	no	freckles	dirt	scar	burn injuries	abrasion	body painting/tattoo							
Hair length	no	short	medium	long										
Hair color	irrelevant	white	blond	red	grey	brown	black	other						
Hair style	irrelevant	pony tail	bob	afro	mohawk	other								
Beard size	no	small	medium	big										
Face shape	round	square	diamond	oval	heart	rectangular								
Special handicap	no	elephantiasis	missing leg	missing arm	splint on leg	splint on arm	bandage	neck brace						
Pose	standing	walking	running	sitting	squatting	lying	jumping	falling	reeling	ducking	crawling	bending down	crouching	other entertainment	untypical
Body position to sensor	front	side	back											
Face position to sensor	front	side	back											

Figure 12: Zwicky box for Pedestrians

Apart from their general description, the clothing of pedestrians significantly changes their visual appearance and deserves special attention. The pedestrian clothing Zwicky box shown in Figure 13 enables to characterize the different kinds of clothing that a pedestrian may wear. The materials and colors of the clothes are specified using the surface and color Zwicky boxes below. In order to reduce the complexity of the description, we decided to limit the description

to an upper part and lower part clothing type, whereas upper part refers to clothing above the hip and lower part to clothing below the hip.

Headgear type	no	cap/baseball cap	top hat	cowboy hat	headscarf/veil	helmet	mask	hood	turban	wig	orthodontic headgear	...
Shoes type	no	sport shoes	high heels	sandals	skates	ice skates	skis	boots	ankle boots			
Outerwear type	no	coat	jacket	vest								
Clothing upper part type	no	T-shirt	sweater	shirt	blouse	top	cardigan	burka	dress			
Clothing lower part type	no	trousers	skirt	shorts								
Clothing lower part length	irrelevant	short	medium	long								
Clothing legs type	no	leggins	tights	socks	stockings							
Gloves	no	finger gloves	fingerless gloves	mittens								
Scarf	no	small	medium	big								
Clothing type	casual	chic	work	untypical	sporty							

Figure 13: Zwicky box for Pedestrian Clothes

Pedestrians frequenly interact with other subjects and objects, which may alter their visual appearance and may affect the ability of the pedestrian detection to correctly identify them. Therefore, we dedicated a separate Zwicky box shown in Figure 14 to these kinds of interactions. For several of these interactions, the recognizing the semantics might be of interest, for example, whether a pedestrian is pushing a bicycle or riding on it (specified as two wheeler in the Zwicky box).

Mobile phone	no interaction	in hand on ear	holding with shoulder	in hand not on ear					
Umbrella	no	open	closed						
Head phone	no interaction	in ear	on ear						
Glasses	no	clear	sun						
Jewellery	no	decent	moderate	opulent					
Other carried object	0	1	2	3	many				
Levitating object	no	medium	large						
Buggy	no	normal	twin serial	twin parallel					
Transport cart	no interaction	pulling	pushing						
Two wheeler	no interaction	pushing/holding	leaning on	sitting on					
Wheelchair	no interaction	pushing	sitting						
Rollator	no interaction	pushing	sitting on						
Walking aid	no	crutch	white cane	walking cane	nordic walking poles	stilts			
Pet/Animal	no	leashed	unleashed	guide dog	carried				
Interaction with person	no interaction	holding hands	piggyback	carring on shoulder	hugging	carring baby sling	walking arm-in-arm	carring in arms	fighting
Interaction with vehicle	no interaction	getting on	getting off	opening/closing trunk					
Special Interaction with object	no interaction	throw	kick						
Person occlusion	0-5%	5-25%	25-50%	50-75%	75-95%	95-100%			
Person occluded by	nothing	vegetation	other subject	architecture/ base context	other object	illumination effect			

Figure 14: Zwicky box for Pedestrian-Subject/-Object interaction

1.3.2.3.4 Relative Position and Motion

The relative position and motion Zwicky box shown in Figure 15 enables to describe where an object or subject (e.g., a pedestrian) is located with respect to the camera. This can be used, for example, to ensure that important subjects like pedestrians appear anywhere in the pictures to avoid bias causing pedestrians detection performance to depend on their location in the image. In addition, we provide a coarse description on how the subjects move in relation to the camera, which is particularly interesting for time-coherent sequences of images.

Longitudinal distance to sensor	far away		middle		very close
Lateral position to sensor	far away on left side	in the near on left side	center	in the near on right side	far away on right side
z position to sensor	below	same level			above
Longitudinal motion	no	in same direction		in opposite direction	
Lateral motion	no	to right		to left	
Absolute motion speed	zero	low	medium		high

Figure 15: Zwicky box for Relative Position and Motion

1.3.2.3.5 Weather

The weather (Pegasus layer 5.1) may affect how the entire scene looks like and it may also signficantly impact detection performance as subjects and objects may be less good to see. Depending on the goal of the description, we provide two levels of details.

The weather conditions Zwicky box shown in Figure 16 provides a weather description based on human understandable terms including a grounding in physically measurable quantities. It relates to the different kinds of precipitation and their intensity (e.g., rain, fog, ...) as well as other weather phenomena (e.g., dust and sand storms) and seasonal ground covering (e.g., leaves or snow).

Ground temperature	low (<3°C)				above low		
Ground cover main type	no	snow	ice	hail	leaves	ash	sand
Ground cover height	no	low	medium		high	very high	
Ground wetness	dry	slightly moist	wet with puddles		low flooded	high flooded	
Rain	no	light	medium		strong	heavy	
Snow	no	light	medium		strong	heavy	
Hail	no	light	medium		strong		
Fog	no	light	medium		strong		
Smog	no	light	medium		strong		
Smoke	no	light	medium		strong		
Dust	no	light	medium		strong		
Sand	no	light	medium		strong		
Debris	no	some			many		
Leaves	no	some			many		
Wind type	no	low	medium		high	hurricane	
Wind direction	South	South-West	West	North-West	North	North-East	East	South-East	diverse
Whirls	no				small		
Mirage	no				yes		
Earthquake	no	light	medium		strong		

Figure 16: Zwicky box for weather conditions

The different kinds of weather phenoma listed in Figure 16 have in common that they are caused by particles of a specific material that are in the air. For synthetic data generation, it might be of interest to also describe these particles in more details, e.g., to generate different kinds of snow fall or rain. If this is required, the particle properties Zwicky box in Figure 17 may be utilized. It provides a detailed characterization of particles and how they influence the colors of the overall scene.

Particle size	small	medium	big	huge
Particle spectral characteristic blue shift	neutral	low	high	
Particle spectral characteristic red shift	neutral	low	high	
Particle spectral characteristic green shift	neutral	low	high	
Particle fall velocity	low	medium	high	very high
Particle dispertion factor	small	medium	big	
Particle attenuation factor	small	medium	big	

Figure 17: Zwicky box for particle properties

1.3.2.3.6 Road Conditions

The road conditions shown in Figure 18 describe the current condition of the road surface (Pegasus layer 1). The road surface is influenced on the one hand by the material and wear of the road. On the other hand, the road conditions are significantly influenced by the weather conditions, but they may also be independent to some degree (e.g., it is possible to have a wet road with puddles in bright sunshine). Furthermore, we accounted for some special lost cargo objects that may lie on the road. These, however, could also be part of the object description above.

Road surface	asphalt	concrete	cobble stone									
Road quality	tear	perfect										
Road wetness	dry	slightly moist	wet with puddles	low flooded	high flooded							
Road cover main type	no	snow	ice	hail	leaves	ash	sand					
Road cover height	no	low	medium	high	very high							
Additional road cover type	no	snow	ice	hail	leaves	ash	sand					
Object on road	no	lost cargo	pile	traffic cone	stone	tyre (part)	tension belt	timber	fire	debris	pothole	barrier
Object height	no	low	medium	high	very high							

Figure 18: Zwicky box for describing the road surface

1.3.2.3.7 Light Sources

Besides the weather, the illumination of a scene by different kinds of light sources (Pegasus layer 5.2) is second main influence that may change the entire image. Since light sources and their properties may be easily varied in a rendering engine for synthetic data generation, we developed a rather detailed description of different kinds of light sources, distinguished by where they occur. We first describe the light sources as such before digging deeper into their properties.

Figure 18 shows the Zwicky box for natural light sources. These are light source appearing in nature such as the sun, lightnings, or fire.

Sun elevation	night	astronomical twilight/dawn	nautical twilight/dawn	civil twilight/sunrise	low	medium	day
Sun and sensor interaction	not in FOV, no total reflection			in FOV	not in FOV, total reflection		
Sky	clear	high clouds completely covered		low partly clouded	low completely covered		
Moon phase	not visible		new		half	full	
Thunderbolts	no	low activity		high activity			
Northern lights	no			yes			
Fire	no	low activity		high activity			
Meteorite	no			yes			

Figure 19: Zwicky box for natural light sources

A second important kind of light source are light sources attached to other road users (e.g., lights of other cars) shown in Figure 19. These are of particular interest as they are often facing the camera directly, for example the headlights of oncoming traffic or the rear lights of the lead vehicle. We distinguish different light technologies such as halogen, LED, and Xenon lights as they different significantly in their properties.

Headlight high beam HD Matrix	no	direct	reflected	direct & reflected
Headlight high beam LED	no	direct	reflected	direct & reflected
Headlight low beam LED	no	direct	reflected	direct & reflected
Headlight high beam Halogen	no	direct	reflected	direct & reflected
Headlight low beam Halogen	no	direct	reflected	direct & reflected
Headlight high beam Xenon	no	direct	reflected	direct & reflected
Headlight low beam Xenon	no	direct	reflected	direct & reflected
Foglight LED	no	direct	reflected	direct & reflected
Foglight Halogen	no	direct	reflected	direct & reflected
Taillight LED	no	direct	reflected	direct & reflected
Taillight Halogen	no	direct	reflected	direct & reflected
Brakelight LED	no	direct	reflected	direct & reflected
Brakelight Halogen	no	direct	reflected	direct & reflected
Indicator LED	no	direct	reflected	direct & reflected
Indicator Halogen	no	direct	reflected	direct & reflected
LiDAR	no	direct	reflected	direct & reflected
Emergency light blue LED	no	direct	reflected	direct & reflected
Emergency light blue rotating	no	direct	reflected	direct & reflected
Emergency light red LED	no	direct	reflected	direct & reflected
Emergency light red rotating	no	direct	reflected	direct & reflected
Signal light orange LED	no	direct	reflected	direct & reflected
Signal light orange rotating	no	direct	reflected	direct & reflected
Vehicle position light LED	no		yes	
Vehicle position light Halogen	no		yes	
Light projections	no		yes	
Bicycle front light LED	no	direct	reflected	direct & reflected
Bicycle front light Halogen	no		yes	
Bicycle back light LED	no		yes	
Bicycle back light Halogen	no		yes	
Torch white	no	direct	reflected	direct & reflected
Carried position lights LED	no		yes	
Carried laser	no	direct	reflected	direct & reflected
Camera Flash Light	no		yes	

Figure 20: Zwicky box for light sources of other road users

The third kind of light sources are light sources of the traffic infrastructure shown in Figure 20 that the camera passes while the automated vehicle moves along the road. In particular, these are different kinds of traffic lights, street lights and variable traffic lights. In addition, they may again use different light technologies.

Traffic Light Low Power LED	no	green	yellow	red	red yellow	yellow flashing
Traffic Light Standart LED	no	green	yellow	red	red yellow	yellow flashing
Traffic Light Halogen	no	green	yellow	red	red yellow	yellow flashing
Street Light LED	no				yes	
Street Light Halogen	no				yes	
Street Light High Pressure Metal Vapor	no				yes	
Street Light Natrium Vapor	no				yes	
Street Light Fluorescent	no				yes	
Variable Traffic Sign LED	no				yes	
Variable Traffic Sign Halogen	no				yes	
Self Luminous Traffic Sign	no				yes	
Beacon Light	no				yes	
Construction Zone Surveillance Green Light	no				yes	
Traffic Enforcement Camera Flash Light	no				yes	

Figure 21: Zwicky box for light sources of road infrastructure

The fourth and final category of light sources that we distinguish are light sources attached to the surrounding architecture such as buildings shown in Figure 21. To this end, there is again a huge amount of different kinds of lights that may be attached to buildings. Thus, we aimed for a coarser description and propose to aim for an iterative refinement during the development process.

Building Light	no	inside	outside	inside+outside
Advertising Board LED Video Wall	no		yes	
Advertising Board illuminated	no		yes	
Facade illumination	no		yes	
Laser	no	direct	indirect	
Season Decoration Lights	no		yes	
Flood Light	no		yes	

Figure 22: Zwicky box for light sources of attached to buildings and the surrounding architecture

Finally, we provide a description of the properties of a single light source in Figure 22. Even though most light sources appear to be constant to a human observers, particularly LED light sources are pulsed as one example. In addition, light sources may have different angles in which they emit light ranging from very narrow spot lights to omni directional lights such as a classic light bulb without a lampshade.

Light Source Type	constant		sinus		pulsed
Modulation frequency	no impact	below sensor	as sensor in phase	as sensor out of phase	above sensor
Pulse width ratio	no	very low	low	medium	high
Extention	point/spot		small	medium	high
Beam angle	focused	small	medium	high	omni directional
Effective Intensity	tbd.				

Figure 23: Zwicky box for properties of light sources

1.3.2.3.8 Surfaces

Surfaces and their patterns (Pegasus Layer 7.1) can be characterized by their ability to reflect and absorb light. These properties are of particular interest when designing materials to be used as part of a rendering engine for being able to synthesize realistic images. For describing the contents of a resulting image, these detailed properties are less important. Therefore, we aimed for a more concise specification.

The surface Zwicky box shown in Figure 23 enables to characterize the surface of an object including the clothing of a pedestrian. It needs to be applied to any surface of any subject and object in the scene.

Reflection directionality	diffuse		silky gloss	mirroring
Retroreflectivity	no		yes	
Structure	no	low	medium	high
Translucence	no	low	medium	high
Wetness	no	low	medium	high

Figure 24: Zwicky box for describing surfaces

The pattern Zwicky box shown in Figure 24 enables to describe patterns on a surface. This is most relevant for clothing where patterns like stripes are quite common. The list of patterns that we developed is yet incomplete and only exemplary as the safety relevance of patterns on clothing was yet unclear.

Pattern	no	stripes	plaid	patchy	imprint

Figure 25: Zwicky box for describing patterns on a surface

1.3.2.3.9 Colors

The color of an object is a relative properties that significantly depends on the lighting conditions. In addition, several color models such as RGB (red-green-blue) or HSV (hue-saturation-value) exist for describing colors in digital images. In the Zwicky box shown in Figure 25, we chose to resort to the RGB color model and preliminary clustered the different ranges in 5 clusters to enable creating a finite number of colors.

Spectral range 1 response (blue)	no	low	medium	high	very high
Spectral range 2 response (green)	no	low	medium	high	very high
Spectral range 3 response (red)	no	low	medium	high	very high

Figure 26: Zwicky box for describing colors

1.3.2.3.10 Sensor Effects

Two images of the very same scene may look significantly different if they were recorded by two different sensors as cameras may employ imagers and lenses with different properties. In the

Zwicky box in Figure 27, we collected some of the most important sensor effects (Pegasus Layer 8) for video cameras.

