HAL
open science

From Simulation Data to Test Cases for Fully Automated Driving and ADAS

Christoph Sippl, Florian Bock, David Wittmann, Harald Altinger, Reinhard German

To cite this version:

Christoph Sippl, Florian Bock, David Wittmann, Harald Altinger, Reinhard German. From Simulation Data to Test Cases for Fully Automated Driving and ADAS. 28th IFIP International Conference on Testing Software and Systems (ICTSS), Oct 2016, Graz, Austria. pp.191-206, 10.1007/978-3-319-47443-4_12 . hal-01643731

HAL Id: hal-01643731

https://inria.hal.science/hal-01643731v1

Submitted on 21 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

From Simulation Data to Test Cases for Fully Automated Driving and ADAS

Christoph Sippl[1,2], Florian Bock[2], David Wittmann[3], Harald Altinger[1], and Reinhard German[2]

[1] Audi Electronics Venture GmbH
Sachsstr. 20, 85080 Gaimersheim, Germany
{christoph.sippl,harald.altinger}@audi.de
[2] Department of Computer Science 7,
Friedrich-Alexander-University, 91058 Erlangen, Germany
{florian.inifau.bock,reinhard.german}@fau.de
[3] Chair of Automotive Technology,
Technical University of Munich, Boltzmannstr. 15 85748 Garching, Germany
{wittmann}@ftm.mw.tum.de

Abstract. Within this paper we present a new concept on deriving test cases from simulation data and outline challenging tasks when testing and validating fully automated driving functions and Advanced Driver Assistance Systems (ADAS). Open questions on topics like virtual simulation and identification of relevant situations for consistent testing of fully automated vehicles are given. Well known criticality metrics are assessed and discussed with regard to their potential to test fully automated vehicles and ADAS. Upon our knowledge most of them are not applicable to identify relevant traffic situations which are of importance for fully automated driving and ADAS. To overcome this limitation, we present a concept including filtering and rating of potentially relevant situations. Identified situations are described in a formal, abstract and human readable way. Finally, a situation catalogue is built up and linked to system requirements to derive test cases using a Domain Specific Language (DSL).

Keywords: Virtual validation, ADAS, fully automated vehicles, simulation, test case generation, DSL

1 Introduction

Today's driver assistance functions and emergency systems help to avoid accidents and support the driver in critical situations. As the system boundaries are clearly defined, test cases can easily be specified. On the contrary, Advanced Driver Assistance Systems (ADAS) and fully automated vehicles ensure safety and comfort while driving in a normal mode. Here, defining all relevant test cases poses problems for developers due to the large amount of dynamic objects, including pedestrians and cyclists in urban traffic as well as the variety of priority rules and traffic guidance. So far, comprehensive test concepts and structured

test case generation, such as Equivalence Class Partitioning (ECP), Boundary Value Analysis (BVA) and Predicate Testing improve the efficiency of software testing as stated by Eo *et al.* [1], but are not well-suited to identify all possible and relevant situations for fully automated vehicles and ADAS.

Therefore, simulation-based development and new concepts for virtual validation are needed, instead of testing new driving functions with the help of many thousands of test kilometres. There are already numerous tools for simulation-based function development and testing, thus driving simulators with realistic, environment-sensitive behaviour of road users (e.g. pedestrians, cyclists and cars) can be used to generate a huge amount of data. This data contain new situations, which are relevant for testing certain driving functions. An automatic identification of these situations and the comparison with an existing set of situations in the test suite improve the overall test coverage of fully automated driving functions and ADAS. Manual inspection and filtering of the relevant situations or describing test cases is not recommended with respect to time and budget.

2 Related Work

In general, test cases are described by analysing the obligatory system requirements. In case of ADAS and fully automated driving functions, this can be done by evaluating data, produced by model-based simulation or empirically collected data. Zofka *et al.* [2] presented an innovative data-driven method and a concept contrary to previous approaches in order to create critical traffic situations from recorded sensor data. This concept allows reconstruction and parametrization of real world traffic scenarios. These reconstructed test scenarios can be re-simulated by deviating parameters in order to evaluate and test ADAS components. This approach may modify already observed and identified situations, but cannot detect completely unknown events.

Prior Schuldt *et al.* [3] presented a method to assign test cases automatically to X-in-the-loop simulation techniques using quality criteria. In [4], a modular virtual test repository is presented to reduce the number of required test cases for validation of driving functions by systematic test case generation with consistent test coverage. This approach improves the overall test process by using simulation techniques and provides evaluation methods. However, using this method, generated test cases are derived from predefined parameters which have impact on the system specification and requirements, scenario catalogues and existing guidelines and standards. Thus, complex and not yet identified situations are not taken into account. Stellet *et al.* [5] summed up challenging tasks on testing fully automated vehicles and ADAS and worked out, why automated driving functions cannot be tested by defining system level criteria. Stellet *et al.* argue, that "such concepts are too simplistic for future continuously intervening automated driving functions". In their work, a number of research questions are pointed out that remain unanswered to date. One of these questions is: "How to overcome the dilemma of testing the entire complexity of real-world traffic?"

