2024 IEEE/SICE International Symposium on System Integration (SII)
January 8-11, 2024. Ha Long, Vietnam

# Advancing MBSE for ADAS/AD: Automated Scenario Generation

Serdar Kınay[1], Ufuk Bolat[1], Buse Yakın Gökdemir[1], Kaan Babacan[1], Namık Zengin[1] and Erhan Özkaya[1]

2024 IEEE/SICE International Symposium on System Integration (SII) | 979-8-3503-1207-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/SII58957.2024.10417219

*Abstract*— **This paper explores the integration of model-based systems engineering (MBSE) and scenario-based engineering (SE) approaches in the development of advanced driver assistance systems (ADAS) and automated driving (AD). The increasing complexity of automotive systems necessitates efficient system design and validation processes. MBSE provides a framework for capturing system functionality and managing complexity, while SE focuses on testing and analyzing system features across various scenarios. By combining these approaches, system development efforts can be streamlined, leading to more efficient project management. This paper reviews existing literature on MBSE and SE in the context of ADAS development and highlights the potential benefits of their integration. An MBSE workflow is introduced where the key focus is on the behavioral aspects of the system. Additionally, a scenario generation pipeline which constitutes scenario databases is proposed to ensure comprehensive testing of system requirements. Results show that with automated scenario generation, the target vehicle cut-out scenario is successfully realized and simulated based on the formally described system behavior as part of the MBSE model.**

## I. INTRODUCTION

The advancements in the automotive industry are accelerating the emergence of increasingly safer and smarter vehicle technologies coupled with the progress of technology, day by day. While promising a safer and more comfortable driving experience, systems of this nature encounter fundamental challenges in the development process due to the convergence of various disciplines and the consequent need for optimal system design. Hence, this necessitates the utilization of the V-cycle in project management processes, along with the imperative for the effective implementation of systems engineering and validation as well as verification approaches [1].

According to The International Council on Systems Engineering (INCOSE), systems engineering is an interdisciplinary approach and means to enable the realization of successful systems [2]. This approach focuses on defining customer needs and required functionality early in the development cycle. It involves documenting requirements, followed by design synthesis and system validation, while considering the complete range of factors: operations, cost and schedule, performance, training and support, testing, manufacturing, and disposal [3]. Automotive original equipment manufacturers (OEMs) are facing a rapid increase in system complexity due to various factors. As an illustrative example, a standard luxury car may include numerous electrical control units (ECUs) responsible for processing over a hundred million lines of code. These types of complexities can lead to a considerable number of requirements, often reaching into the millions, with intricate interconnections numbering in the hundreds of thousands. Such challenges can strain an organization's ability to efficiently oversee the product development process [4].

In the literature, both traditional and model-based systems engineering (MBSE) approaches have been applied to Advanced Driver Assistance Systems (ADAS) [5], [6], [7]. Kaiser and colleagues developed an MBSE workflow for the Advanced Emergency Braking System (AEBS). This workflow involves accelerated design stages with early validation through simulation and safety analysis. The process refines the system architecture iteratively from functional to physical stages, aligning requirements and architecture. The structure encompasses static safety, reliability, and cybersecurity analyses, simulations at varying fidelity levels, and multidisciplinary optimization for component selection [7]. The motivation behind Kaiser's study stems from implementing the proposed MBSE workflow for the AEBS function.

Following the definition of requirements and use cases, the progression moves towards the validation and verification phase as known in the V-cycle framework. Validation and verification of automotive systems encompass testing and analyzing system features across various scenarios to ensure adherence to specified functional and non-functional requirements. Verification entails checking the dynamic behavior of a system under test against explicitly stated requirements. Conversely, validation confirms whether the system has been developed to fulfill its intended functionality [8]. Despite efforts to capture system functionality through explicit requirements, a gap persists between intention and execution, particularly in the intricate features of contemporary automotive systems [9].