Straylight from protection layer or lens	low	medium	high	
Saturation effect in detector	low	medium	high	
Color contrast resolution of detector	low	medium	high	
Luminance contrast resolution of detector	low	medium	high	
Temporal resolution of detector	low	medium	high	
Transfer curve in detector	linear	piecewise linear	logarithmic	
Color calculation in ISP	ok	creates color artefacts		
Signal loss in protection layer, lens or detector	low	medium	high	
Geometric distortion in protection layer, lens or detector	low	medium	high	very high
Noise performance caused by sensor or ISP	low	medium	high	
Sensor position	low position	standard	high position	very high position
Geometric resolution of detector	Full HD	HD	VGA	

Figure 27: Zwicky box for sensor effects

1.3.2.4 Usage of the Domain Model

The domain model has been used for various purposes in KI Absicherung. These include the definition of an operational design domain, computation of an input coverage, and the usage for structured data generation.

1.3.2.4.1 Definition of an ODD

An initial ODD for the project has been defined as part of the P1 process. The goal was to have a common understandig on what context elements and variations should be supported by the data production for tranche#3 and resulted in an operational design domain for the data production for tranche #3.

The work was then extended by the process P3 for defining an ODD for the entire project considering different views on this topic, namely from data production, function definition and safety assurance. The ontology offered a common language for all different perspectives, although its main usage was from the perspective of the data production. Consequently, the operational design domain for the overall function developed by P3 was based on the ODD description created by P1. For the definition of the ODD, Zwicky boxes have the advantage that they allow a table like representation and an overview of what is part of the ODD and what is not (e.g. time → day is part, while time → night is not part of the ODD). The representation as a ontology however allows more complex structures and sub structures and introducing sub classes for more detailed representations when needed or pool multiple classes into a new reference (e.g. all pedestrians and cyclists are VRUs).

The review process of the ODD and the link the ODD helped to test the ontology for completeness, but also to have further improvements of the level of detail. Most class of the ontology were either put on a positive or negative list depending on if the class is part of the ODD (see ODD Definition: Draft Version - 0.1). A very good help was the definition of the classes using physical properties.

1.3.2.4.2 Definition of Meta-Annotations for Datasets

The domain model and the ontology derived thereof provide the basis for the meta-annotations defined as part of project result E4.1.4b. The goal of these meta-annotations is to provide more detailed information on the contents of a single image. As shown by the data producers in AP2.5, a significant amount of such meta-annotations can be computed directly during rendering. In addition, as part of AP4.1, we have created an asset ontology for describing the pedestrian assets that were used in the rendering process and using the ontology generated based on the domain model. To this end, the project results clearly show that a detailed labeling according to the domain model is possible, at least for synthetic data.

1.3.2.4.3 Data Coverage

In AP4.4, Bosch proposed a test method based on combinatorial testing take leverages a domain model in the form of Zwicky boxes for computing an input coverage or data coverage measure, i.e., how well does a data set cover the elements of a domain model. This test method has also been investigated as part of the evidence workstream 08 - data coverage and data distribution based on a small subset of the input domain model.

1.3.2.4.4 Structured Data Generation based on NCAP-like Scenarios

In collaboration with AP2.5, we further investigated how an excerpt of the domain model can be used for structured data generation for a specifc test scenario. Here, an NCAP-like scenario was chosen where a pedestrian enters the road while being partially occluded by parked vehicles. The domain model has been used to express the context elements to be varied and the abovementioned combinatorial testing method has been used to derive data specification that could be processed automatically by the data production in AP2.5 for the tranche#6 data. After initial works by Bosch, this technique was further actively used in the evidence work stream 09 - NCAP scenario under the lead of ZF.

1.3.3 Conclusions

The development of the domain model gave an initial validation of the strategy for analyzing the input domain that we developed in KI Absicherung showing that it indeed works. We have demonstrated that a domain model can indeed be built and we outlined how the domain model has already been used within KI Absicherung. In addition, we presented the contents of the resulting domain model and how it has been structured into a number of Zwicky boxes that, in turn, have been associated to a number of different clusters. We believe that the applications of and analyses based on the domain model show the value and benefits of creating such a domain model.

1.4 E4.1.2c Final: nur projektintern für KI Absicherung verfügbar

1.5 E4.1.3 Final: nur projektintern für KI Absicherung verfügbar

1.6 E4.1.4a Final: Beschreibungssprache und Ontologie der Dimensionen (zur Veröffentlichung)

1.6.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Bericht*

Criteria	Classification according to VHB
Group/Cluster	*C*
Type of content	Spezifikation
Classification level	*PU*

1.6.2 Description of the result

In E4.1.4a an ontology is created for the domain of AI based pedestrian detection. This ontology which specifies categories, their properties, and the logical relations between the elements of the categories should ideally be compatible with or even integrable into the ontology developed in the project "V&V Methoden" (VVM). The latter focuses on the characterization of the urban environment in order to assess criticality and define safety strategies. In E4.1.4b, the ontology of E4.1.4a will finally be transferred into machine-readable formats serving different goals.

1.6.2.1 Approach

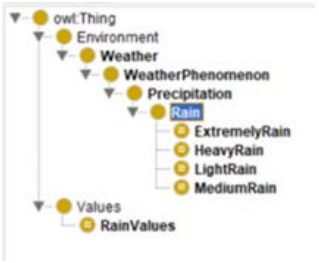
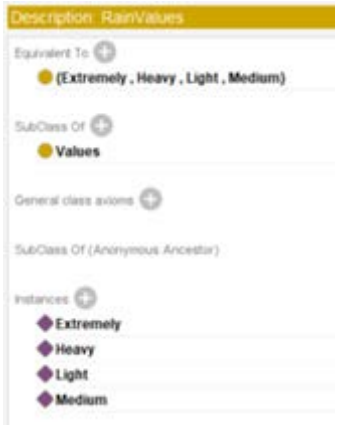
The first step in the creation of an ontology is the definition of its scope and capabilities. It was concluded that the ontology should be able to fulfill the following tasks:

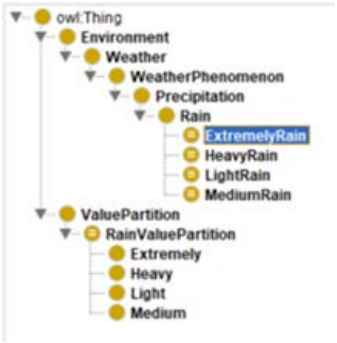
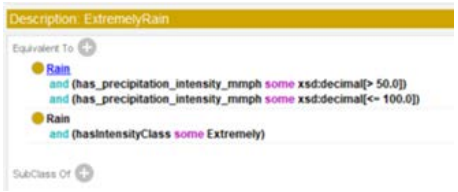
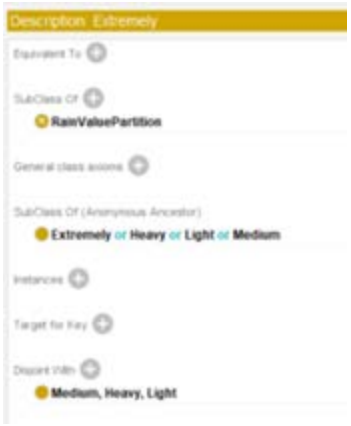
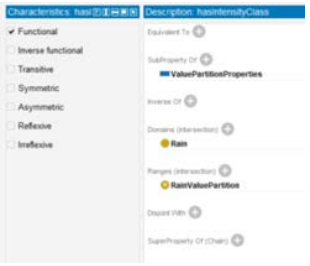
- Define a scene (i.e., a snapshot) on basis of ontology classes, subclasses, or instances
- Check if the defined scene is consistent according to the definition and rules of the ontology
- Incorporate all characteristics and alternatives necessary to describe the urban pedestrian scenario sufficiently
- Allow classification of entities according to given properties and value ranges

To define the characteristics and dimensions necessary to describe an urban pedestrian scenario sufficiently a morphological analysis utilizing SCODE was performed by experts, resulting in a set of Zwicky boxes (E4.1.2c). These are parameters (dimensions) with a corresponding set discrete value. These dimensions should be as independent of each other as possible.

The Zwicky boxes should be translated into an ontology. As a format for the ontology “Web Ontology Language” (OWL) was chosen, since it is a predominant format. Through the implementation of a script, the SCODE Model of the Zwicky boxes can be automatically converted into an OWL ontology. For this translation, three modelling approaches in the ontology are possible:

- Using subclasses only
- Using and enumerated class (i.e., instances)
- Using a value partitio

	Class hierarchy	(Sub)Class ExtremelyRain	Other classes	Data properties	Object properties
Subclasses			None		None
Instances			(Sub)Class "RainValues" with instances 	Analogous	

	Class hierarchy	(Sub)Class ExtremelyRain	Other classes	Data properties	Object properties
Value Partition			"Extremely" as (sub)class of "RainValuePartition" 	Analogous	

Since using subclasses only is the simplest construction and enables the fast conversion of Zwicky boxes, it is used as the modelling approach. Furthermore, this approach can be easily extended by the Value Partition approach if it proves not to be sufficient in the future. Zwicky box alternatives are mapped either as instances without subclasses or as properties. The general mapping rules are:

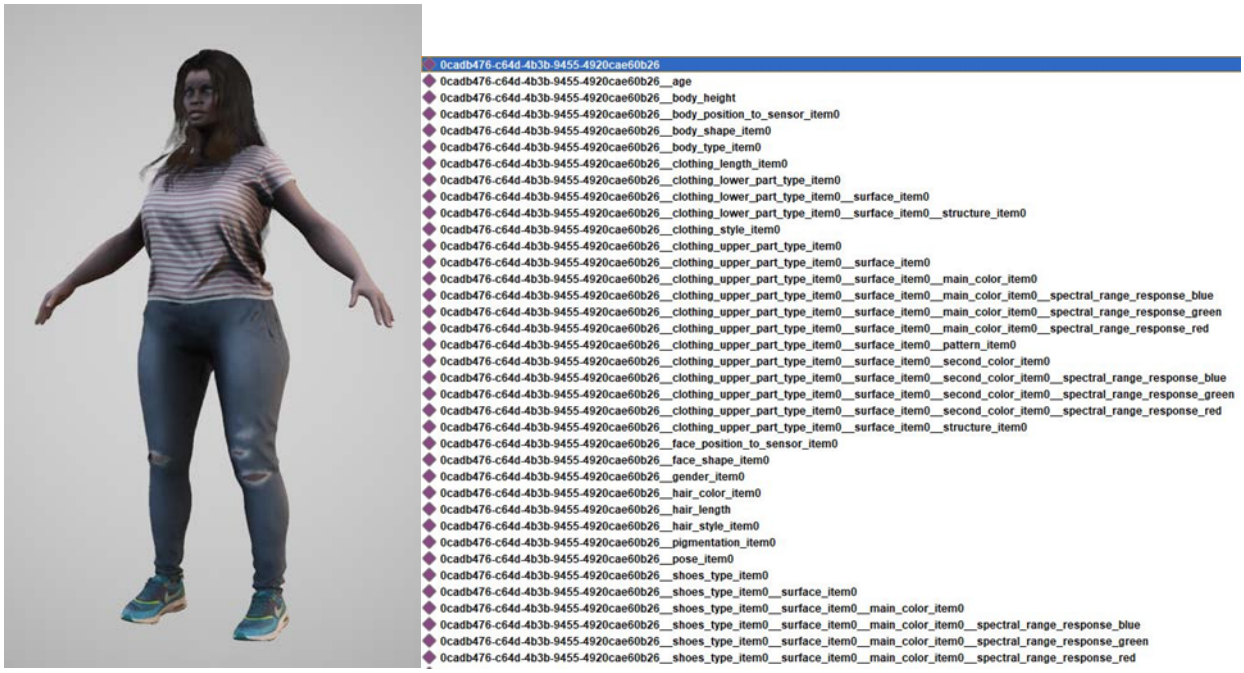
- Zwicky box names are mapped to classes
- Dimensions are mapped to classes
- Alternatives with ranges are mapped to subclasses
    - Ranges are represented using data properties
- Alternatives without ranges are mapped to instances
- Selected alternatives are mapped to properties

The resulting ontology classes were clustered into 10 groups to ensure better cooperation between the project partners. The resulting clusters are weather, light sources, sensor effects, relative position and motion, pedestrian, objects, road condition, color, surface and Verification and Validation Methods (VVM) Project. The resulting ontology consists of a master ontology as the root, which import the different cluster-ontologies. Further enhancements to the ontology were performed directly on the ontology via the tool Protégé. Common data properties and server super-object properties for structuring the object properties of the different cluster ontologies were relocated into a separate utils-ontology that is referenced by each of the cluster ontologies.

Through iterative review and re-structuring of the different cluster-ontologies in working groups, the readability and usability of the ontology was enhanced. More precisely, new super-classes to introduced further hierarchy were introduced, and complex classes were simplified by introducing new object properties and additional classes with subclasses. Part of the enhancement process was a cooperation with V&V Methods, in which a subontology was created that used the core ontology of VVM as a sorting of the KIA ontology on a higher level.

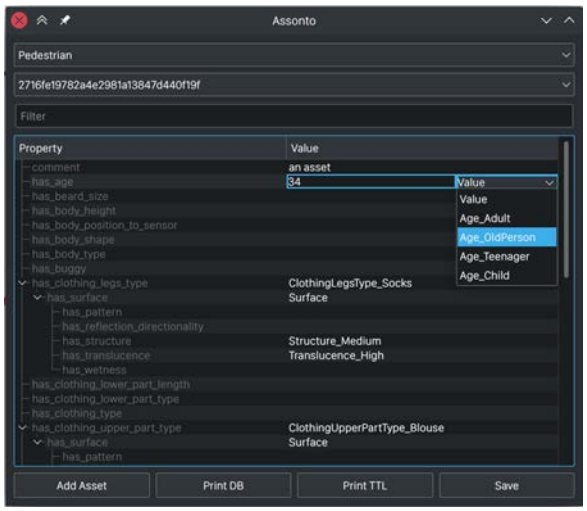
1.6.2.2 Asset Ontology and Assonto

In addition to the master ontology, an asset ontology was created that covers the description of assets available in the Master Asset Catalog for simulation assets based on the different classes, subclasses, and properties available in the master ontology.