To ensure consistent terminology regarding the terms *scene*, *situation* and *scenario* we follow the definitions given by Ulbrich *et al.* in [6]. They reflected various definitions and pointed out their understanding of the terms with regard to fully automated driving. Reduced to the key facts and following the definition given in [6], the terms are described as below:

Scene. "A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities.[...]"

Situation. "A situation is the entirety of circumstances, which are to be considered for the selection of an appropriate behavior pattern at a particular point of time. It entails all relevant conditions, options and determinants for behavior. A situation is derived from the scene by an information selection and augmentation process based on transient [...]as well as permanent goals and values. Hence, a situation is always subjective by representing an element's point of view."

Scenario. "A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions and events as well as goals and values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time."

Aim of This Work. This paper drafts a concept to derive system test cases for black box testing from simulation data. In the first step of this concept, an environment-sensitive behaviour simulation generates a large quantity of data. Then, the simulation data is pre-filtered in order to identify traffic situations, in which dynamic objects may affect the target vehicle. Upon our knowledge, standalone criticality metrics experience limitations and might not be adequate for a reasonable rating according virtual test and validation of fully automated driving functions and ADAS. Thus, the pre-filtered data are rated by a new factor to extract relevant situations. This new factor can be parametrised by developer specifications or use case specific targets. Then, identified situations are described formally and in an abstract way to build up a situation catalogue. System requirements are linked to the situation catalogue to define test criteria and derive a test suite from evaluated simulation results using a Domain Specific Language. The situation catalogue linked to systems requirements represent the input stimuli for system testing of fully automated vehicles and ADAS. The generated test cases can be used during typical development stages e.g. Software in the Loop (SiL), Hardware in the Loop (HiL), etc. A common simulation environment might be Virtual Test Drive [7].

3 Traffic Conflict Techniques for Fully Automated Driving

Traffic Conflict Techniques (TCT) have come a long way since they were introduced in the late 1950s. Several studies have been conducted to evaluate traffic conflicts and criticality metrics have been developed and extensively discussed since the late 1970s. Amundsen and Hydn [8] defines a conflict as "an observational situation in which two or more road users approach each other in space and time to such an extent that a collision is imminent if their movements remain unchanged". Time-To-Collision (TTC) [9] and Post-Encroachment-Time (PET) [10], Deceleration-To-Safety-Time (DST) [11] and various modifications of TTC and PET like Gap-Time (GT), the Proportion of Stopping Distance (PSD), Time-To-React, Time-To-Maneuver and Initially-Attempt-Post-Encroachment-Time (IAPT) became effective measurements for the rating of traffic conflicts and the development of Collision-Avoidance-System (CAS) and Pre-Crash Systems (PCS). They are also used in the field of accident research. Rodemerk's [12] general criticality criterion represents a collision risk in potential collision areas, using motion prediction models and the knowledge of the course of the roadway. Common to all these metrics, they only calculate, whether a collision or a conflict zone occurs if participating objects do not change their path or speed. A rating of the traffic situation thus, can only be processed if there is an imminent conflict or accident.

Fully automated vehicles and ADAS have to process situation analysis and interpretation. An adequate interpretation is done by taking all relevant dynamic objects into account. Identifying these situations to process situation analysis and interpretation, they have to be detected and rated much earlier than known criticality metrics can provide. So, situations which may look harmless or initially pose no danger might also be interesting for interpretation, due to environment reasons and missing or vague traffic guidance. To be able to develop appropriate strategies for fully automated driving, it is necessary to analyse complex traffic situations at a time, when surrounding objects and their influence to the target vehicle cannot be assessed by conflict and criticality metrics. Analysing apparently uncritical, but complex traffic situations is inevitably to quantify a situation as a whole and further information regarding the environment, objects and traffic regulations have to be taken into account. A factor to calculate the influence of surrounding objects and all relevant attributes of a situation to the target vehicle due to its future actions can be used for a suitable situation interpretation. This is indispensable for fully automated driving and ADAS, especially in urban space. In addition, such a criterion enables new techniques to identify situations in simulation data as well as real traffic data, which will not be detected using well known conflict and criticality metrics. Junghans *et al.* [13] shows methods to detect atypical situations and actions like U-turn, driving wiggling lines and traffic violations. As Detzer *et al.* [14] mentions "atypical situations refer to incident, which differ from the usual case, but most of all present a danger to road users", the situations Junghans *et al.* are able to de-

tect are caused by driving actions and decisions of human drivers, that in fully automated vehicle are not allowed to happen.