As part of the validation process, the growing complexity in the development and testing processes of autonomous driving systems has led to the emergence of scenario-based engineering (SE) needs [10]. To reiterate previous remarks, MBSE is also gaining significance in the development of automotive applications [11]. MBSE addresses complexity control, traceability and enhanced communication across interdisciplinary areas. Various methodologies have been employed in MBSE applications, as they facilitate system development in multidisciplinary domains [12]. Additionally, studies have compared these MBSE methodologies [13]. Utilizing both MBSE and scenario-based testing approaches in V-cycle applications can potentially streamline system development efforts and enable more efficient project management. This further reinforces the idea that the possibility of combining MBSE methods and scenario-based approaches

[1]AVL Research and Engineering, Istanbul, Turkey namik.zengin@avl.com

in system development is feasible.

While the number of studies in the literature in this field are small, some stand out. For example, Sippl and colleagues [14] elucidate the impact of scenarios and generated artifacts on systems engineering. They achieve this by creating machine-readable scenario descriptions in OpenSCENARIO and OpenDRIVE [15] formats. Furthermore, Kremer et al. [16] employ use case diagrams and scenario allocation diagrams to establish associations with actors and the environment. Another study develops, specific MBSE diagrams for Operational Design Domain (ODD) and scenarios, conducting Safety of the Intended Functionality (SOTIF) analysis using stereotypes and SysML relationships for the Multi Story Car Park Chauffeur function [17].

The motivation of this study is to establish an integrated workflow for applying both MBSE and SE methods in the development processes of ADAS or AD domains. In this study, for the selected ADAS function Adaptive Cruise Control (ACC), the systems engineering phases were initially carried out using MBSE, and subsequently integrated with the structure obtained through SE. Proposed methodology distinguishes itself from the existing studies by putting a greater emphasis on automatizing the machine-readable scenario file generation based on the MBSE model. This greatly reduces the workload necessary for the alignment of the system engineer and the scenario engineer owing to it's holistic approach for the entire process.

The study's outline is as follows: Section II addresses the problem definition. Subsequently, the utilized MBSE methodology is explained in Section III. Section IV elaborates on automating functional scenario generation and the example simulation result. The research culminates with the Section V, presenting the study's conclusions.

## II. Problem Definition

In this section, we delve into the problem definition and outline the steps of the solution process. The workflow pertaining to the identified problem is visually depicted in Figure 1, providing an overview of the process. The process

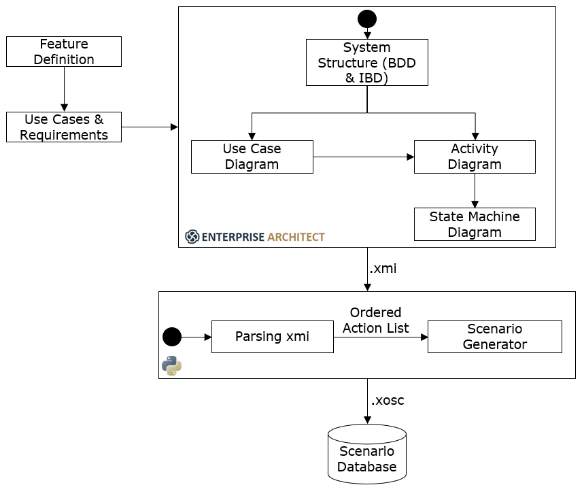

Fig. 1. Workflow of the project

begins with the selection of the ADAS feature ACC as the focus. In the first step, the workflow is implemented using this chosen feature. Moving on to the second step, use case and requirement definitions are established for the ACC function. This step serves as the starting point for creating MBSE diagrams. The figure illustrates that system structure, use case, activity, and state machine diagrams are generated within the Enterprise Architect environment [18], based on the previously defined use cases and requirements. After the MBSE diagrams are created, the structure is exported from Enterprise Architect in .xmi format and then imported into the Python environment. Subsequently, the xmi structure is parsed, and all actions are compiled, marking the commencement of the scenario generation phase. The generated scenarios, formatted in OpenSCENARIO format, are collected to form a scenario database. The primary objective of this study is to automate this intricate process, enhancing the efficiency of managing MBSE, validation, and verification processes. This automation is achieved through the use of the designated OpenSCENARIO player.