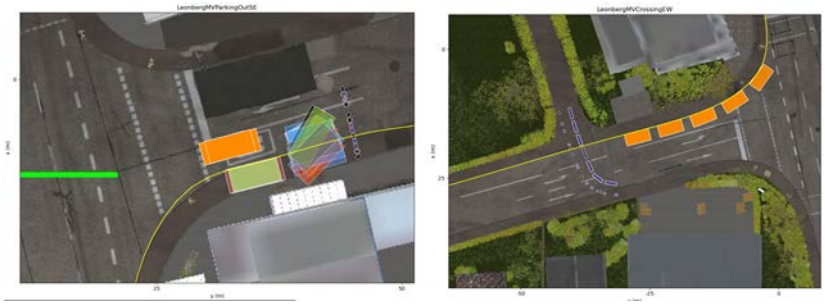

```
"0cadb476-c64d-4b3b-9455-4920cae60b26": {
    "pedestrian.Age": {
        "class": "Adult",
        "val": null
    },
    "pedestrian.BodyHeight": {
        "class": "160cm-200cm",
        "val": null
    },
    "pedestrian.BodyPositionToSensor": {
        "class": "Side"
    },
    "pedestrian.BodyShape": {
        "class": "Obese"
    },
    "pedestrian.BodyType": {
        "class": "Pear"
    },
    "pedestrian.ClothingLength": {
        "class": "Long"
    },
    "pedestrian.ClothingLowerPartType": {
        "class": "Trousers"
    },
    "pedestrian.ClothingLowerPartType__Surface__Structure": {
        "class": "Medium"
    },
    "pedestrian.ClothingType": {
        "class": "Untypical"
    },
    "pedestrian.ClothingUpperPartType": {
        "class": "T-Shirt"
    },
    "pedestrian.ClothingUpperPartType__Surface__MainColor__SpectralRangeResponseBlue": {
        "class": "SpectralRangeResponseBlue",
        "val": 100.0
    },
    "pedestrian.ClothingUpperPartType__Surface__MainColor__SpectralRangeResponseGreen": {
        "class": "SpectralRangeResponseGreen",
        "val": 100.0
    },
    "pedestrian.ClothingUpperPartType__Surface__MainColor__SpectralRangeResponseRed": {
        "class": "SpectralRangeResponseRed",
        "val": 100.0
    },
    "pedestrian.ClothingUpperPartType__Surface__Pattern": {
        "class": "Stripes"
    },
    "pedestrian.ClothingUpperPartType__Surface__SecondColor__SpectralRangeResponseBlue": {
        "class": "SpectralRangeResponseBlue",
        "val": 70.6
    },
```

To fill the asset ontology, a tool (Assonto Tool by Valeo) was developed and utilized. Assonto is a small python tool that provides a Qt based GUI for an easy generation of the asset ontology. It uses "asset templates" which are just a list of root ontology classes. The program reads all other information from the master ontology, including properties, valid alternatives for enumeration properties and sub properties of enumeration properties. When changing the property, the program creates automatically all instances that are necessary to describe the property. For traceability reasons, the information on the tranche, the asset first appeared in and its belonging to the ODD is included in the asset ontology.

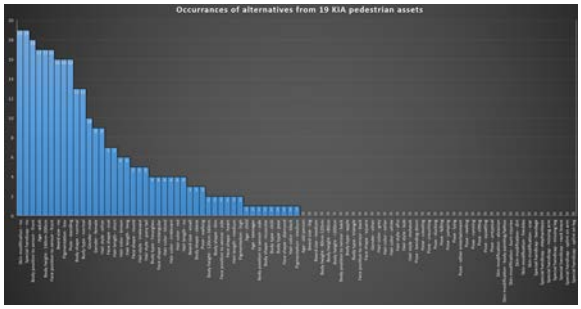


1.6.2.3 Evaluation

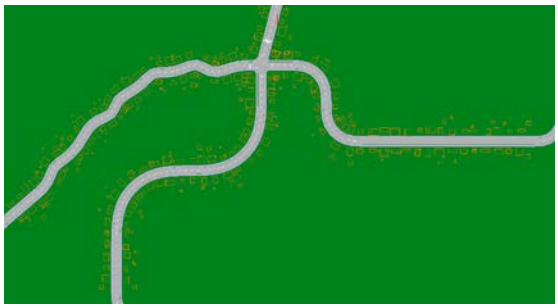
To verify the usability of the KIA ontology to request images we participated in GitLab issue #498 "Create NCAP like scenarios with many systematic variations" (after migration Jira KIA-498). The scenarios are characterized through Zwicky boxes with 13 dimensions to which combinatorial testing is applied. A tool tool to visualize and export all of these generated test cases to the JSON format was developed. The JSON files are used by the data producers to generate the NCAP scenario images. This work helped to streamline and complement the mapping from ontology to data request format. See Systematic image generation for NCAP scenarios for more details.



Another application of the ontology for the selection of additional pedestrian assets was develped. Asset description and applied different analysis techniques were used to come up with suggestions for the purchase of additional pedestrian assets. See also Combinatorial Testing for Asset Definition.



Based on Zwicky boxes for junctions, roads, and lanes, two new base contexts in OpenDrive format for BIT-TS were created. The focus was on many (random) curves and slopes with up to 8% elevation gradient to get a high variety of scenes and pedestrian distribution. All 11594 images of tranche #5 were generated by BIT-TS based on these OpenDrive maps (see also preview frames).



ZF has reviewed the existing elements from the Zwicky boxes for the integration of a action description language, so that the pedestrian interactions can be also respected during the further work. This shall support the identification of safety critical situations on a higher level and the analysis of real data based on this concept is planned.

Other use case for the KIA ontology were the mapping to corner case with AP2.2, the definition of the ODD with P3 and the cooperation with AP1.2 on performance limiting factors.

1.6.3 Conclusion + references

E4.1.4b was able to generate an ontology out of the 22 SCODE Zwicky Boxes, developed in the scope of E4.1.2c. It consists of a master ontology, that import several topical ontologies. The ontology was further extended and adjusted by both restructuring and adding additional hierarchy. Next to the master ontology, an asset ontology was created that covers descriptions of all assets available in the Master Asset Catalog provided by AP2.5.

The applicability of the ontology was shown through its utilization in the systematic image generate (NCAP scenarios), the combinatorial testing for asset definitions, and the generation of new ground contexts.

A publication describing the in KIA developed Ontology and setting it in context to its application in the engineering of automotive AI was created (Ontology Publication). The publication was presented on the Date 2022 Conference. The **final paper** is available via IEEE.

1.7 E4.1.4b Final: Maschinenlesbare Formatdefinition (zur Veröffentlichung)

1.7.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Bericht*
Group/Cluster	*C*
Type of content	Definition
Classification level	*PU*

1.7.2 Description of the result

E4.1.4b deals with the development of machine-readable formats based on the ontology formulated in E4.1.4a. Machine-readable formats are key for the automated processing and analysis of data.

The development resulted in four formats:

- Description of the ontology via OWL2
- The Data Specification and Description Format (DSDF), designed for specifying and requesting images for the training and analysis of AI-enable pedestrian detection algorithms
- The Action Description Language, designed to enable the specification and requesting of data that include pedestrian and camera movement and actions
- The Meta-Annotation-format designed for the annotation of simulated and real images, for the enhanced analysis of the performance of detection algorithms in regards to those annotations.

Some of the key working packages were:

- Conversion of the human-readable description into a tool-independent machine-readable format for generating simulated test data (format definition)
- Interpreter between human-readable format (e.g. SCODE output) and machine-readable format (e.g. xml)
- Collection of the requirements of the simulation tools from TP2 with respect to the input-given formats and their consideration in the format definition
- Cooperation with the tool vendors when testing the implement ability of the identified context dimensions identified and their variations

All list of formats used in KIA can be found here: List of Metadata Formats, Definitions.

1.7.3 Approach

There exist several needs for data description formats depending on the step in the KIA-Process. First, a data format for the ontology description is needed. For this the Web Ontology Language (OWL) is selected, since it is well established and widely used for such tasks. From the multitude of file formats for OWL, RDF/Turtle is selected, because it has the greatest compatibility with git out of the options. The second needed step in the KIA-Process is the requesting of data from the simulation tools. This step should be solved by the Data Specification and Description Format (DSDF). DSDF should be a JSON-based format that captures information of the ontology and the Zwicky-Boxes and enables the requesting of data from simulation. It is designed to represent a single point in time and focuses on the perspective of a selected camera. A translation of this format to the OpenScenario and OpenDrive formats is desired to enable the interoperability with arbitrary simulation frameworks, and was developed. Moreover, to extend the scope of the format, an action description language is developed, that enables the description of pedestrian and vehicle actions on top of the DSDF format and with that can describe temporal developments. The results of the format are not directly used in this project, but ensure the usability of the developed beyond the use cases of this project. The third step in the KIA-Process is the analysis of KI-Algorithms on datasets of the images from simulation. For this, a meta-annotation format is needed, that describes the attributes of the images from simulation in detail and so enables the analysis of the performance of KI-Methods in relation to these attributes. The meta-annotation format is based on the JSON-format. All mentioned formats are described in the following chapter.

1.7.4 Result

1.7.4.1 Data Specification and Description Format (DSDF)

The Data Specification and Description Format (DSDF) is a JSON-based format, that is primarily used in TP2 and is designed to enable the requesting of data from simulation tools. It translates the concepts of the KIA-Ontology into a data requesting JSON-format.

To ease the implementation of data requesting for other simulation framework providers, the translation to the OpenScenario and OpenDrive formats lend themselves. Since OpenScenario 1.2 missed some concepts that were necessary for the translation of the DSDF format, an extension of the OpenScenario specification initiated (https://code.asam.net/simulation/openscenario-1/openscenario/-/issues/377). As an example, the ability to define a sensor and attaching it to a vehicle or object was added. This can be used to define the camera position to simulate images from in OpenScenario. Also a pedestrian pose can be defined through the PedestrianPose enumeration:

Name	Description
standing	Pedestrian is standing up
sitting	Pedestrian is sitting down
lying	Pedestrian is lying down
ducking	Pedestrian is ducking

Name	Description
squatting	Pedestrian is crouching/squatting
bendingDown	Pedestrian is bending down
walking	Pedestrian is walking
running	Pedestrian is running
reeling	Pedestrian is reeling
crawling	Pedestrian is crawling
jumping	Pedestrian is jumping
falling	Pedestrian is falling

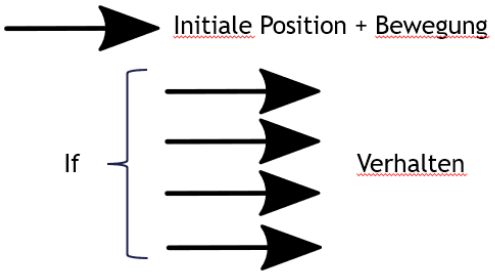
A full list of added concepts to OpenScenario can be found in https://code.asam.net/simulation/openscenario-1/openscenario/-/issues/377.

For the systematic generation of the requirement files, Mackevision defined a format for requesting images, based on the DSDF format. Bosch and ZF created a tool called NCAP Scenario generator (https://luxproject.luxoft.com/stash/mvc/projects/KIA/repos/tp4_ap4.1_e4.1.5_ncap_scenario_generator/browse) that enables the systematic generation of such files and a set of these files was created for the following NCAP like scenario: https://luxproject.luxoft.com/jira/browse/KIA-2564.

Bosch developed a methodology to convert and enrich synthetic data generated by Mackevision to a more user-friendly JSON format structure and enriching it by new meta information calculated by post processing measures. Building on the developed formats, analysis and experiments with meta data in regards to Data Coverage and Data Distribution were performed as part of Work Stream 8: Data Coverage and Data Distribution.

1.7.4.2 Action Description Language

In the context of KIA only the description of single points in time is necessary. Nevertheless, the excitability of the developed formats to the description of temporal aspects is desirable. This was pursued with the development of the Action Description Language. The goal was to describe a sequence of pedestrian actions in relation to the camera and the road geometry. For the description of actions, the definitions of Zwicky boxes had to be reworked. For example, the positioning of objects can only be set in the camera view through the Zwicky boxes. When describing the camera view for a single point it time this is sensible, but when looking at a temporal progression, not all objects are in the camera view at all times. Since, similarly to the DSDF format, the format should be compatible with as many simulation frameworks as possible, the action description language was designed to be translatable to OpenScenario. The basic concept of the action description is to define an initial position and motion for all objects and then further specify the behavior of these objects when they reach certain points in the map or in relation to the ego vehicle (camera):

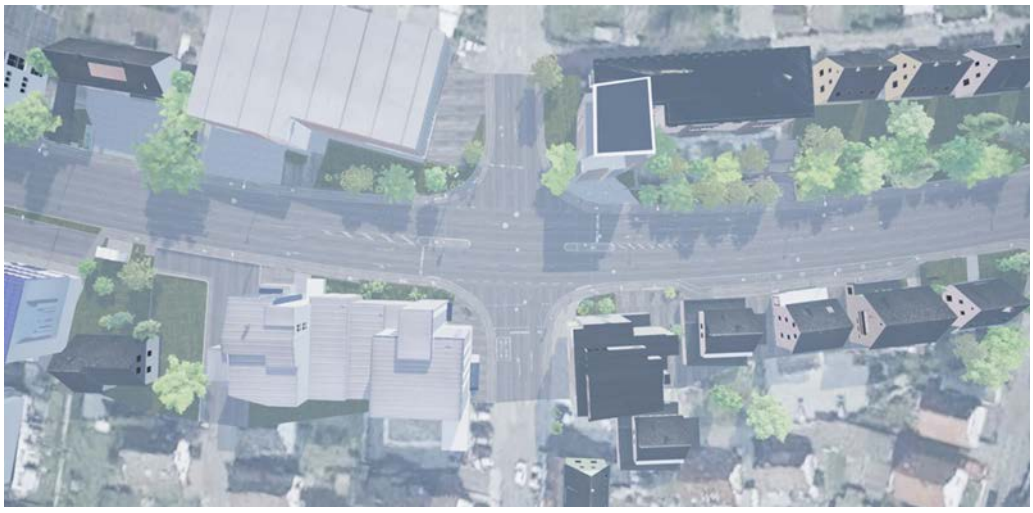


In OpenScenario this can be achieved through the Event - Trigger mechanisms. The development and testing of the Action Description Language is done by adapting code of the ScenarioRunner of the CARLA simulation framework (http://carla.org/).

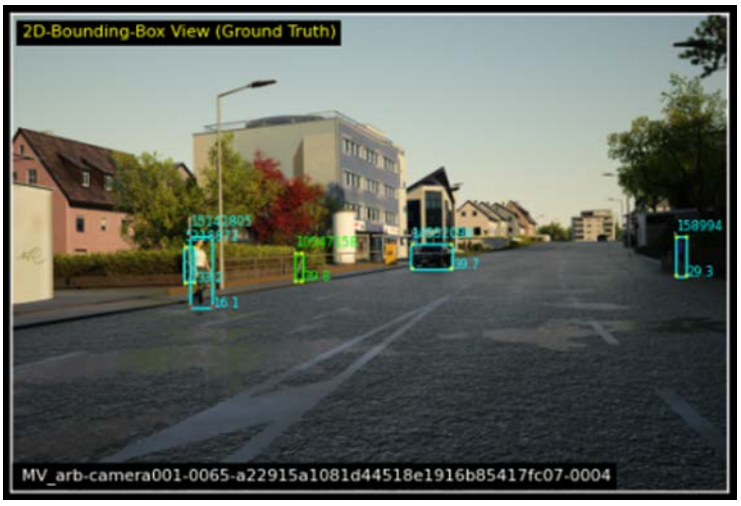
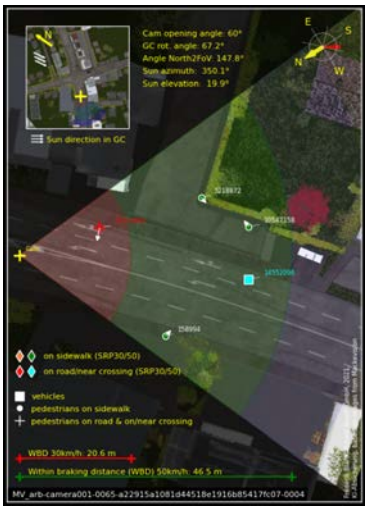
1.7.4.3 Meta-Annotation Format for extended Analysis

An image from simulationcontains much more information as is given through the requesting of such an image. Therefore, for the analysis of KI-Methods for pedestrian detection, the Meta-Annotation specification (4.1.4b Ergebnisbericht (M30) - Enriched metainfo description format) was developed. As a technical term for this format "general-globally-per-frame-analysis-enriched_json" was coined. In contrast to the Meta-Annotations need for DNN-Training (e.g., labels), this format includes further semantic information, which is required for the analysis, the testing, and the safety assurance of DNNs in the context of pedestrian detection. The format combines and aggregates information from multiple sources into one data format.