4 Concept

To derive test cases for fully automated driving and ADAS, we propose a multi-layer concept as pictured in Figure 1. This concept shows necessary steps from an environment-sensitive behaviour simulation, an extraction of relevant situations through to the derivation of executable test cases. As seen in Figure 1, step 1 shows an environment-sensitive behaviour simulation which generates a large amount of data. In the next step, the generated data are filtered and individual situations are rated to extract relevant situations for fully automated driving and ADAS. Extracted situations are described formally and in a textual, human understandable way, cf. step 3. In the fourth step, a situation catalogue is build up with the help of the textual description. The situations of the catalogue get linked to the formal scenario description. This enables an automated re-simulation of individual situations afterwards. Using the situation catalogue and system requirements, a test catalogue including tags to the formal description of the situation catalogue can be derived as presented in step 5 of Figure 1. The final step of this concept is, extracting executable test cases automatically. Figure 1, step 6 shows a linking to the formal scenario description. Having the tagging of the derived test cases to the formal description, an automated simulation of the test cases can be done. From an automation perspective, Step 1 to 4 can be easily automated with respect to parameter specification (filters, etc.) which need to be performed manually. The automation level of step 5 will depend on the level of formalism used within requirements documents. From a todays practitioners perspective this will be manual work. Step 6 might be automated using various templates. In the following, the steps are described in detail and open challenges are given.

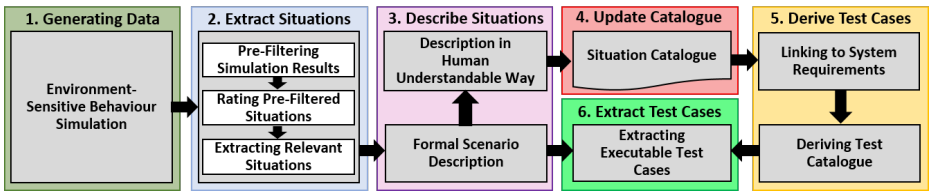

Fig. 1. Concept for deriving test cases from simulation data.

4.1 Environment-Sensitive Behaviour Simulation

The first step of this concept is the generation of simulation data using a probabilistic environment-sensitive behaviour simulation. A belonging scenario description consists of a logical database for the environment description (e.g. roads

with their type and lanes, lane marking, traffic signs, etc.), dynamic elements (e.g. pedestrians, cyclists, vehicles, traffic lights), goals and values for dynamic objects and actions and events (cf. Ulbrich *et al.* [6]). A scenario description can exist in different formats such as XML or HTML files. All participating objects pursue individual goals controlled by behaviour models close to reality. The scenario description, the behaviour models for the dynamic objects and the logical database for the road network represent the input for the simulation and the presented method.

A target vehicle continuously moves along defined routes and further dynamic objects react situationally according to each other and to the target vehicle. Regarding the use case and implemented functions, the subject vehicle is also able to react situational. Thus, this simulation method with realistic, environment-sensitive behaviour models tries to model the complexity of real-world traffic and generates data. These data include probabilistic relevant, not known situations which have to be tested to ensure full and consistent validation of highly connected and automated vehicles. This simulation will be operated continuously and generates data by executing the dynamic elements plans (actions, paths, events, etc.). As the simulation continues, new situations will occur, which do not need to be scripted in advance and might represent a realistic scenario. Thus, randomly generated data might be too extensive to be analysed manually.

It is possible to include a driving simulator to this environment-sensitive behaviour simulation. This enables human interactions while the simulation process. Thus, driving studies in a virtual environment can be done and generated data can be evaluated afterwards. This overcomes limitations of behaviour models and a wider range of variations of included situations can be achieved.

The outcome of the simulation run contains states, positions and circumstances of every dynamic object and element for every frame. Depending on the development process and granularity of the used model and functions for the ego vehicle, the simulation results may also contain sensor views, output of control units or bus messages. This can be achieved by linking for example a HiL simulator. These extracted simulation results then are processed by the following steps of this concept.

Exemplary Situation. In order to acquire a feel for the conceptual approach, we have taken out an exemplary traffic situation in urban space. Throughout, this situation will be picked up in the following to exemplify specific steps. The environment-sensitive behaviour simulation is not restricted to simulate urban traffic scenarios. Also highway traffic can be simulated.