## III. MBSE Methodology

### A. *Overview of the Method*

The MBSE workflow which was used in this study is shown Fig. 2. At the ADAS feature level, there are four main stages. Firstly, multiple sources are utilized to define requirements. Stakeholder needs provide a general framework along with regulations, expert knowledge, and ODD specifications to finalize all requirements. A requirements diagram is employed to establish connections between these requirements and associated blocks.

Secondly, for structural analysis, Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD) are utilized to model the system context and logical architecture. This phase involves building relationships with various control modules and ensuring traceability of signals. Concurrently, behavioral analysis is conducted. Use case diagrams, activity diagrams and state machine diagrams are parts of behavioral analysis to show that how the ADAS function works. SysML relationships, such as 'satisfy' and 'trace' are employed to establish traceability between requirements and use cases. State machine diagrams portray function states, while relevant use case scenarios are interconnected. Following these analyses, the verification and validation stage involves linking the ADAS scenario database with the associated requirements.

### B. *MBSE Diagrams*

A system is represented through both behavioral and structural forms within the context of MBSE approach. There are various diagrams within the MBSE tools to serve this purpose. While BDD and IBD are used to represent the structure, the Use Case (UC) and the Activity Diagrams are instrumental in portraying the system's behavior.

BDDs are utilized to illustrate a system's logical and/or physical architecture. In this study, the system structure is designed such that the 'System Context' is identified as the

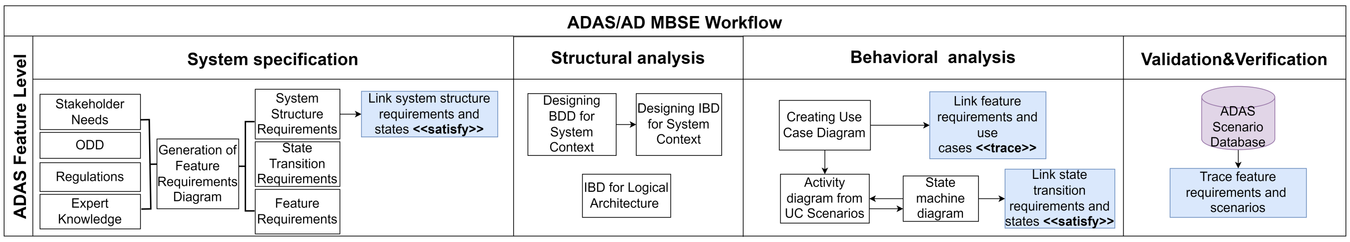
ADAS/AD MBSE Workflow
ADAS Feature Level
System specification
Structural analysis
Behavioral analysis
Validation&Verification
Stakeholder Needs
ODD
Regulations
Expert Knowledge
Generation of Feature Requirements Diagram
System Structure Requirements
State Transition Requirements
Feature Requirements
Link system structure requirements and states <<satisfy>>
Designing BDD for System Context
Designing IBD for System Context
IBD for Logical Architecture
Creating Use Case Diagram
Link feature requirements and use cases <<trace>>
Activity diagram from UC Scenarios
State machine diagram
Link state transition requirements and states <<satisfy>>
ADAS Scenario Database
Trace feature requirements and scenarios

Fig. 2. General view of MBSE workflow

primary block, encompassing three part properties: ’Feature’, ’Environment’, and ’Other Systems’.

A UC Diagram serves the function of visually delineating the functionalities offered by a system. It achieves this by depicting actors, their objectives (presented as use cases), and any interconnections among these use cases. Essentially, a UC Diagram showcases how a system is employed. The links between actors and use cases signify the interactions taking place between actors and system components to accomplish tasks associated with the use cases. In this study, a base use case named ”DDT” (Dynamic Driving Task) is defined along with the two extending use cases, ’Cruise’ and ’Follow’.