The annotation format specifies the ground context and the location of objects.:



Besides the description relative to the ground context, a description based on the camera view and a birds-eye view around the camera is given in the meta-annotation format:

Camera perspective & field of view	bird-eye-view
	

Per object instance in the image, the asset id of the object is given and resolved to add further semantic information available for the asset through the asset ontology. E.g., for pedestrians, age, gender, body size and form, skin, hair length, color and cut, clothes and their surfaces and colors, as well as the pose of the pedestrian is given. Moreover, the genral-globally-per-frame-analysis-enriched_json contains aggregated information of the following types:

- general-globally-per-frame-analysis_json
- 2D Bounding Box (2d-bounding-box-enriched_json)
- 3D Bounding Box (3d-bounding-box-fixed_json)
- 2D Skeletons (2d-skeletons_json)
- Contrast Performance Limiting Factors Information (Bosch contrast plf-information)
- Depth Information (depth_exr-fixed)
- Instance Segmentation (semantic-instance-segmentation-fixed_png)
- Train Information (general-globally-per-frame-train-fixed_json)

A detailed description of the general-globally-per-frame-analysise-enriched_json can be found in 4.1.4b Ergebnisbericht (M30) - Enriched metainfo description format.

1.8 E4.1.5 Final: nur projektintern für KI Absicherung verfügbar

2 AP4.2 Gesamtstruktur Argumentation und Sicherheitsanforderungen für KI-Funktion

3 AP4.3 Nachweisstrategie für eine „abgesicherte“ KI-Funktion

4 AP4.4 Entwicklung von Testmethoden zur Bestätigung der Wirksamkeit des Assurance Case

4.1 E4.4.1a Final: nur projektintern für KI Absicherung verfügbar

4.2 E4.4.1b Final: nur projektintern für KI Absicherung verfügbar

4.3 E4.4.2 Final: nur projektintern für KI Absicherung verfügbar

4.4 E4.4.3a Final: Argumentation einer Abdeckung des neuronalen Netzwerkes (zur Veröffentlichung)

4.4.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Report*
Group/Cluster	*E - Modell; Methodik; Konzept*
Type of content	*Concept*
Classification level	*PU*

4.4.2 Description of the result

4.4.2.1 Motivation

The need for thorough verification and validation (V&V) of deep neural networks (DNN) is currently undoubted and a major motivation for this project. Since there are yet no formal methods that would allow to prove that a neural network is good enough for a given task in a system context, V&V resolves to testing of the DNN with a finite set of sample images. This directly leads to the fundamental question of testing in practice: How to decide when a software (in this case a DNN) has been tested enough for a given task? While the actual test methods that are used for the definition of test sets is part of the project result E4.4.1, this result focuses on arguing that the neural network has been tested *enough*, which is also referred to as *test adaquacy*.

In classical software testing, arguing test adaquacy is usually assisted by the use of different kinds of coverage metrics that provide a measurable criterion when to stop testing. Thereby, coverage can be measured on different levels rangin from more high-level requirements coverage approaches down to structural white-box coverage metrics on the source code level. In recent year, there have been initial works trying to transfer a notion of coverage to DNNs. In this report, we present and discuss different coverage metrics for DNNs and discuss their effectiveness. Effectiveness means that a high coverage implies adaquacy of test data whereas insufficient test data can be revealed by lower coverage scores. Please note that the general effectiveness of KPIs is subject to project result E4.4.2 and that this report solely focusses on the aspect of coverage.

4.4.2.2 Notions of Coverage

In this section, we review the definition of coverage as it originates from the software testing domain. Thereafter, we present fundamental considerations from literature that form the basis of definitions of coverage for DNNs, which are currently the definitions of neurons and activations. In addition, we discuss the uses for coverage in a testing process.

4.4.2.2.1 Coverage in the Testing Domain

Coverage is a widely-used metric in software testing. Most well-known are structural coverage metrics based on source code such as statement coverage, branch coverage, or Modied Condition/Decision Coverage (MC/DC) used for white-box testing of small software units. However, coverage criteria are not restricted to code and can be selected based on the corresponding test level. Additional examples include coverage of requirements and respective equivalence classes on an embedded software / software requirement level. Hence on a lower level, there are white-box coverage metrics in which we measure based on internals of an implementation, while black-box coverage based on the software's purpose is used on a higher functional level. In summary, a coverage metric supports the testing process by providing a quantitative measure that must be adequate for the test level and purpose.

From an academic perspective, Bron et al. (Bron et. al 2005) discuss properties that a coverage metric should satisfy. They state that a coverage metric should:

1. have an underlying static **model**,
2. be (practically) **coverable**, i.e. we can achieve full coverage,
3. be **actionable**, i.e. we know how to continue, and
4. have an **exit strategy** when reaching full coverage.

For a statement coverage goal, it is quite apparent how these properties are useful. We can build a coverage model upon the given source code. As long as we find a non-covered statement, we either formulate a corresponding new test or verify via analysis that a given statement is not reachable. If statement coverage is reached, we may stop testing on code/unit level for this coverage goal.

4.4.2.2.2 Coverage in the Context of DNNs

In this section, we describe definitions from literature on what resembles a coverable unit (or neuron) in a DNN and how a different metrics differ in their scopes.

4.4.2.2.3 Neuron Definitions

In source-code-based coverage metrics, the coverable items originate from the software structure, i.e., these are statements, branches, conditions etc. In DNNs, we intuitively could use neurons as a coverable item. However, there does not exist a unique definition in literature of what constitutes a neuron in a DNN. There are two general views one could take on coverage.

The *biologically inspired view* considers a neuron an element that receives input signals, aggregates them, and depending on some threshold res a signal to other neurons. Translated to ML, a neuron comprises an element in a convolutional or linear layer including the subsequent ReLU. Other layers, such as pooling layers, may not be considered.

The *(software) architecture inspired view* considers a neuron any switching element that aects the computation paths of the function (i.e. of the DNN). This would include any layer that comprises switching elements, such as fully-connected layers, non-differentiable activation functions like ReLUs as well as max pooling layers (but not average pooling layers). Some non-linear, but differentiable activation functions, e.g. ReLU, may also be split into a finite number of equivalence classes to account for non-linearities. Conceptually, the switching elements divide the huge input space into local linear halfspaces that (i) could be individually tested but (ii) increase combinatorially with model depth.

Regretfully, the different definitions from the literature do not clearly follow the above views and use different neuron definitions.

The differences mainly present themselves in two respects: (i) How to handle spatial layers such as convolutional layers and (ii) which layers to include. The earliest related work is DeepXplore (Olah et al. 2018) whose neuron denition is further used in DeepGauge (Ma et al. 2019), Dissector (Tian et al. 2017), and the work by Kim et al. (Kim et. al 2019). This definition reduces convolutional layers to a single neuron per filter, i.e. all elements in a single filter are reduced to their mean activation which removes fine-granular information about spatial coverage. Other works also consider the spatial distribution of convolutional layers such as Sun et al. (Schwalbe and Schels 2020). In their neuron definition, pooling layers and final predictions are excluded. As a conclusion, there exists no generally accepted definition on what constitutes a coverable unit in a neural network and, thus, each metric should be accompanied by a neuron definition.

4.4.2.2.4 Hierarchy of Coverage Scopes

As in classical software testing, coverage metrics may have different scopes. Assuming an architecture based on ReLU non-linearities, where $\mathrm{ReLU}(x) = \max(x, 0)$, we may count a neuron as covered if for a given input x its value after application of the ReLU is non zero. Full coverage, with respect to this measure, would thus be achieved if a given testset X can be found such that for any neuron of the DNN at least one datasample x_i in X exists that the unit is covered. A generalisation to pair-wise metrics is straightforward, but owing to the often sequential structure of (then feed forward) DNNs a further restriction to adjacent layers might be suitable. Ultimately, the scoping could be widened to complete coverage of all possible combinations of activations across the entire network.

While Neuron coverage as single point measure is a weak metric, pairwise coverage metrics like Sign-sign coverage (SSC) (Sun et al. 2019) constitute stronger metrics. Additionally, as shown in analysis of two layer networks (Ergen and Pilanci 2020), the negative side of pre-activations may results in interesting switching behavior of a neural network and should also be investigated. Going from neuron coverage metric focussing on single neurons to pair-wise coverage metrics and from pair-wise to even larger scopes makes these metrics potentially stronger as they intuitively require more input data to be satisfied, but it also exponentially increases the number of items that need to be covered, which might ultimately be impossible due to the high effort for large DNNs.

4.4.2.3 Uses for Coverage

In a testing process, coverage may serve different purposes that should be distinguished. These are:

- Argue Test End / Test Sufficiency, i.e., if I reach 100% coverage, I can stop testing.

- Identify Novel Test Cases, i.e., what additional test can I do to progress to 100% coverage?
- Establish confidence in the value of a KPI or metric, i.e., if the data I used for computing the KPI only covers a part of a network, I should doubt the confidence of the resulting value.

In this respect, using coverage for identifying novel test cases could be relevant to the works on test methods in E4.4.1. Establishing confidence in the value of a KPI is particularly relevant for arguing the effectiveness of methods and measures in E4.4.2.

4.4.2.4 Approaches for Measuring Coverage

There exist different classes of approaches for measuring coverage depending of the information and coverable items they use. In principle, we may distinguish:

- **Input coverage**
  Input coverage uses a model of legal / available inputs that are provided to the DNN and argue coverage with respect to the input model. One technique in this category is described in Semantic Input Model Based Coverage below.
- **Structural white-box coverage**
  Structural white-box coverage applies neuron and activation definitions as defined above for checking whether all neurons of the DNN have been covered by a test set. We discuss effectiveness of existing approaches from literature in White box, structural coverage below.
- **Coverage on proxy network / latent space**This class of approaches builds a separate proxy model, e.g., based on the latent space features of an autoencoder, for providing coverable items and tries to cover these items.
- **Coverage of semantic concepts**This class of approaches tries to identify semantic concept representations in neural network and to cover these (see e.g. Testing with Concept Activation Vectors (TCAV) by Kim et al.)
- **Output coverage**Output coverage refers to the outputs of the neural network, i.e., the bounding boxes or semantic segmentation maps, and creates a coverage criterion on these.
- **Combined Approaches**Combined approaches combine two or more approaches from the above categories to argue coverage.

4.4.2.5 Results on Coverage for Neural Networks

In this section, we summarize the results that were obtained by the different partners.

4.4.2.5.1 Semantic Input Model based Coverage (Bosch)

We developed an approach to black-box, domain oriented coverage leveraging a semantic domain model as developed in AP4.1. Such descriptions that contain each possibly relevant effect are inherently large and we would like to test each of these effects and ultimately also the interactions between these effects. As such, we need testing approaches and corresponding coverage approaches that better scale with model size as well as with the degree of interaction

that needs to be considered. For black-box testing of SW, there already exists such an approach with Combinatorial testing. This approach was developed to avoid the combinatorial explosion of testing all interactions (i.e. the full power set of possibilities) of an input model. We transferred this approach towards input coverage for DNNs and performed several experiments.

1. We analyzed the input coverage of two public datasets (Cityscapes, A2D2) by using the ground truth labels of the semantic segmentation maps as a proxy for an input model. The results on this experiment can be found in our published paper.

2. We repeated the experiment on tranche #3 data from KI-A again taking the semantic segmentation maps as a proxy for an input model. The results showed, as expected, that several domain elements were not present in the data.

3. We utilized a real input model from AP4.1 and the meta-annotations provided by MackeVision for data tranches #4, #5, and #6 to analyze input coverage of these data tranches. The results of the experiments are documented as part of the evidence workstream 08 - Data Coverage and Data Distribution and in the project result E4.4.2 as we discuss the effectivity of the approach.

The obtained results, particularly for the third experiment, are promising. For some domain elements, we could observe a correlation between low data volume for specific combinations of domain elements and an insufficient performance of the neural network.

4.4.2.5.2 Structural White-box Coverage on MNIST (Bosch, Fraunhofer IAIS)

In this work, we first study in detail different notions of coverage and the effect on the number of coverage items that need to be considered. Second, based on a standard definition of coverage (based on individual neurons) discussed above, we check coverage on a standard image classification task (MNIST). Here we perform different studies on MNIST, e.g. we study the effect of test data augmentation on coverage in different layers as well as a sensitivity analysis on image classes available in the test set. Finally, we compare test time augmentation to coverage-based testing approaches. The experimental results show that:

1. Coverage definitions matter greatly and have a large effect on coverage elements and therefore on resulting coverage. Hence, these must be clearly defined and compared in any empirical evaluation.

2. Coverage metrics may not provide the information we would expect (depending on task and network architecture as well as neuron definition). As an example, for our experiments, we saw that even with only a subset of classes we can achieve full coverage of later layers.

3. Coverage-based test generation has in this set of experiments no advantage over test data augmentation (that does not need the feedback from the network).

More details on the approach and our results can be found in the published paper.

4.4.2.5.3 Structural White-box Coverage on Illustrative Task (Bosch)

Inspired from the result on MNIST, we created an illustrative machine-learning task that is amenable to formal analysis for deeply investigating the properties of structural coverage metrics. Our task is: Given 4 input pixels in the domain [-1;1]^4, count how many pixels are

greater than 0. For this task, we create and train a simple fully-connected network. Then, we encoded the trained network including its weights and biases into an SMT-solver and employed the SMT solver to analytically compute inputs for the trained network that cover a specific neuron / coverage item. Then, the SMT-solver either returns an input for the network or it returns that the problem cannot be solved, i.e., the activation cannot be established on the network under the given input constraints. We use the SMT-solver to optimize for three different structural coverage metrics and compare the results to two sampling approaches as a baseline.

We came to the following key results:

- As expected, the more elaborate a coverage metric is the more test samples are necessary for achieving coverage.
- Intuitively, we would expect that more test samples imply higher coverage as new samples will only be generated if they increase coverage. However, this does not hold true for all cases.
- There are cases, where we can achieve 100% coverage when allowing *All* samples, but where we cannot achieve 100% coverage when allowing only *ODD* samples.
- There are cases for SSC where even with samples from *All*, no 100% coverage is possible. For a realistically sized network, this case cannot be distinguished from the previous case where the missing coverage was introduced by restricted the input to in *ODD*.
- Samples from *All* that are outside *ODD* have a high cross-entropy loss, i.e., the network performs badly on these. As a consequence, it is at least questionable whether it is meaningful to extend a test set by such samples just for increasing coverage.
- The results were similar for both models, the possible coverages were slighty higher for the adversarial trained model.

In summary, coverage metrics are hard to interpret if the metric converges to a value below 100%. This means if we determine that a certain neuron is not coverable, we cannot determine (i) whether it is coverable with *some* input and moreover (ii) with some input inside our ODD. As a consequence, the coverage metrics are not actionable in a sense that we can derive information on which particular inputs we lack in our test set. In addition, filling the test set with out-of-ODD samples for increasing coverage has questionable effect. Since they are out-of-ODD, the network under test will usually not be trained on such samples and, thus, the network needs to extrapolate for classifying them and results in high loss in our samples in our case.