Exemplary Situation: Urban Traffic. We presuppose a X-intersection (cf. Figure 2). The target vehicle (object 1) plans to turn left while the oncoming traffic (object 2) has got a green traffic light signal. Beside that, a pedestrian crossing, also regulated by traffic lights has got a green traffic light signal too. A pedestrian (object 3), located on a pedestrian walk leading to the pedestrian crossing, moves straight forward to the crossing.

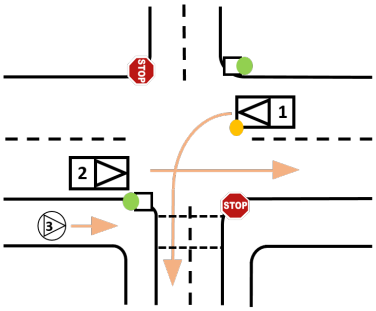

Fig. 2. Exemplary situation which is of importance for situation interpretation and cannot be detected by well known criticality metrics in case of certain circumstances.

Open Challenges. As this environment-sensitive behaviour simulation is able to produce data, there are some open questions to be answered. To get a better representation of reality and improve the following overall process, a verification has to be done, whether an adequate number and variations of scenes were generated. A verification can be done in two ways, via logging and checking afterwards or online analysis. If an online analysis is done, the simulation has to provide editing specific parameters while it is running. To guarantee a wide range of relevant situations in the simulation results, it would be conceivable to include variations during the simulation process. This can be done by automated editing the behaviour models while the simulation runs or define periodic recurring elements on specific points of the environment and different states of the behaviour models. Also a challenging task is the reset and roll back of individual elements during the simulation process, if they run into crashes or deadlocks.

4.2 Extracting Relevant Situations

Pre-Filtering. For fully automated vehicles, relevant situations contain all objects, elements and circumstances that may affect the vehicle in its trajectory planning. As strategy to apply situation analysis and plan future actions, we suggest the concept of manoeuvre spaces. Manoeuvre spaces describe abutting areas divided by stopping lines and logical points, where traffic guidance requires an analysis and interpretation of the situations and, if necessary, an adaptation of the driving actions. Manoeuvre spaces result from existing stopping lines at intersections and lane junctions caused by traffic guidance and logical stopping points resulting from "turning while conditional compatibility". Required information therefore can be extracted from the used logical datasets (e.g. OpenDrive [15], RoadGraph [16]). For an example, we extracted manoeuvre spaces for a simulated complex intersection (cf. Figure 3). The source format was an XODR-file

following the OpenDrive specification. The visualisation was done by using Unity 3D[4].

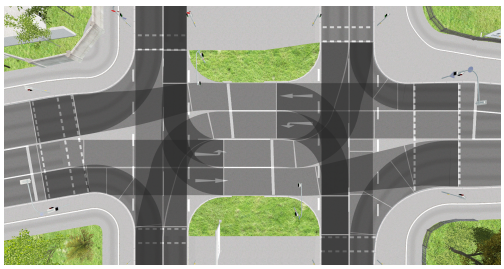

Fig. 3. Extracted manoeuvre spaces (grey surfaces) for a simulated intersection.

After extracting manoeuvre spaces in a defined area or radius around the subject vehicle, the localisation and association to the traffic lanes is necessary. The objective of the localisation and mapping to its corresponding traffic lane is to calculate, which lanes are reachable and following the lane of the target vehicle, if traffic regulations are going to be upheld. This can be expected, because fully automated vehicles have to act rule-consistently. Furthermore, as fully automated vehicles already know their actual route, only traffic lanes along the planned route have to be taken into account.

Simultaneously, all dynamic objects and elements have to be extracted and included. After that, the detected objects will be mapped to their related traffic lane or (traffic) surface, similar to the mapping of the subject vehicle to its traffic lane. Then, for every object an estimation for future trajectories or future occupied areas will be calculated. This is done by using motion prediction models (cf. [18], [19], [20], [21], [22], [23]). Using motion prediction models afterwards, instead of using the implemented behaviour model for simulation, for pre-filtering has got advantages. Motion prediction models give different results than simulation behaviour models. Behaviour models take their surrounding and circumstances into account and plan their actions situationally. Motion prediction models estimate possible trajectories and future occupied areas on the basis of observed behaviour and give multiple estimated actions, assessed by a probability value. Due to use case or developer requirements, motion prediction models should be able to be adapted, e.g. how they take traffic light states and traffic regulations into account.

In conjunction with a defined temporal forecast, one can figure out potential overlapping zones with the computed manoeuvre spaces of the subject car. By having this information, the current scene can be reduced to its relevant objects according to a defined temporal forecast. The filtered situation now rep-

[4] Unity 3D is a game and graphic development platform to build high-quality 2D and 3D games and visualisations [17].

resents the relevant dynamic objects for the target vehicle and in which future manoeuvre it might be influenced by other objects.