In an MBSE approach, State Diagrams serve to exhibit the feasible states a feature may transition into, as well as the ensuing transitions. Transitions are triggered based on the conditions of state transition, facilitated by signals flowing through connections between states. In this work, the activity diagrams for ’Follow Mode’ and ’Cruise Mode’ are configured as the sub-states of the active state.

Activity Diagrams, designed to provide intricate details about use cases and depict the sequence of events transpiring within a use case as a flow of events, are harnessed. Given the design of the feature in this study, two primary modes, ’Follow’ and ’Cruise’ are defined. Activity diagrams are employed to illustrate the behaviors of these modes. When considering ’Follow Mode’ as an instance, its activity diagram employs three swimlanes representing ’Environment,’ ’Feature,’ and ’Other Systems’ which are the three factors influencing the activity’s progression defined within themselves. Throughout the activity, actions are sequenced and interconnected via object flows, control flows and input/output pins, through which information or logic passes through. Initially, the feature derives information from the environment and assesses it. The evaluation process is modeled as an activity instead of an action, so that it can have multiple actions to represent the steps of the evaluation process embedded. Following evaluation, decision nodes are applied to segregate branches based on the assessment outcome. If it is decided that there is no target object, a signal will be sent (“FollowToCruise”) to change the feature mode by initiating the cruise mode transition. Conversely, if a target object is detected, the feature engages in various tasks such as maintaining the following distance and disseminating HMI information. Simultaneously, related tasks are executed by other systems as shown in Fig. 3.

In the environment information evaluation process which

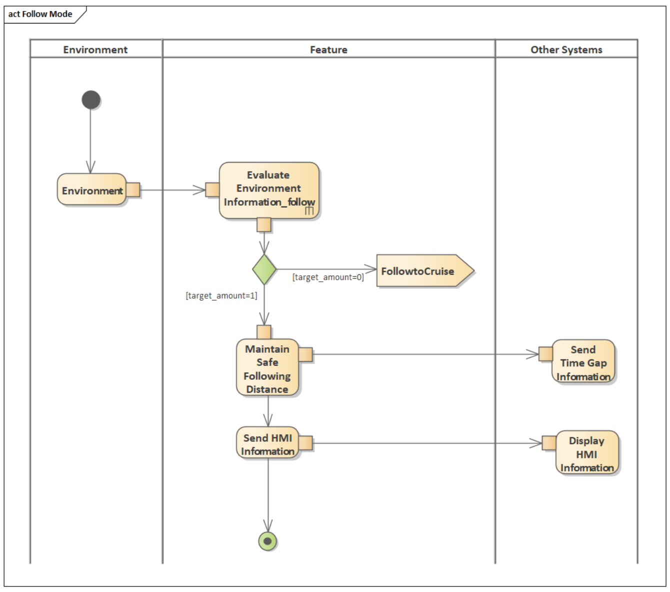

Fig. 3. General view of the Activity Diagram for Follow Mode

is shown in Fig. 4, the action will be determined with respect to the target object’s action. If the target object is within the specified detection range, it will be identified as a follow-worthy target. Accordingly, the value specification will be *1*. If the target object cuts or gets out of specified range by accelerating or moving beyond during the action, it can be inferred that there is no object to follow within the range and the assigned value to the value specification variable will be set to *0*. The value allocated to value specification variable along the followed path will be supplied to the decision-making node and assessed as the target value.

## IV. Automatic Scenario Generation

This chapter describes the process of parsing SysML diagrams created using the scenario-based approach after they have been exported in .xmi format. It outlines the rationale for using the ordered list of actions obtained from the parsing process in the creation of scenarios. Detailed explanation of different scenario types utilized throughout the process is also included.