The experiment with more details on the results has been accepted as a joint chapter with Fraunhofer IAIS in the upcoming KI-A book.

4.4.2.5.4 Using Coverage to Increase Confidence in KPIs (Continental)

We conducted our experiment on the following setting:

- DeeplabV3+ trained on the KI-Absicherung "Tranche 3" data split as kindly provided by the project partner Intel
- Classification results on the official "Tranche 3" test set

- Compute activation in the batch norm units of the latent space using forward hooks

We compute 3 different performance metrics individually on the test images:

- Pixel Accuracy
- mean intersection over union
- F1 score

Experiments:

We compute the mean performance of a randomly drawn sample from the test set. This is done 1000 times each. A box plot for each of the three accuracy metric can be seen below. It can be observed that, while the mean performance over all sample sizes remains approximately stable, the variance of the individual values of the metric is reduced when using more samples as the basis of the evaluation of a metric.

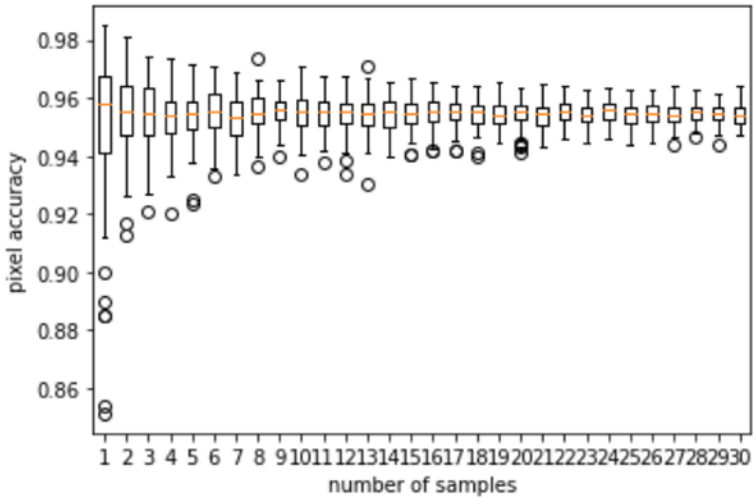


Figure 1: numerical results for KPI "pixel accuracy"

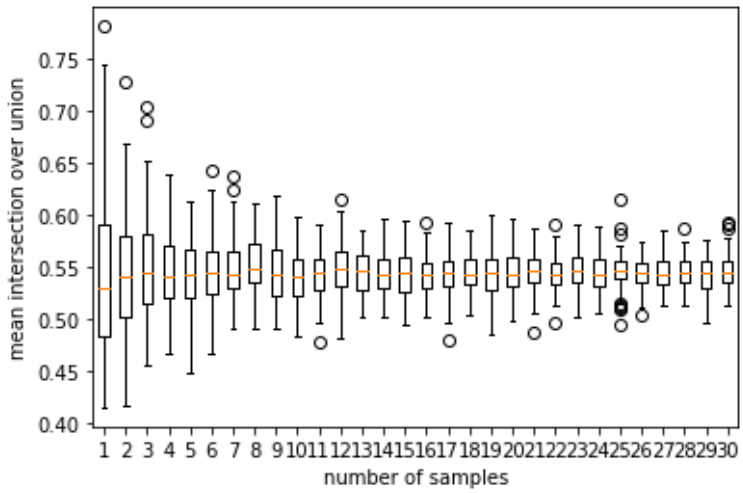


Figure 2: numerical results for KPI "mean intersection over union

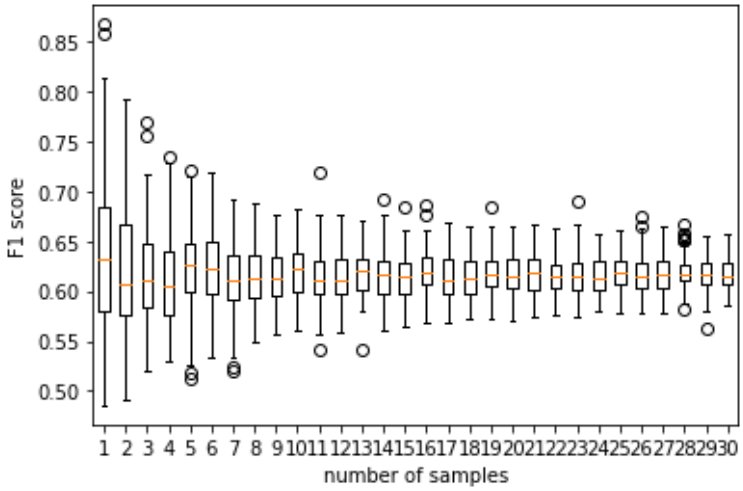


Figure 3: numerical results for KPI "F1 score

As a second experiment, we compute the accumulative neuron coverage over the images in a sample while varying the threshold for which a neuron is counted as "activated". We plot the span of the performance metric as described above against this neuron coverage for every defined threshold. We can observe that in all cases the span drops by increasing neuron coverage, which is not surprising since this is more or less equivalent to adding images into the sample. Neuron coverage seems to run into saturation pretty quickly and decreases when selecting a larger threshold. This is also not surprising since this just filters more neurons out of the list of neurons that are regarded to be activated. However, selecting a lower threshold might lead to higher coverage in the far end of the graph but this also means that for single images the activation is already high.

From this, we draw the following conclusions:

- The fact that we can not reasonably achieve a "full" neuron coverage could be an indicator that the images in our test are quite similar. This is very intuitive since we have data from an simulated urban traffic scenario from the same project. One might speculate that adding more diverse data may increase variance in the neuron coverage over different images.
- The average performance of the for testing only one image is visually already quite representative for the whole test. A reason for that could be that testing on a pixels level renders quickly a representative

4.4.2.6 Utilizing Attention-based Heatmaps for Robustness Analysis (ZF)

Beyond of the other described points in the result, the heatmaps from the SSD300 Release3v1 were analyzed to build an argument about the robustness. For the calculation of the heatmaps the method "heatmap based consistency validation" from TP3 was used, which is build on the HydraPlus-Net approach. With these heatmaps it's possible to do a white box analysis and make an assumption on which kind of input data the network decides it's output. In the first approach the heatmaps are generated with the feature extractor that was trained on another dataset than the testing dataset, which was used to analyse the heatmaps. This includes the assumption, that the network is able to generalize into the testing domain. The results with this network have shown, that the class based heatmap consistency in figure 1 doesnt's correlate well with the

detection performance of the network. After this observation the SSD300 network was adapted to a SSD512 input resolution and combined with the aforementioned heatmap method. The feature extractor was trained with the both domains of data producers (physics and realtime rendering) to avoid a domain gap and exclude the assumption about the generalization capability. The resulting heatmaps for the SSD512 network show some specific "bias" in the images (figure 2 and 3) independent of rendered image type and make it difficult to analyze the results. One approach for tackling the bias problem would be to filter it out by applying a threshold to the heatmap, this has the drawback, that overall lower values in the activations won't be considered in the later analysis of the consistency validation. On the other side it would also be possible to argue that only activations over a specific threshold are valid. Additionally the analysis for a coverage of the heatmap activations of the semantic input domain like pedestrians within a certain distance, rotation, occlusion, contrast and pose would be possible. In general we assume that the network is brittle in those images, where the activation energy is low on the pedestrians. For the validation of the hypothesis a baseline network with unique heatmaps is the foundation, as otherwise the results can be misinterpreted. In conclusion the results from the heatmap evaluation didn't show unique heatmaps from both network either SSD300 or SSD512.

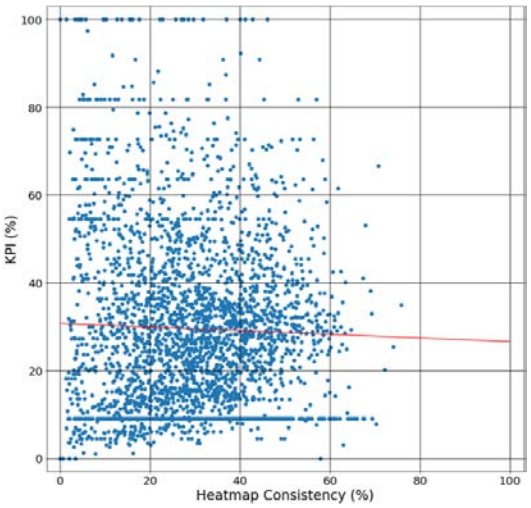


Figure 1: Evaluation of heatmap consistency and performance based on the initial used SSD300 feature extractor

Figure 2: Original image

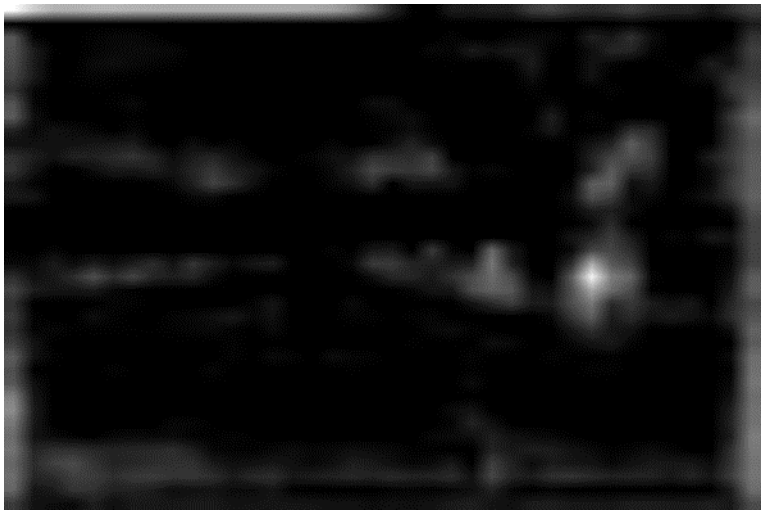


Figure 3: heatmap from SSD300 feature extractor netowork

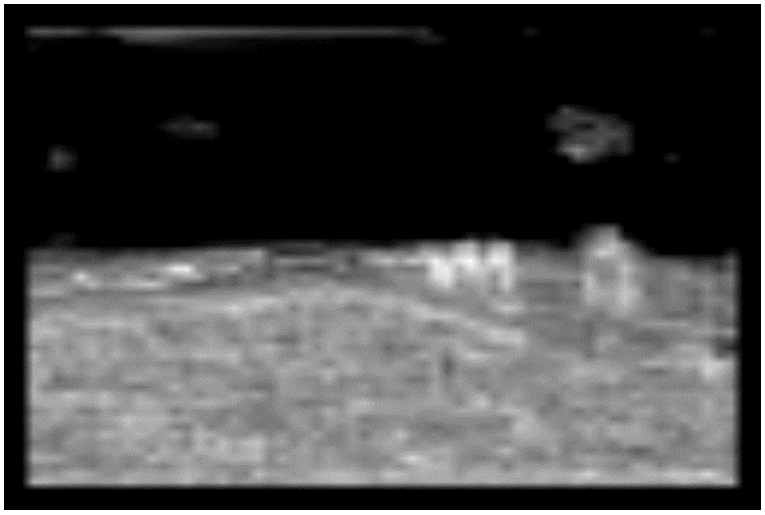


Figure 4: heatmap from SSD300 feature extractor netowork

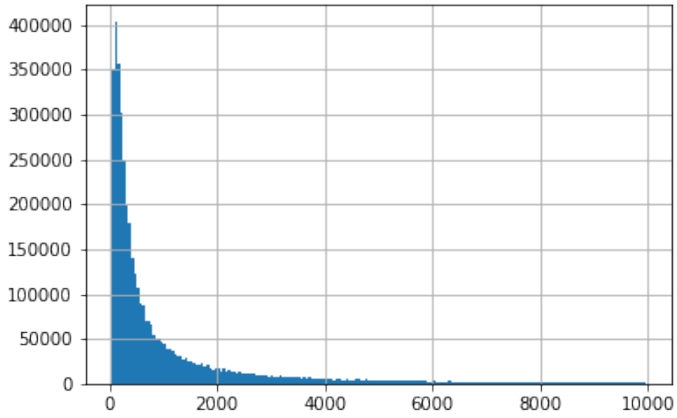


Figure 5: bounding box size (pixels) distribution

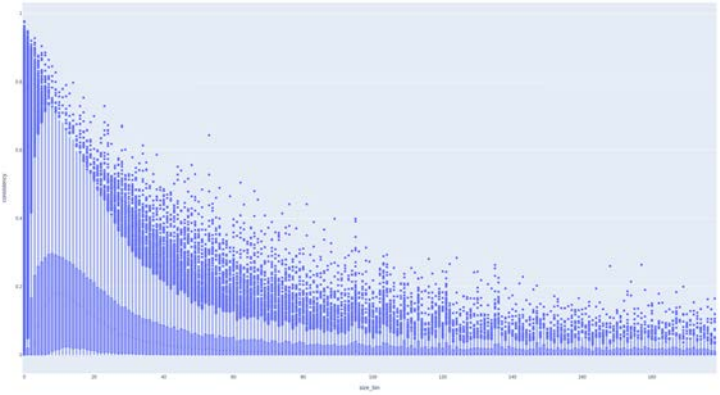


Figure 6: Bounding box heatmap consistency related to each size bin with width of 500 pixels

In summary it's shown that the consistency depends on the size of the single pedestrian instances (figure 6) within the image and therefore, the feature extractor isn't capable to indicate the ground truth bounding boxes independent of their sizes. Additional a test dataset must be used that contain an equal distribution of bounding box sizes when analyzing the impact of image pertubations on the heatmaps and the resulting consistency. In the figure 5 it's shown that the size distribution of the bounding boxes isn't equal and strongly biased in the test dataset. As both requirements are not fulfilled, the analysis for the brittleness based on heatmaps is spared out and the heatmap extraction method must be further improved and the dataset prepared by either filtering or request the specific data.

4.4.3 Final Results and Conclusions

The obtained results show that measures and metrics for test adequacy are still in their infancy for neural networks. Even though we obtained initial, promising results for input coverage, the general effectiveness of the method in still to be shown on more use cases. In contrast, the results for structural coverage on the network itself or on the latent space are not convincing and hard to interpret for large neural networks as they are used for pedestrian detection. As a result, such metrics should be used with great care if at all.

4.4.4 Literature

(Bron et al. 2005) Bron, A., Farchi, E., Magid, Y., Nir, Y., Ur, S.: Applications of synchronization coverage. In: Symposium on Principles and Practice of Parallel Programming. pp. 206-212 (2005)

(Ergen and Pilanci 2020) Tolga Ergen and Mert Pilanci. Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models. In Silvia Chiappa and Roberto Calandra, editors, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 4024–4033. PMLR, 26–28 Aug 2020.