The pre-filtering parameters can be edited by the developer in a configuration file. Using this, specific requirements or use cases can be included and different types of situations can be identified. It is possible, taking only pedestrians, other vehicles or cyclists into account or reduce the simulation results due to specific traffic routes or environments like intersections, pedestrian crosswalks, acceleration lanes, traffic light regulations, etc. It is also possible to filter individual situations, e.g. targets vehicles camera field of view. Situations can be found, where e.g. a defined number of pedestrians are in the field of view of the camera or dynamic objects have a defined orientation to the target vehicle.

Rating Traffic Situations. The influence of identified relevant objects to the target vehicle is calculated by extracting the information when overlapping manoeuvre spaces will be reached by the target vehicle. The extracted time will be adapted to the estimated motions of the dynamic objects and a *probability of occupancy* can be calculated. The exact definition and calculation of the probability of occupancy will be part of a future publication. So, every manoeuvre space gets a probability for being occupied by a dynamic object when the target car approaches. A calculated high probability of occupancy for a manoeuvre space at a specific point of time does not imply, that this manoeuvre space was crossed by a dynamic object in the further simulation process. It merely indicates, that this situation would have been of importance for the situation analysis and interpretation at a certain point of time. By having a probability and the extracted time when the target vehicle will reach the manoeuvre space, developers have the possibility to filter the simulation data due to a temporal forecast or a pre-defined probability of occupancy will be exceeded. To pick up the thought of an evaluation framework, the probability of occupancy can be expanded by including worst case assumptions like traffic rule violation and atypical behaviour. Furthermore, the probability of occupancy can be linked or expanded by already known criticality metrics, if a collision course exists by estimated trajectories of the dynamic objects.

Exemplary Situation: Identify the Situation. Known criticality metrics cannot rate the exemplary situation (cf. 4.1, Figure 2), if the target vehicle is standing or driving with a certain speed, because an imminent collision does not exist. This situation, is of importance for situation interpretation, because of various ways to challenge this traffic situation and plan future actions. For example, the target vehicle waits until the oncoming traffic has passed, then continues the planned route with enough speed to pass the pedestrian walk, before the pedestrian reaches the crosswalk. Another possibility is to wait until both objects (2 and 3) have passed. Applying motion prediction models and the concept of manoeuvre spaces, this situation will be identified as relevant. Concerning the used motion prediction model, the estimated trajectory of the pedestrian (object 3) will cross the planned trajectory of the target vehicle (object 1). Also the oncoming traffic

(object 2) will intersect the planned trajectory of the target vehicle. For this situation a high probability of occupancy will be calculated and the situation will be automatically detected.

Executing Relevant Situations. After pre-filtering and rating situations according to the use case or developer specifications, relevant situations have to be extracted. Therefore, the probability of the occupancy (and possibly a linked criticality metric) can be seen as a search criterion, which has to be parametrized by the developer. So, situations which exceed a defined value of probability or criticality can be found and extracted from a huge amount of simulation results. In order to get the outset of the situation, the related scene has to be reduced to its relevant objects and circumstances. Decisions of actions of the participating objects (cf. Ulbrich *et al.* [6]) have to be taken into account.

Open Challenges. As the concepts allows filtering and extracting relevant situations for interpretation, there are still some limitations. If multiple dynamic objects in one situation are viewed on their own and might be rated as "not relevant", certain circumstances of these objects and their combination might be relevant for situation interpretation. Also a specific sequence of events, like atypical behaviour, might become interesting for situation analysis. An open task is to extend the suggested concept, to be able to detect such situations or sequences of events.

4.3 Describing Executed Situations

Formal Situation Description. After the extraction of relevant situations from simulation data, the situations have to be described formally to enable further automated processing like comparison to other situations. A formal description for situations consists of the sum of its elements, its corresponding concretised parameters and the sequence of events. A formal description language may follow a scheme like the one given by Geyer *et al.* in [24].

Exemplary Situation: Formal Description. As an example for a formal description of the exemplary situation we used a XML based format. Because this example should demonstrate how a formal description can be done, the description of the situation is reduced to the key facts:

```
<!DOCTYPE FORMAL DESCRIPTION SITUATION #1>
<infrastructure>
   <trafficlight name="tl1" type="simple"
       pos="45,25,4" state="green"/>
   <trafficlight name="tl2" type="advanced"
       pos="-45,20,4" state=green"/>
   <trafficsign type="stop" pos="45,-25,3"/>
   <trafficsign type="stop" pos="-45,25,3"/>
```

```
</infrastructure>
<dynamic>
   <person name="p1" pos="-45,-30,2" direction="2" speed="1.5"/>
   <car name="car1" pos="50,20,2" direction="3" speed="20"/>
   <car name="car2" pos="-50,-20,2" direction="2" speed="10"/>
</dynamic>
```