### A. XMI Parsing

The resultant .xmi file encompasses information about every SysML diagram present in the project file. However, in this study, owing to the scenario-based approach, the

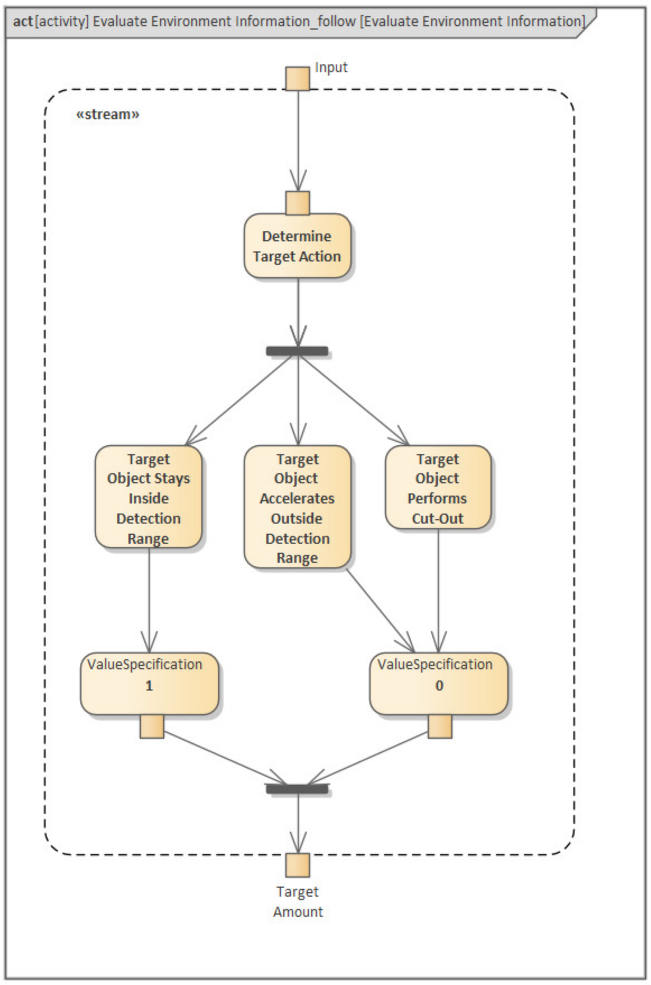
act [activity] Evaluate Environment Information_follow [Evaluate Environment Information]
Input
«stream»
Determine Target Action
Target Object Stays Inside Detection Range
Target Object Accelerates Outside Detection Range
Target Object Performs Cut-Out
ValueSpecification 1
ValueSpecification 0
Target Amount

Fig. 4. The view of the Evaluation activity

primary emphasis on parsing activity diagrams to extract ordered action lists. Various pieces of information such as the name, id, and type of each SysML element within the activity diagram are garnered through the parsing process. The critical aspect is to establish the relationships between SysML elements in order to obtain the ordered action list. For each activity diagram, the *Initial Node* is designated as the starting point. Then, for each element, the other elements to which they are linked are identified one by one, resulting in an ordered list of SysML elements. Ultimately, the ordered list of SysML elements is integrated with the sub-activity diagram. This methodology has been applied to the activity diagrams described in the previous section, as well as other activity diagrams prepared within the scope of the project. The resulting ordered action list is shown in Fig. 5.

## B. Scenario Generation

With the ordered action list, the next objective is to leverage this knowledge to construct a scenario database which will ensure that each state transition requirement and by extension, each use case, activity diagram and action is tested appropriately. Since each activity diagram fundamentally encompasses actions that either trigger state transitions or sustain the current feature state, a template which can provide a flow of actions combined with their corresponding state transitions will make it possible to achieve a rule-based full factorial scenario generation pipeline. This process is developed using the Python programming language where scenario databases are stored within Pandas database structures and can be conveniently exported as CSV files for further applications. In the final stage, OpenSCENARIO and OpenDRIVE files can be exported in their respective formats.

The ISO 34501 [19] standard provides a guideline for categorization of ADAS/AD scenarios based on their abstraction level. Adhering to this scenario classification forms the initial stride toward ensuring a robust scenario generation process that remains reusable and comprehensible across diverse projects and among various stakeholders at all levels.