(Gladisch et al. 2020) Gladisch, C., Heinzemann, C., Herrmann, M., Woehrle, M.: Leveraging combinatorial testing for safety-critical computer vision datasets. In: Workshop on Safe Articial Intelligence for Automated Driving (2020)

(Kim et al. 2019) Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: International Conference on Software Engineering (2019)

(Ma et al. 2019) Ma, L., Juefei-Xu, F., Xue, M., Li, B., Li, L., Liu, Y., Zhao, J.: Deepct: Tomographic combinatorial testing for deep learning systems. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 614-618. IEEE (2019)

(Olah et al. 2018) Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., Mordvintsev, A.: The building blocks of interpretability. Distill (2018). https://doi.org/10.23915/distill.00010

(Pei et al. 2017) Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: Automated whitebox testing of deep learning systems. In: Symposium on Operating Systems Principles. pp. 1-18 (2017)

(Schwalbe and Schels 2020) Schwalbe, G., Schels, M.: A survey on methods for the safety assurance of machine learning based systems. In: European Congress Embedded Real Time Software and Systems (2020)

(Sun et al. 2019) Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural Test Coverage Criteria for Deep Neural Networks. ACM Transactions on Embedded Computing Systems (TECS), 18(5s):1–23, 2019

(Tian et al. 2017) Tian, Y., Pei, K., Jana, S., Ray, B.: DeepTest: Automated testing of deep-neuralnetwork-driven autonomous cars. In: arXiv:1708.08559 (2017)

(Xihui Liu et al. 2017) Xihui Liu, Haiyu Zhao, Maoqing Tian, Lu Sheng, Jing Shao, Shuai Yi, Junjie Yan, Xiaogang Wang: HydraPlus-Net: Attentive Deep Features for Pedestrian Analysis. In: HydraPlus-Net: Attentive Deep Features for Pedestrian Analysis (thecvf.com)

4.5 E4.4.3b Final: nur projektintern für KI Absicherung verfügbar

4.6 E4.4.4a Final: nur projektintern für KI Absicherung verfügbar

4.7 E4.4.4b Final: nur projektintern für KI Absicherung verfügbar

5 AP4.5 KI-Teststrategie und KI-Testplan als Ausgangspunkt für eine Produktfreigabe

5.1 E4.5.1a Final: Empfohlene Testmethoden für KI- Funktionen im Bereich Objektdetektion (zur Veröffentlichung)

5.1.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Document*
Group/Cluster	*D4; D5*
Type of content	*Guideline*
Classification level	*PU*

5.1.2 Description of the result

Result is described in one common result summary for E4.5.1a and E4.5.1b, in: E4.5.1b Final: Vorschlag einer generalisierten Teststrategie für KI-Funktionen im Bereich Objektdetektion

5.2 E4.5.1b Final: Vorschlag einer generalisierten Teststrategie für KI-Funktionen im Bereich Objektdetektion (zur Veröffentlichung)

5.2.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Document*
Group/Cluster	*D*
Type of content	*Guideline*
Classification level	*PU*

Editorial remark: this section summarizes E4.5.1a and 4.5.1b.

5.2.2 Description of the result

The result of E4.5.1 is a test strategy for AI functions in object detection. The test strategy is a set of recommendations on test method classes that can be used to test AI functions in the context of object detection. Test methods of TP3 and TP4 are in focus, but further methods are also considered. In order to be able to specify a test strategy, a lifecylce model for ML development has to be introduced first, such that the test activities can be located in that lifecycle.
The test strategy does not introduce an own risk register, but risks are tracked and managed via the assurance case.

5.2.2.1 ML life cycle

Machine Learning (ML) is a special form of Artificial Intelligence (AI) in which computer algorithms learn to form or parameterize predictive models - e.g. Deep Neural Networks (DNN) - from datasets in an iterative optimization process against some cost function. This training process is usually automated. In the last decade, ML has emerged as a key success factor to solve complex automation tasks due to performance reasons and as ML supersedes the need for manual feature engineering in many cases. Especially, in computer-vision tasks, DNN-based ML-approaches have shown supperior performance and good generalization capabilities compared to previous approaches. In the development of automated driving (AD) functions, the approach of supervised ML, which requires the data to be labeled with a ground truth reference, is widely used. However, despite technological advances, research questions remain open about improving the level of trust and quality that can be placed on ML-based functions (e. g. a DNN-based pedestrian detection function) applied in an Operational Design Domain (ODD) with open context [1] as is typical for AD applications. Here a 100% complete interpretation of the environment in which the ML-based function works in is not possible, and thus, the ML-based function might operate beyond its original design intent. This requires monitoring and an iterative approach. As ML models can solve very complex perception tasks in one step, the underlying ML model(s) usually dominate(s) the behavior of an ML-based function, and thus, govern(s) the overall performance. The performance of a ML model is mainly determined by three factors:Analyze domain shift between the dataset and the data obtained

- the ML model architecture implemented as software (SW) code, which mainly determines the learning capacity and capability,
- the training strategy including optimization criteria, cost functions and hyperparameters, which allows a weighting or priorization of certain training aspects,
- and the data sets for training, validation and testing, which are the most variable parts that may grow or change during iterative development.

In contrast to classical feature engineering, ML models learn all their functional behavior from the data, and not by straightforward programming based on non-/functional requirements. Hence, requirements towards the I/O-behavior of an ML-based function are implicitely given by the data, too. Therefore, the data sets play a significant role in ML. In order to understand ML models we need to understand the data they are trained and tested on, but this can only be done iteratively. The quality of the ML model depends on the quality of the data used for training or testing (w.r.t. balance, diversity, coverage of the ODD, decision boundaries, corner cases, out-of-distribution samples, ...) in combination with appropriate verification and validation. The data quality - and with it the confidence in ML - demands an iterative, systematic and structured process incorporating hierarchical requirements engineering for the quality, composition and coverage of the ML data sets as prerequisite for evidence- and data coverage-based safety argumentation.

For classical SW development, predominantly in automotive context, V-model-based systematics are well established, integrating hardware (HW, aka electronics hardware) engineering and SW engineering processes within an overall process landscape. However, there is no established systematics yet for integrating data requirements for ML-based functions (e.g. perception functions) or typcial ML workflows themselves in the overall engineering systematics.

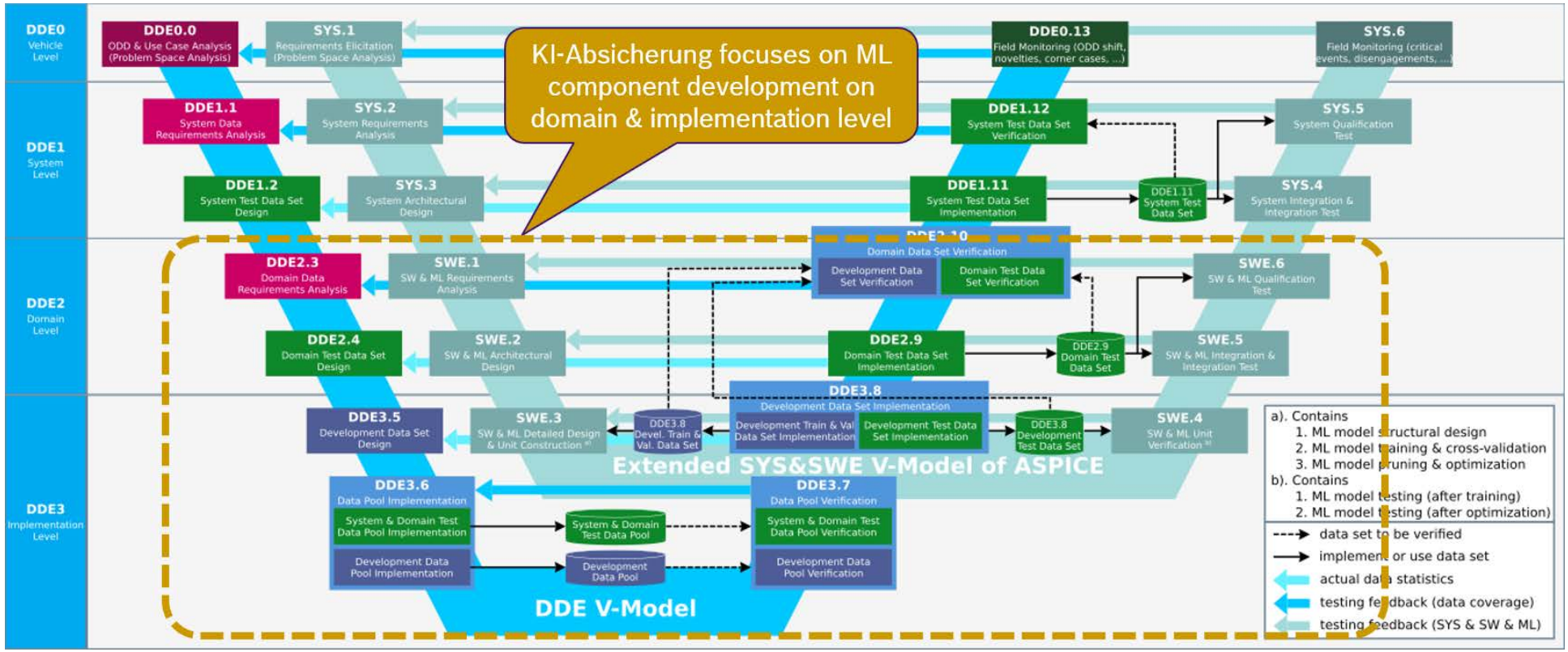


Fig. 1: Extended DDE Process Reference Model now also including the process steps DD0.13 and SYS1 on design level DDE0, which is the basis for the ML Life Cycle model of KI-Absicherung

In KI-Absicherung, we propose a consistent ML Life Cycle model to include ML workflows and to systematically specify, implement and verify data sets for training, validating and testing SW units / SW components with ML models. This ML Life Cycle model of KI-Absicherung is based on the Data-driven Engineering (DDE) process [2], which integrates data requirements and data-driven engineering for ML in a generic, structured and consistend process based on a DDE V-model for the data that collaborates with the well-known SYS/SWE V-model (ASPICE) via the data sets (s. fig. 1). The actual architectural design, the implementation, the training and testing of ML models is included in the SYS/SW part, whereas the training and test data sets are provided by the data part of the DDE process model. This approach closes the gap in the process landscape and addresses the open context challenge in requirements engineering in the new domain of Data-Driven Engineering (DDE). With this systematic

approach, we can derive data requirements from an ODD model and iteratively increase validation & verification (V&V) coverage for the system's ML-based functions within an open context environment. The major intent is to develop predictable ML-based functions based on data sets with well known contents while reducing the number and combinatorial complexity of textual artifacts as much as possible. Therefore, we discretize the ODD, or the data space, respectively, and break it down to a manageable level. Additionally, we reduce combinatorial complexity in requirements management by separating the scenario- and open-context-related data requirements from the rather generic non-/functional SYS/SW/ML requirements (e.g. performance requirements). A recombination is automatically accomplished during training or testing on the respective data sets. The DDE process also allows inclusion of any connectionistic ML or V&V methods.

In KI-Absicherung, the DDE process has been reviewed by the stakeholders of AP4.5. In AP4.5, the DDE process reference model has been extended by the process steps DDE.0 and SYS.1 on design level DDE.0 and field monitoring in the process steps DDE0.13 and SYS.6. What is more, detailed block diagrams of the ML workflows have been derived, which are added to the DDE process, s. fig. 2 and fig. 3. The ML workflows are integral parts of the process steps SWE.1 to SWE.6, which are also labeled as ML.1...ML.6 to symbolize that we may have SW & ML units including ML models here. Design of SW & ML unit architecture, ML model implementation and training as well as model optimization and pruning are part of process step SWE.3 / ML.3, whereas model verification, ML unit integration and integration testing as well as SW & ML unit testing are included in process step SWE.4 / ML.4.

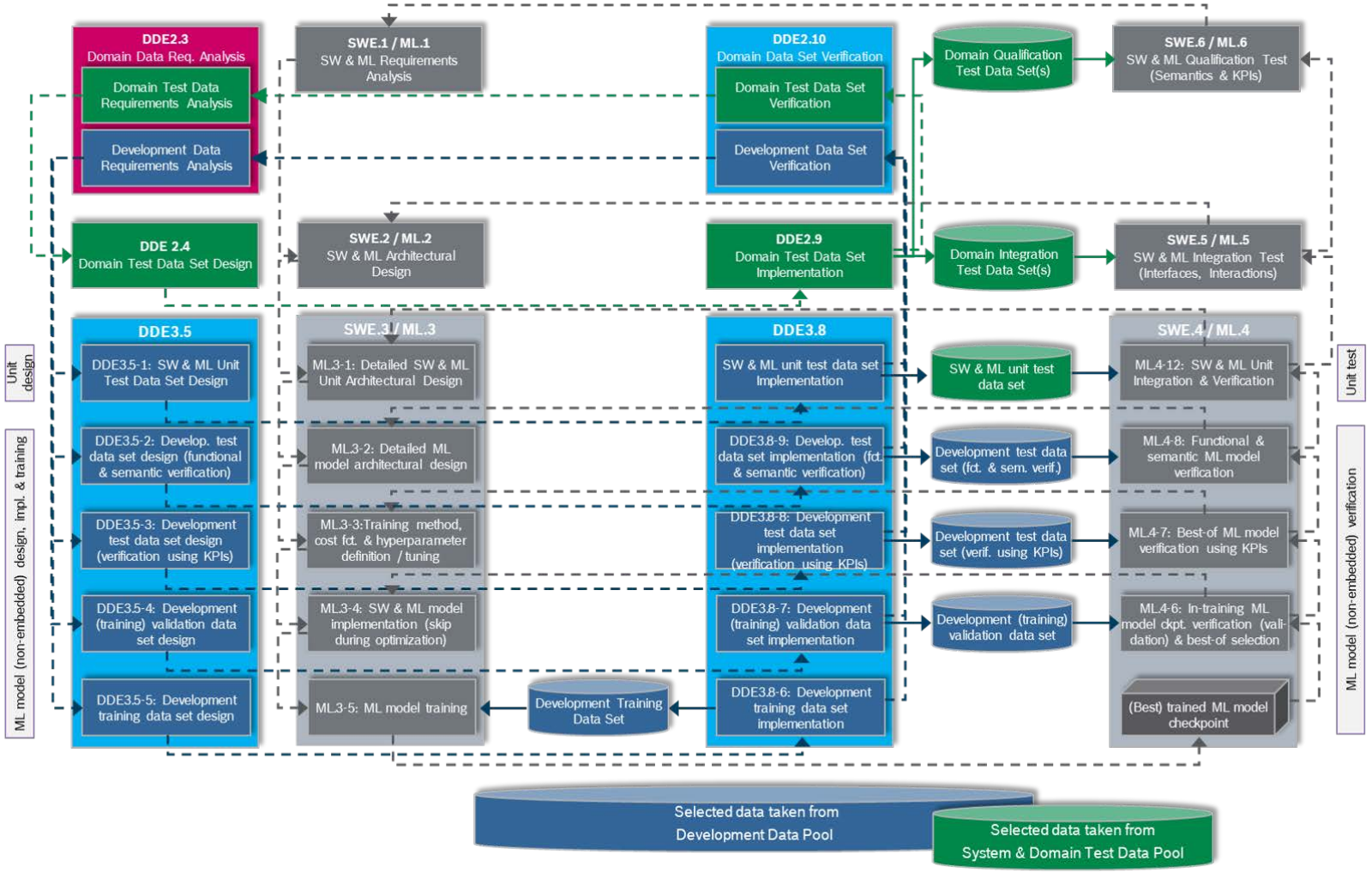


Fig. 2: Detailing of the DDE process steps DDE.3.5, SWE.3 / ML.3, DDE3.8 and SWE.4 / ML.4 for stage 1 focusing on the design and unit test of SW units with ML models as well as on the design, implementation, training, cross-validation and verification of ML models as part of SW units / SW components. This also includes data requirements analysis on domain level as well as design, implementation and verification of the different development training, development validation and development test data sets for the ML models, or the test data sets for the SW units / SW components with ML models, respectively.