Abstract Situation Description To be better understandable by humans, we recommend a textual description generated from the formal scenario description (cf. Figure 1, step 3). An advantage of a textual description of a relevant situation is, that additional information (e.g. obstacle in lane, vehicles in front in the same lane, traffic light states,...) in plain text can be added. This textual description is tagged with the formal scenario description. So, predefined scenario element sets (e.g. traffic jam, cut in object, pedestrian crosswalks) for re-simulation, readable in an easy way for developer, can be executed. Using scenario based development, intended system behaviour for re-simulation and test cases can directly be derived, based on the abstract situation description in combination with the tagged formal scenario description.

Exemplary Situation: Textual Description. After generating the formal description, a textual and human understandable description of the identified situation can be extracted automatically. For an example, a textual description of the exemplary situation may look like the following:

```
TEXTUAL DESCRIPTION SITUATION #1:
Crossing situation with
    2 traffic lights,
    2 road signs,
    2 cars,
    1 person.
Lane 1 consists of
    2 traffic lights at (45,25,4|-45,20,4)
        with the states (green|green).
    2 cars at (50,20,2|-50,-20,2)
        with direction (4|2+3) and speed (20|10).
Lane 2 consists of
    2 road signs (stop|stop) at (45,-25,3|-45,25,3).
A person at (-45,-30,2)
    with direction (2) and speed (1.5)
    is crossing lane 2.
```

Beside extracting the formal and abstract description out of the simulation results, it is possible to generate an illustration of the identified situation (cf. Figure 4 for our exemplary situation). Therefore, the logical database is used to generate the infrastructure representation. Dynamic objects and elements, including their positions, then can be embedded.

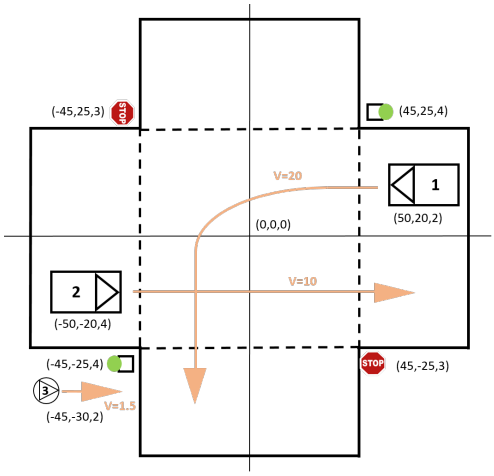

Fig. 4. Generated illustration of the exemplary situation using the logical database and the formal description.

Situation Catalogue. Using the abstract, textual and human understandable description, a situation catalogue can be build up (cf. Figure 1, step 4). As the situation catalogue should be dynamically extendible, new situations have to be compared with the existing set of situations. For this purpose, the tagged formal description can be used for automated comparison and evaluation of the catalogue. We also suggest a classification of the collected situations by driving actions, developer specific demands or use cases. Uncertainty in classifying a situation can be handled by identifying corresponding elements, parameters and events of the formal description. The built up situation catalogue claims what a system for automated driving has to manage.

Open Challenges. Establishing formal and abstract description languages and building up a suitable situation catalogue implies some open and challenging tasks. A situation can be described by ambiguous possibilities and in different ways. A sufficient large description language has to be found to display all necessary information. On the contrary, a description language has to be of manageable scale and variety, to assure efficient search, classification and evaluation of situations. In consequence, an appropriate selection of the abstraction level and choice of relevant elements and parameters is inevitable.

4.4 Build Up Test Catalogue

Linking To System Requirements. To maintain a certain degree of traceability, links between each situation listed in the situation catalogue and the related system requirements have to be created and documented. This enables the user to view the requirements which are relevant for a specific situation, as

well as to filter all situations covering a particular set of requirements. Additionally, situations or requirements with no established links can be identified to guarantee the integrity of the system. Textual requirements are either written in a natural language or in a formal way. Especially in the automotive domain, a natural language is often chosen as primary solution, because the specifier can stick to familiar descriptions and phrases. Formal styles are much more laborious to get used to. A link between the requirements and the situation catalogue can be established in form of a requirement identifier reference embedded in the situation description and the tool-supported tracing of these connections.

Exemplary Situation: Requirement. System requirements to handle the exemplary situation may have the following form:

```
REQUIREMENTS FOR SITUATION #1:
#1: If Ego turns left, the system has to give way to
    oncoming traffic.
#2: If Ego turns into a lane, the system has to give
    way to crossing pedestrians and cyclists.
#3: In give way situations, the system has to stop in front of
    relevant conflict zones until a safe passing is possible.
```

Deriving Test Cases. After establishing the links, test cases have to be created to be able to test the system. This can be done by hand, which requires the test engineer to review and understand the requirements and the situations, which is prone to errors and misinterpretations. A better solution is to partly automatize the test case creation. In our case, both the requirements and the situation catalogue are textual, which advices a textual generation technique.