The foundational phase of scenarios is functional scenarios which has the highest abstraction level where the actors (i.e. road users in general such as vehicles, buses, motorcyclists and pedestrians), their behaviors as well as the environment information is described in a non-formal, human readable manner with sentences, often complemented by sketches visualizing the scenario. In the proposed process, this step is communicated to the stakeholders through the activity diagrams and use cases generated in the preceding steps.

Subsequently, abstract scenarios are crafted. These are formalized descriptions of functional scenarios where the scenario is machine readable but still described at a high level with the combination of static and dynamic elements. Static elements can be derived from the ODD of the feature encompassing domains in which the feature should operate without issues. Essential ODD elements to make up the static parts of the abstract scenarios are the drivable area type (i.e. motorway, intersection), number of lanes, curvature of the road as well as the speed limit. The formulation of dynamic elements demands a more intricate approach. Another relevant standard which can guide this process is ISO 34502 [20] which includes a formal approach for traffic related critical scenario classification. While creating strictly safety critical scenarios is not in the scope of this work, the approach is suitable. During simulations, the subject vehicle is expected to be controlled through a controller in a software-in-the-loop (SiL) manner, obviating the need to define the subject vehicle's behavior in the created scenarios. Consecutively, the amount, relative location and the behavior of the surrounding vehicle will constitute the dynamic element of the defined scenarios. The amount is determined based on how many target/surrounding vehicles are required for a parsed action to occur. Target locations are formalized via a grid approach where grid names around the subject vehicle denote its lane with respect to the subject vehicle. Moreover, target behaviors can entail lane changes in left or right directions or keeping a lane while accelerating or decelerating. Finally, rules are applied to all of the combinations, ensuring that predefined state transitions and actions correspond to the accurate combination of target vehicle location and behavior.

| State Control | Action 0 | Action 1 | Action 2 | Action 3 | Action 4 | Action 5 |
|---|---|---|---|---|---|---|
| Cruise - Follow | Environment | Evaluate Environment Information_cruise | Determine Target Action | Target Object Performs Cut-in | SendSignalAction | -- |
| Cruise - Follow | Environment | Evaluate Environment Information_cruise | Determine Target Action | Target Object Appears in Front | SendSignalAction | -- |
| Cruise - Cruise | Environment | Evaluate Environment Information_cruise | Determine Target Action | No Target Object | Adapt Set Speed | Send HMI Information |
| Follow - Cruise | Environment | Evaluate Environment Information_follow | Determine Target Action | Target Object Accelerates Outside Detection Range | SendSignalAction | -- |
| Follow - Cruise | Environment | Evaluate Environment Information_follow | Determine Target Action | Target Object Performs Cut-out | SendSignalAction | -- |
| Follow - Follow | Environment | Evaluate Environment Information_follow | Determine Target Action | Target Object Stays Inside Detection Range | Maintain Safe Following Distance | Send HMI Information |