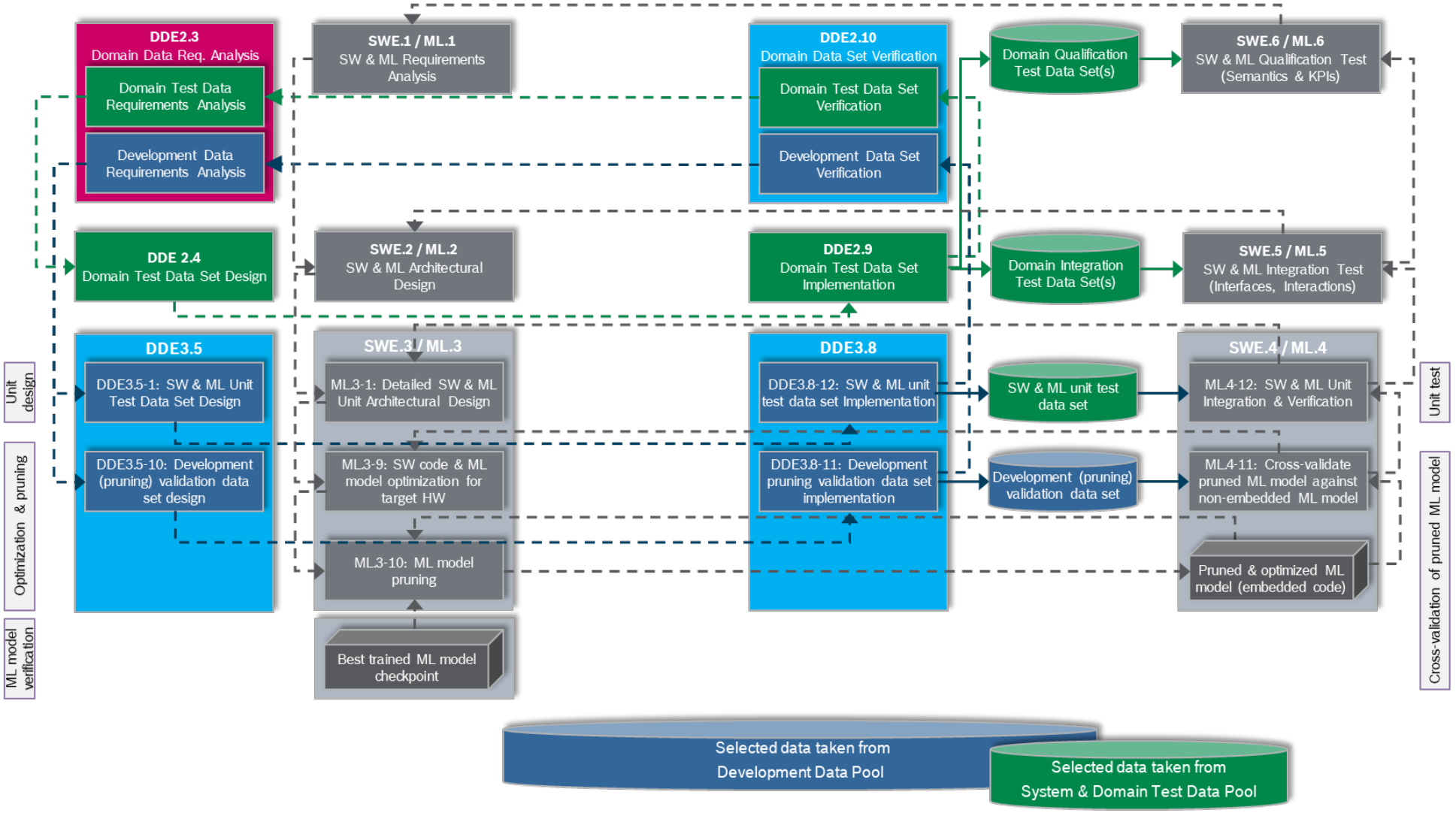


Fig. 3: Detailing of the DDE process steps DDE3.5, SWE.3, DDE3.8 and SWE.4 focusing on the design and unit testing of SW units with ML models as well as on SW code optimization for the target HW including ML model pruning and cross-validation against the original ML model.

The block diagrams in fig. 2 and fig. 3 give an insight into the sub-process steps in SWE.3 / ML.3 and SWE.4 / ML.4. The process steps DDE3.5 and DDE3.8 as well as SWE.3 / ML.3 and SWE.4 / ML.4 are sub-divided into multiple steps and into two different stages. Stage 1, which is shown in fig. 2, covers the detailed SW & ML model architectural design, the SW & ML model implementation, the ML model training and optimization as well as the ML model cross-validation & verification. Stage 2, which is shown in fig. 3, covers the SW code optimization for the target HW including ML model pruning and cross-validation of the pruned model (embedded code) against the original best-of ML model (non-embedded code). Stage 2 is executed after stage 1 is completed. This may involve multiple iterations between SWE.3 / ML.3 and SWE.4 / ML:4 as well as multiple iterations in the DDE loop involving steps DDE2.3, DDE3.5, DDE3.8 and DDE3.10 if it turns out during development that changes of the data requirements or the data set contents are needed because an ODD shift or an ODD extension has been realized at a certain point of time during development, for example.

Below you can find a documentation of the Test Strategy of KI-Absicherung that has been built based on the V&V methods developed and evaluated in KI-Absicherung. The Test Strategy has been mapped to the respective ML Life Cycle model / DDE process reference model within the scope of KI-Absicherung focusing on the domain and implementation level DDE2 and DDE3, respectively. The mapping between the ML Life Cycle model / DDE process reference model and the Test Strategy of KI-Absicherung is shown in fig. 4.

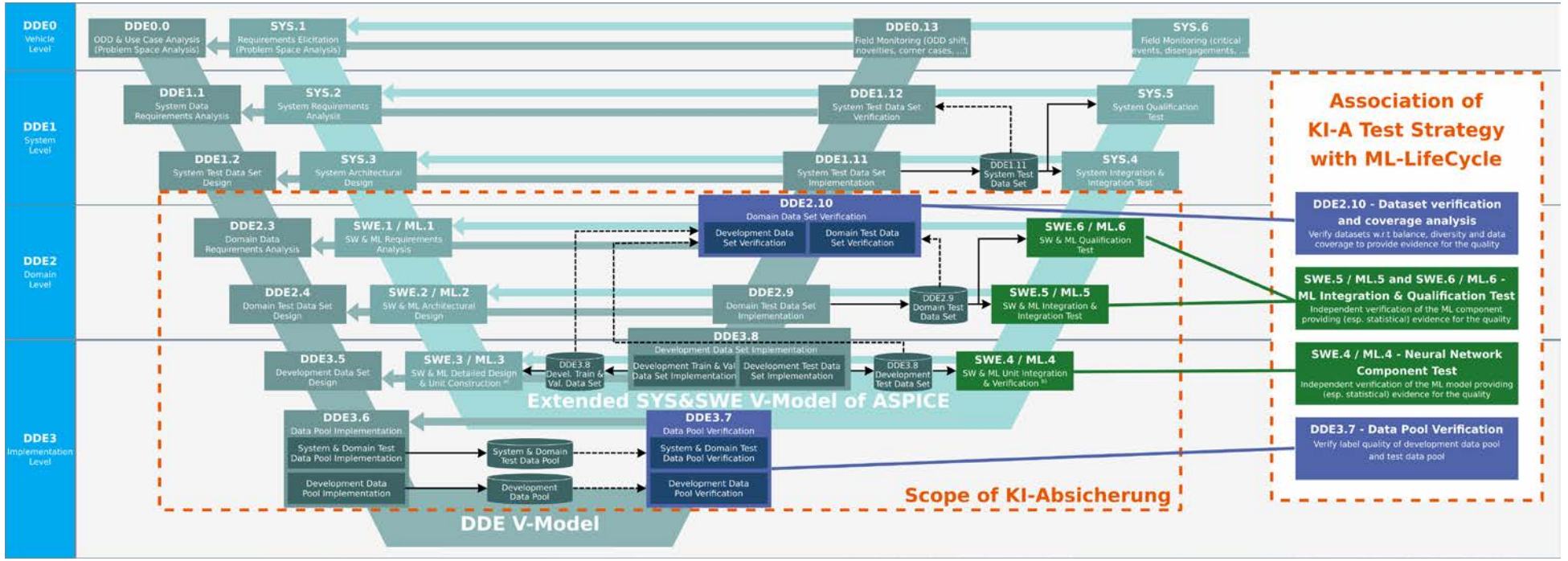


Fig. 4: Association of the Test Strategy of KI-Absicherung (based on the V&V methods developed in KI-Absicherung) with the DDE process reference model, or the ML Life Cycle Model, respectively.

5.2.2.2 Test strategy

The test strategy contains a set of recommendations for (groups of) methods to be used for testing. These can in general be use-case-specific. Therefore, it is important to note that the scope of the test strategy in this project is limited to DNN-based object classification/detection in perception. Nevertheless, (groups of) methods mentioned in this test strategy might also be applicable to similar use cases besides pedestrian detection, which is studied in the project.

5.2.2.3 Test subprocesses

Note: In a concrete project, the ML test activities should be aligned with the tests on SW and system level, such that tests that require a very broad data basis happen if possible on earlier levels in the process such that potential problems that may arise from a combination of interactions with other components are detected and addressed on the appropriate design levels.

5.2.2.3.1 Dataset verification & coverage analysis (DDE process step: DDE2.10 domain dataset verification)

Note: See reference [3] for data safety in general

Objective:

- Verify the training dataset, the development dataset and the test dataset (e.g. w.r.t. ODD coverage, balance/fairness, diversity, coverage of edge/corner cases, coverage of descision boundaries or out-of-distribution samples)
- Provide qualitative and quantitative evidence for the quality and coverage of the datasets

In:

- Unit-under-test: training dataset, development dataset, test dataset, assurance dataset
- ODD definition

Out:

- Test report with evaluation results w.r.t. data coverage

Subactivities:

- Preliminaries:
    - Verify independence of the test dataset to the development dataset and the training dataset
        - Methods: e.g. direct dataset comparison, analysis of implicit dependencies (possibly with PFMEA), e.g. data split on frame level instead of sequence level
- General dataset analysis:
    - Analysis of data collection gaps:

- Methods: Create an ontology with equivalence classes for all relevant attributes, analyze whether all attributes of the ontology are covered, possibly combined with an n-wise combinatorial approach
  - Note: For further guidelines to analyse the input space refer to E.4.1.2.a.
  - Typically, the data analysis is iterative, starting with a frist set of relevant attributes and adding further attributes, which turn out to have a significant effect on the ML performance, by considerung more combinatorial classes or increasing the combinatorial space (i.e. by increasing n in an n-wise combinatorial approach).
    Note: Not only the existence of data for all relevant attributes/combinations is relevant, but also their frequency distribution in order to identify underrepresented data classes in the respective dataset.

- Analysis of data fidelity:
  - Ensure that the data corresponds to that of the target sensor up.
  - Methods, e.g.:
    - Use the target sensor setup for data collection.
      - Note: this can already be checked in step DDE 3.7.
    - Analyze domain shift between the dataset and the data obtained from the target sensor setup.
      - Mapping to KI-A Safe AI Mechanism Taxonomy (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree"):
        - 1. Safe AI Mechanisms > Dataset Optimization > Outlier Detection/Anomaly Detection/Corner Case Selection > End-to-End (Anomaly) Classification > Input & Output Observation Seperately (e.g. measure distributional shifts)
        - 2. Safe AI Mechanisms > Dataset Optimization > Inspection of Domain Mismatch (of train/test data)
- Note: Label quality is addressed in test subprocess *Dataset label quality analysis* (DDE3.7 *Data pool verification*).

- Test dataset specific analysis:
  - Verify ODD coverage of test dataset.
    - Methods: See analysis of data collection gaps.

- Verify that safety relevant cases are substantively represented in the test dataset and the assurance dataset.
  - Methods: i.e. identify safety relevant cases in the test dataset and the assurance dataset and combine this analysis with the KPI analysis in section NN component test to check whether in combination with the KPIs, the amount is sufficient.
- Analyze statistical relevance of test dataset:
  - Methods:
    - Compare the frequency distribution of features covered by the test dataset with the real/natural frequency distribution of features occuring within the ODD.
    - Determine statistical hardness (e.g. standard deviation) of claims that can be obtained from the test dataset.
    - Maybe: Examine neuron coverage achieved in testing.
      - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Search-based Testing> Coverage Guided > Coverage of Latent Space / Concolic Testing (white box) > Variants of Neuron Coverage (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Maybe: Examine equivalence class coverage using the n-wise combinatorial approach.
      - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Search-based Testing> Coverage Guided > Coverage of Input Space (black box) > Ontology > Combined Dimension Coverage (e.g. Combinatorial Testing) || OR: Individual Dimension Coverage (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")

- Note: In subprocess *NN model component test*, search based testing and other techniques may be used to identify regions in the input space with low ML performance. This motivates extending the training dataset, the development dataset, the test dataset and the assurance dataset over the iterations in the development process.

5.2.2.3.2 Dataset label quality analysis (DDE: DDE3.7 data pool verification)

Objective:

- Verify label quality of the training dataset, the development dataset and the test dataset

In:

- Unit-under-test: Training dataset, development dataset, test dataset, assurance dataset
- Labelling specification

Out:

- Test report with evaluation results w.r.t. label quality

Subactivities:

- Analyze label quality:
    - Methods, e.g.:
        - Create equivalence classes for labels, analyze samples from the dataset, such that each equivalence class is covered (e.g. occuded/non occluded/truncated bounding boxes).
        - In case of manual labels: Perform statistical analysis.
        - Analyse data that leads to false NN prediction results, priorized by a loss function.
        - Maybe: For synthetic data, perform a PFMEA (process failure-mode and effects analysis) to discover how tool or asset faults can lead to faults in the data and to introduce mitigation measures.
        - Tool qualification (ISO 26262-8, section 11) might also be considered.

5.2.2.3.3 NN model component test (DDE: SWE.4 / ML.4)

Note: Test activities typically reveal weaknesses that motivate improving/extending the datasets in iterations.

Objective:

- Provide intependent verification of the trained model.
- Provide qualitative and quantitative (esp. stochastic) evidence for the quality.

In:

- Unit-under-test: Generated final NN model with final parameters on the target HW or emulator, with minimal pre- and post-processing as it is needed to conduct the tests
- KPI calculation methods
- KPI requirements, incl. confidence intervals
- Test dataset and further assurance data (as far as it is not generated within the used test methods)

- ODD definition

Out:

- Test report with: Calculated KPIs and evaluation results

Subactivities:

For the test method classes listed in the subactivities below, we provide an estimation on the utility of the respective test method class what concerns pedestrian detection based on experience within project "KI Absicherung".