A *Domain-Specific Language (DSL)* is a programming language limited to a specific domain and capable of automatically generating diverse textual and graphical artefacts (cf. Figure 5). Such artefacts include, for example, diagrams, models and even source code in different general purpose languages (e.g. C++). Although DSLs can be graphical as well, the textual nature of our source documents leads to a textual DSL as optimal solution.

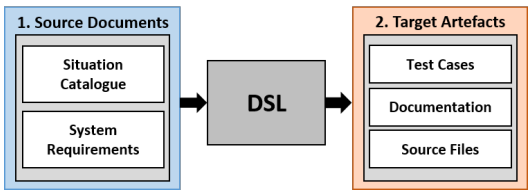

Fig. 5. DSL workflow for test case generation.

This textual DSL can directly use syntax and semantics of the situation catalogue and the system requirements. An automatic interpretation of both

documents is possible, although it might be challenging. The feasibility of this automatism has to be examined in detail and will be part of a future publication. The advantage of this approach is the maintenance of the readability for humans and the usage of already specified patterns. The DSL then aggregates all relevant information out of the situations and requirements and generates predefined artefacts. The main type of artefact in our case are the test cases extracted out of the situations, which then can be used in manual and automated system testing. Additionally, the test cases and the related results can be reintegrated, tagged and included in other environment-sensitive behaviour simulations. This can be done automatically, due to the fact that the derived test cases from the situation catalogue are tagged with a formal description.

Exemplary Situation: Executable Test Case. Using the extracted descriptions and the system requirement, test cases for all relevant objects of the situation can be generated automatically. The range of the parameters can differ depending on predefined legal or technical constraints. For our exemplary situation, the test case may have the following form and range of parameters:

```
TEST CASE #1, SITUATION #1:
RANGE OF PARAMS:
     p1: speed=[0;8]
   car1: speed=[0;30]
   car2: speed=[0;50]
EXPECTED SYSTEM BEHAVIOUR:
   The system should turn left on lane 2.
TEST CRITERIA:
   Does the target vehicle give way to the vehicle and pedestrian?
```

Open Challenges. Our presented concept derives test cases, but no real test oracle. The comparison of the system behaviour to the extracted test criteria can be potentially automated, but will not be considered here. A further issue that needs to be solved, is how an automated valuation method can be set up. Up today a human test expert has to rate every situation. A first implementation of a test oracle might be to define a passed test case as preventing a collision with other objects.

5 Conclusion and Future Work

We discussed well known criticality metrics regarding their usability in the field of rating traffic situations and pointed out, why their use for fully automated vehicles and ADAS is not sufficient. Fully automated vehicles and ADAS have to analyse and interpret traffic situations at every point of time. Thus, relevant situations have to be identified to ensure full testing and validation. Virtual simulation methods are getting more important and are producing a large quantity of data. The challenge thereby is to be able to execute relevant situations and

derive test cases. As a solution, we suggest a multilayer model concept to filter simulation data, rate relevant situations, transfer them to a situation catalogue and derive executable test cases. Using DSLs, situations can be presented in a formal and textual, human understandable way and linked to system requirements. This enables a throughout automation for deriving test cases from simulation data. To demonstrate the benefit of this approach, an exemplary situation was taken out and picked up consistently to exemplify specific steps of this concept. Figure 1 presents a good overview to our approach.

Future work will contain an exact definition of the probability of occupancy and its parametrization. Further parameters like traffic rule violation and atypical behaviour are going to be included, to cover more specific developer requirements and get a more detailed rating of the situation executing framework. Beside that, we are working on an appropriate formal and abstract description capable of building up the situation catalogue and using a DSL to automatically link the requirements and derive executable test cases.