- Preliminaries (KI-Absicherung estimation: very useful)
    - Define strategy that allows for iterated testing without implicit learning of the test dataset.
        - Methods: e.g. addition of fresh data or organizational independence without sharing of concrete test results (to be used with caution).
    - Verify that the results on safety relevant cases in the test data and assurance data are sufficiently reflected in KPIs.
        - Methods: e.g. identify safety relevant cases in the test dataset and in the assurance dataset and check whether KPIs react if the results for the safety relevant cases were negative (e.g. replace NN result by random results).
- Statistical testing based on test dataset (KI-Absicherung estimation: very useful)
    - Note: The statistic evidence has to be considered for all tests. Techniques from hypothesis testing are important to draw conclusions.
    - Test-end criterion: The confidence interval of the test results on the respective test data is as small as required.
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Statistical Testing / Hypothesis Testing (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
- Model analysis/review (KI-Absicherung estimation: useful)
    - Analysis w.r.t. fairness/non-discrimination
        - Method: Compare NN performance on fairness/discrimination sensitive data subsets.
    - Analyze NN performance on data subsets in order to identify anomalies. Study the NN behavior for underrepresented data classes and out-of-distribution data samples (see also: robustness tests).

        - Methods: e.g. visual analytics approach from Fraunhofer IAIS, MECH-116617
            - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Interpretability > Local Interpretation (i.e. on single data points) > Visual Analytics (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Tests based on network topology/training method (e.g. translation invariance)
        - Methods: Analyze potential weaknesses of NN architecture/training and examine this on respective data samples (e.g. data samples with rotated objects with different orientation)
        - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Dataset Optimization > Data Augmentation (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
        - Analysis using interpretability methods
            - Methods: See interpretability methods in TP3 method table, e.g. heatmap based attention consistency validation (MECH-156340).
            - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Interpretability (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Test-end criteria: All selected methods used, all analyzed weaknesses compared to a predefined thresholds or goals
- Tests based on corner cases, expert knowledge (KI-Absicherung estimation: useful)
    - Note: This activity will typically request additional, specific assurance data, from the data collection/generation process.
    - Methods: Systematically create corner cases from expert knowledge and AP2.2 corner case methodology.
        - Note: For example, this may include tests based on NCAP scenarios. Corner cases may be motivated by all process levels (e.g. vehicle testing, field monitoring).
    - Mapping to KI-A Safe AI Mechanism Taxonomy (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree"):
        i. Safe AI Mechanisms > Dataset Optimization > Outlier Detection / Anomaly Detection / Corner Case Selection

ii. Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Corner Case and Expert-based Testing

- Test-end criteria: All analyzed weaknesses compared to a predefined threshold or goal

- Search-based testing (KI-Absicherung estimation: useful)
  - Input space coverage-guided search-based testing:
    - Method: "Methodology to identify influencing parameters" (TSTM-0011 "Valerie")
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Search-based Testing > Coverage Guided > Coverage of Input Space (black box) > Ontology > Individual Dimension Coverage || OR: Combined Dimension Coverage (e.g. Combinatorial Testing) (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Test-end criterion: Input space coverage for selected dimensions
  - Performance-guided search-based testing:
    - Method: "Search-based testing for computer vision" (TSTM-0005)
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Search-based Testing > Performance Guided (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Test-end criterion: The number of identified issues per iteration falls repeatedly below a predefined threshold for selected modifications of input data (i.e. a kind of convergence w.r.t. the input data modifications).
  - Latent space coverage-guided search-based testing:
    - Method: Coverage-guided fuzzing testing framework (TSTM-0001)
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Incomplete / Sampling-based Test Methods (adaptive or non-adaptive) > Coverage Guided > Coverage of Latent Space / Concolic Testing (white box) (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
    - Test-end criterion: Latent space coverage for selected coverage metric, possible also depending on selected input data modifications.

- Coverage-guided testing (KI-Absicherung estimation: useful)

- Note: Coverage may rather be regarded as complementary to other test methods in providing a completeness metric than as a standalone test method.
- Coverage criteria will potentially request or generate additional assurance data.
- Input space coverage:
  - Method: Combinatorial testing (TSTM-0004)
  - Note: This should automatically result from activity Dataset Verification
- Neuron or latent space coverage:
  - Methods: e.g. coverage-guided fuzzing testing framework (TSTM-0001), TSTM-0007
- Test end criteria: Selected coverage goals are achieved.
- Mapping to KI-A Safe AI Mechanism Taxonomy: See above (Search-based testing)

- Robustness analysis (KI-Absicherung estimation: very useful)
  - Note: Test methods for robustness analysis typically generate additional assurance data.
  - Robustness tests w.r.t. natural corruptions:
    - Methods: e.g. AugMix, Neurocat work in AP4.4, and from TP3: Robustness Testing Framework (MECH-133124), Photometric Robustness Estimation (MECH-418874)
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Robustness Testing / Metamorphic Testing / Data Augmentation > Natural Corruptions (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
  - Analyze adversarial robustness:
    - Methods: e.g. Aversarial Robustness Toolbox (ART), Adversarial Attacks Assessment (MECH-232004), Distorted Images Assessment (MECH-376159), Adversarial Augmentation of Point Clouds for Domain Generalization (MECH-018870)
    - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Robustness Testing / Metamorphic Testing / Data Augmentation > Natural Corruptions > Adversarial Attacks (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")
  - Evaluate response on input distribution shift and out-of-distribution data
    - Methods: e.g. data augmentation bei out-of-distribution objects

- Test end criteria: Selected kinds and selected degree of input modifications covered

- Formal verification
  - Note: Applicability depends on whether relevant properties can be specified and proven. For object detection this is limited.
  - Example: Formal verification of robustness properties (MECH-936804)
  - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Complete Test Methods > Formal Verification (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")

- Analyze resource limitations (runtime, …)
  - Mapping to KI-A Safe AI Mechanism Taxonomy: Safe AI Mechanisms > Verification > Resource Consumption Test Methods (more info: check Taxonomy Tree; related methods: Methodenkatalog column "Mapping of Mechanism in TP3 Taxonomy Tree")

5.2.2.3.4 ML integration test (DDE: SWE.5 / ML.5 + SWE.6 / ML.6)

Unit-under-test: Two or more composed final NN models with final parameters on the target HW together with the code that combines them including the a minimal pre- and post-processing needed to conduct the tests.

Example: Perception DNN composed with VAE for ODD detection and a SW unit merging their results

Analogy: SW-SW integration test

If the NN models are guaranteed to be stochastically independent, this step can be omitted and normal integration testing can directly be done. Otherwise, specific methods are required that reflect the ML nature of the unit under test. A threads to stochastical independence e.g. is that the same dataset was used.

Applicable methods:

- Same methods as above in NN model component test

5.2.3 References

[1] S. Burton, L. Gauerhof, and C. Heinzemann, "Making the case for safety of machine learning in highly automated driving," in Proc. International Conference on Computer Safety, Reliability, and Security, Dec. 2017, pp. 5–16. Link: https://link.springer.com/chapter/10.1007/978-3-319-66284-8_1

[2] Zhang R., Albrecht A., Kausch J., Putzer H. J., Geipel T., Halady P.: A requirements engineering approach for machine learning in automated driving. 29th IEEE Int. Req. Eng. Conf., Notre Dame, South Bend, USA, Sept. 20-24, 2021. Link: https://ieeexplore.ieee.org/document/9604596.

[3] The SCSC Data Safety Initiative Working Group: Data Safety Guidance. Version 3.3, Feb 20212, https://scsc.uk/scsc-127F.

[4] ISO 26262 "Road vehicles — Functional safety", 2nd edition, 2018.

5.3 E4.5.2 Final: nur projektintern für KI Absicherung verfügbar

5.4 E4.5.3a Final: Bericht zum Stand-der-Technik und Dokumentation der Anforderungen an die Teststrategie für KI-Funktionen im Bereich Objektdetektion (zur Veröffentlichung)

5.4.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Document*
Group/Cluster	*Test strategy*
Type of content	Literature evaluation and test strategy requirements
Classification level	*PU*

5.4.2 Description of the result

As a basis for creation of the test strategy in E4.5.1, a state-of-the-art (SOTA) analysis was conducted and requirements for the test strategy were derived from it and provided to E4.5.1. The considered literature is listed below. In general there is only limited work specifically on testing of ML. The requirements we elicitated can be grouped into the following classes: general requriements, requirements on the input and the output, respectively, of the test process, requirements on how methods are recommended, recommended methods, requirements on the test process and requirements on data w.r.t. testing. Many requirements are not ML specific. For the purpose of KI Absicherung, the most important classes are recommended methods and requirements on data.

5.4.2.1 Literature

- ISO/PAS 21448: Road vehicles — Safety of the intended functionality. International standardization organization, 2019.
- ISO 26262: Road vehicles — Functional safety. International standardization organization, 2nd edition, 2018.
- ISO/IEC/IEEE 29119: Software and systems engineering — Software testing. International standardization organization, 2016.
- Automotive SPICE® Process Reference Model, Process Assessment Model Version 3.1, November 1 2017, URL: http://www.automotivespice.com/
- ISO/IEC 25010: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International standardization organization, 2011.

- ISO/IEC 25022: Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use. International standardization organization, 2016.
- ISO/IEC 25023: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality. International standardization organization, 2016.
- ISO/IEC/IEEE 15288: Systems and software engineering — System life cycle processes. International standardization organization, 2015.
- IEEE P1012: Standard for System, Software, and Hardware Verification and Validation. IEEE, 2018.
- DIN SPEC 92001-1: Artificial Intelligence – Life Cycle Processes and Quality Requirements – Part 1: Quality Metamodel. Deutsches Institut für Normung, 2019.
- HLEG AI. "High-level expert group on artificial intelligence." Ethics guidelines for trustworthy AI (2019). https://ec.europa.eu/futurium/en/ai-alliance-consultation
- ANSI/UL 4600: Standard for Safety for the Evaluation of Autonomous Products (Draft Version). ANSI/UL, 2019.
- ISO/TR 4804: Road vehicles — Safety and cybersecurity for automated driving systems — Design, verification and validation. International standardization organization, 2020.
- Waymo Safety Report, https://waymo.com/safety/ , 2019.
- Zhang, J. M., Harman, M., Ma, L., & Liu, Y.. Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering, 2020.
- Cremers, A. B., Englander, A., Gabriel, M., Hecker, D., Mock, M., Poretschkin, M., ... & Wrobel, S. (2019). Vertrauenswürdiger Einsatz von Künstlicher Intelligenz. Handlungsfelder aus philosophischer, ethischer, rechtlicher und technologischer Sicht als Grundlage für eine Zertifizierung von Künstlicher Intelligenz. Fraunhofer-Institut für intelligente Analyse-und Informationssysteme (IAIS). https://www.iais.fraunhofer.de/content/dam/iais/KINRW/Whitepaper_KI-Zertifizierung.pdf. 2020.
- Burton, Simon & Gauerhof, Lydia & Heinzemann, Christian. Making the Case for Safety of Machine Learning in Highly Automated Driving. International Conference on Computer Safety, Reliability, and Security. LNCS 10489. 2017.
- EU Proposal for Regulation: Artificial intelligence Act. https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence. 21.04.2021.
- ISO/IEC AWI TR 5469: Artificial intelligence — Functional safety and AI systems. International standardization organization, to appear.

- Cluzeau, J. M., Henriquel, X., Rebender, G., Soudain, G., van Dijk, L., Gronskiy, A., ... & Polak, R.. Concepts of design assurance for neural networks (CoDANN). Public Report Extract Version, 1, 1-104. 2020.

5.5 E4.5.3b Final: Ergebnisse der Gremienarbeit (zur Veröffentlichung)

5.5.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Document*
Group/Cluster	*D*
Type of content	Evaluation
Classification level	*PU*

5.5.2 Description of the result

In order to facilitate the adoption of project results, in particular the test strategy, in standardization processes, the projects was in continuous exchange with relevant strandardization activities and also contributed to the setup of the standardization activity ISO PAS 8800 "Road Vehicles — Safety and artificial intelligence".

5.5.2.1 Persued activities

5.5.2.1.1 DIN Normungsroadmap KI

Project members contributed to the areas of AI in mobility and logistics and lead the working group on quality and certification in the first edition of DIN Normungsroadmap KI.

5.5.2.1.2 ISO TR4804 "Road vehicles — Safety and cybersecurity for automated driving systems — Design, verification and validation", ISO TS5083 "Road vehicles — Safety for automated driving systems — Design, verification and validation"

Project members of KI Absicherung took part in the commentation phase of ISO TR4804. They contributed the KI Absicherung point-view to the finalization phase of the tech report. Members of KI Absicherung are now active in the follow-up project ISO TS5083, which in particular tries to address AI in automated driving in a broader scope than solely perception with DNNs.

5.5.2.1.3 ISO PAS 8800 "Road Vehicles — Safety and artificial intelligence"

Building upon ISO TR 5469 "Artificial intelligence — Functional safety and AI systems" and TR4804 a new work item proposal on functional safety of AI in road vehicles has been made to ISO with contribution of project partners. The proposal has been accepted and works in the same timeline as ISO TS5083, with end date October 2023. Both documents are planned to be aligned, with the idea that AI safety topics not specifc to automated driving are supposed to be located in PAS8800. In the current phase of the standardization project, several project members contribute in working groups of PAS8800 and thereby bring project results into the standardization process.

5.6 E4.5.3c Final: Zertifizierungsmethodik zum ordnungsgemäßen Einsatz der Entwicklungsmethoden (zur Veröffentlichung)

5.6.1 Formal Classification

Criteria	Classification according to VHB
Type of result	*Document*
Group/Cluster	
Type of content	Beschreibung der Regeln eines zertifizierten Entwicklungsprozesses
Classification level	*PU*

5.6.2 Description of the result

5.6.2.1 Motivation

The purpose of this result is to evaluate the methodology of KI-Abischerung in the broader context of trustworthy AI and to compare KI-Absicherung with complementary approaches for developing certification methodologies for trustworthy AI.

5.6.2.2 Approach

Fraunhofer IAIS is leading the Project "Zertifizierte KI" (https://www.zertifizierte-ki.de), which aims at fostering requirements and procedures supporting the development and certification of trustworthy AI systems. Together with the German Federal Office for Information Security (BSI) and the German Institute for Standardization (DIN) as well as other research partners, concepts and test procedures for the certification of artificial intelligence (AI) systems are developed. The aim is to ensure technical reliability and responsible use of the technology. Industrial requirements are taken into account through the active involvement of numerous associated companies and organizations representing various industries such as telecommunications, banking, insurance, chemicals, and trade.

In the project KI-Absicherung, a close contact and exchange with the project "Zertifzierte KI" has been established, including mutual presentation and discussion of results on project meetings and workshops.

in the project "Zertifizierte KI", an audit catalogue for guiding the development and for checking the trustworthiness of AI has been released (https://www.iais.fraunhofer.de/de/forschung/kuenstliche-intelligenz/ki-pruefkatalog.html). This audit catalogue has been analyzed and presented in a AP 4.5 workshop. It has been shown that the principle methodology developed in KI-Absicherung complies with the general risk-based approach presented in the audit catalogue. The audit catalogue presents a methodology that accounts for the achieved mitigation of AI related risks in six relevant dimensions (human oversight and agency, fairness, privacy, security, reliability and transparancy). The principle idea behind the certification approach is to achieve a structured risk analysis and documentation of mitigation measures along these six dimensions. In particular, application specific criteria for measuring AI risks and their mitigation have to be defined explicitly.

5.6.2.3 Conclusion

The general concept for trustworthy AI developed in Zertifizierte KI requires an application specific, risk-based approach and evaluation along the identified dimensions of trustworthiness. The KI-Absicherung methodology is a concretization of the general proposed approach in particular for the dimension reliability.