References

1. Eo, J.S., Choi, H.R., Gao, R., Lee, S., Wong, W.E.: Case Study of Requirements-Based Test Case Generation on an Automotive Domain. In: 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion, pp. 210-215. IEEE Press (2015)
2. Zofka, R.M., Kuhnt, F., Kohlhaas, R., *et al.* : Data-Driven Simulation and Parametrization of Traffic Scenarios for the Development of Advanced Driver Assistance Systems. In: 18th International Conference on Information Fusion, pp. 1422-1428. IEEE Press, Washington DC (2015)
3. Schuldt, F., Menzel, T.: Eine Methode für die Zuordnung von Testfällen für automatisierte Fahrfunktionen auf X-in-the-Loop Verfahren im modularen virtuellen Testbaukasten. In: 10. Uni-DAS e.V. Workshop Fahrerassistenzsysteme 2015, pp. 1-12. Uni-DAS e.V., Walting (2015)
4. Schuldt, F., Lichte, B., Maurer, M., Scholz, S.: Systematische Auswertung von Testfällen für Fahrfunktionen im modularen virtuellen Testbaukasten. In: 9. Uni-DAS e.V. Workshop Fahrerassistenzsysteme 2014, pp. 169-179. Uni-DAS e.V., Walting (2014)
5. Stellet, J., Zöllner, J.M., Schumacher, J., *et al.* : Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 1455-1462. IEEE Press (2015)
6. Ulbrich, S., Menzel, T., Reschka, A., *et al.* : Defining and Substantiating the Terms Scene , Situation , and Scenario for Automated Driving. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 982-988. IEEE Press, Las Palmas (2015)
7. Vires Simulationstechnologie GmbH: Virtual Test Drive User Manual, `http://https://www.vires.com/docs/VIRES\_VTD\_Overview\_201403.pdf` (2014)
8. Amundsen, F., Hydn, C. (Eds.): Proceedings of the first workshop on traffic conflicts. Institute of Transport Economics Oslo and LTH Lund (1977)
9. Hayward, J.C.: Near miss determination through use of a scale of danger. In: Highway Research Record, vol. 384, pp. 24-34. The Pennsylvania State University, Pennsylvania (1972)

10. Allen, B.L., Shin, B.T., Cooper, P.J.: Analysis of Traffic Conflict Collisions. In: Transportation Research Record 667, pp. 67-74. National Research Council, Washington D.C. (1978)
11. Hupfer, C.: Deceleration to safety time (DST)- a useful figure to evaluate traffic safety. In: ICTCT Conference Proceedings of Seminar 3, Department of Traffic Planning and Engineering, Lund (1997)
12. Rodemerk, C., Habenicht, S., Weitzel, A., *et al.* : Development of a general criticality criterion for the risk estimation of driving situations and its application to a maneuver-based lane change assistance system Claas. In: IV. IEEE Intelligent Vehicles Symposium, pp. 264-269. IEEE Press, Alcala de Henares (2012)
13. Junghans, M., Saul, H.: Chances for the evaluation of the traffic safety risk at intersections by novel methods. In: VII. Russisch-Deutsche Konferenz für Verkehrssicherheit, pp. 60-67. Sankt Petersburg (2014)
14. Detzer, S., Junghans, M., Kozempel, K., Saul, H.: Analysis of traffic safety for cyclists - an automatic detection of critical traffic situations of cyclists. In: 20th International Conference on Urban Transport and the built Environment, pp. 491-503. WIT Press, Portugal (2014)
15. Dupuis, M., et al.: OpenDRIVE Format Specification, Rev. 1.4. (2015)
16. Knaup, J., Homeier, K.: RoadGraph - Graph based environmental modelling and function independent situation analysis for driver assistance systems. In: 13th International IEEE Annual Conference on Intelligent Transportation Systems, pp. 428-432. IEEE Press, Madeira Island (2010)
17. Unity Technologies Website, `http://unity3d.com/`
18. Bonnin, S., Weisswange, T.H., Kummert, F., Schmuedderich, J.: Pedestrian Crossing Prediction using Multiple Context-based Models. In: 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), pp. 378-385. IEEE Press, Qingdao (2014)
19. Meyer-Delius, D., Sturm, J., Burgard, W.: Regression-based online situation recognition for vehicular traffic scenarios. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, pp. 1711-1716. IEEE Press, St. Louis, MO, USA (2009)
20. Schneider, N., Gavrila, D.M.: Pedestrian path prediction with recursive Bayesian filters: A comparative study. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 8142 LNCS, pp. 174-183 (2013)
21. Ziebart, B., Ratliff, N., Gallagher, G., Peterson, K.: Planning-based Prediction for Pedestrians. In: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems, pp. 3931-3936. IEEE Press, St. Louis, MO, USA (2009)
22. Rehder, E., Kl, H., Stiller, C.: Planungsbasierte Fußgängerprädiktion. In: 10. Uni-DAS e.V. Workshop Fahrerassistenzsysteme, pp. 129-138., Uni-DAS e.V., Walting (2015)
23. Quintero, R., Parra, I., Llorca, D.F., Sotelo, M.A.: Pedestrian Path Prediction based on Body Language and Action Classification. In: 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), pp. 679-684. IEEE Press, Qingdao (2014)
24. Geyer, S., Baltzer, M., Franz, B., *et al.* : Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. IET Intell. Transp. Syst. 8, pp. 183-189. (2014)