Fig. 5. Ordered Action List

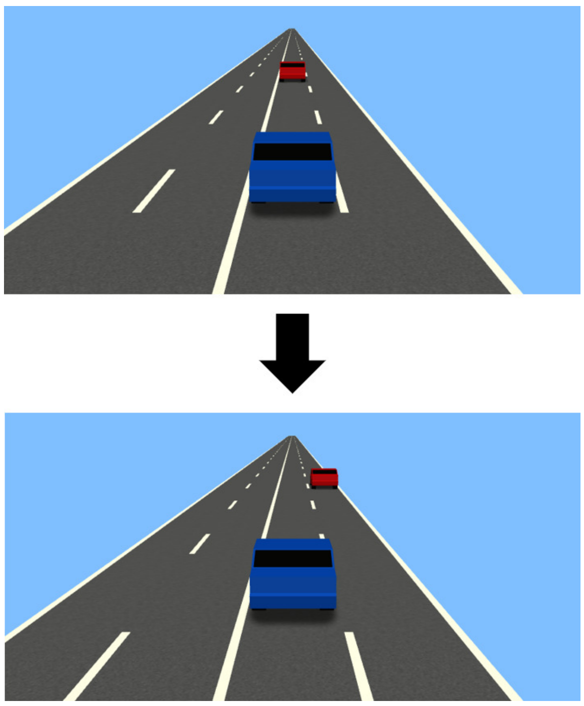

Fig. 6. Example cut-out scenario as seen in esmini scenario player

Logical scenarios encapsulate the value ranges applicable to scenario parameters. These ranges can be given as uniform distributions or normal distributions. The lower and upper limits within each range can be deduced through expert knowledge or derived from naturalistic driving data. As earlier mentioned, the focal point of this study is the assessment of the ACC feature, where the subject vehicle engages in longitudinal movement. This combined with lateral movement will form the basis of an Automated Lane Keeping System (ALKS). The regulation for ALKS [21] furnishes guidelines for devising critical scenarios, alongside the requisite parameter ranges employed during the creation of logical scenarios. Following this, the parameter ranges defined can be sampled to create specific concrete scenarios which describe a unique scenario with exactly one specific scenery and set of events with fixed parameters. It is crucial to note that this sampling process adheres to a full-factorial approach, distinguishing itself from optimization, combinatorial, or search-based methodologies.

The final step is the conversion of concrete scenarios to executable and testable OpenDRIVE and OpenSCENARIO file formats. First, a straightforward OpenDRIVE file based on the described ODD is crafted. Here, the lengths of roadways are computed based on the actors' velocities and the duration required for their individual actions. The OpenSCENARIO segment commences with the initialization of the actor coordinates on the test track founded on the previously defined locations of surrounding vehicles. A story and an act is then added which contain a maneuver group for each actor. A maneuver group is a pair of actor(s) and a maneuver, where the maneuver contains an event which commences execution when the start trigger condition is valid. An action is also defined which describes the movement that will be executed by the surrounding vehicle. Based on the parameters and behaviors defined in the concrete scenarios, the events and the triggering conditions are created. The illustrative outcome as seen in Fig. 6 substantiates the success of the scenario generation process, with an example target cut-out scenario visualized using the esmini scenario player, where the ego vehicle is in color blue and the target vehicle in color red. The maneuver groups for both entities are initialized at the very beginning of the scenario. The initial action for teleportation of entities is also conducted appropriately based on the grid approach discussed earlier. After 5 seconds passes, the trigger for the target cut-out action occurs, when the target vehicle initiates a lane change maneuver with lateral speed as specified in the concrete scenario, resulting in the ego vehicle switching from follow mode to cruise mode state, in line with what was described earlier in the MBSE diagrams section.

## V. CONCLUSIONS

As the autonomy levels of ADAS/AD features advance, the conventional systems engineering approach which is document-based and hard to keep traceability of, is paving the way into the more well-rounded and elegant approach of MBSE. Although MBSE languages such as SysML establish a foundation for generating essential work products, substantial interpretational gaps persist, particularly within the ADAS/AD domain. Moreover, many current methodologies neglect to holistically consider the crucial and increasingly valuable process of ADAS/AD scenario engineering through MBSE.

The proposed approach not only furnishes a template to guide the construction of an MBSE model, focusing on the state machine and associated activity diagrams, but also introduces a rule-based framework enabling the automatic generation of scenarios based on the established model.

In the future, the scenario database will be further incorporated into the MBSE model, aligning each concrete scenario with traceable requirements, thereby facilitating extended validation and verification of these requirements. The generated executable scenario files will undergo meticulous examination using criticality metrics and key performance indicators (KPIs) to ensure the robustness of scenario creation. Additionally, the scope of the feature will be expanded to cover lateral movement and other intricate event detection and response scenarios.

## References

[1] S. J. Kapurch, *NASA Systems Engineering Handbook*. Diane Publishing, 2010.

[2] T. M. Shortell, "Incose Systems Engineering Handbook: A guide for system life cycle processes and activities," *John Wiley& Sons,*, 2015.

[3] C. Haskins, K. Forsberg, and M. Krueger, "Systems Engineering Handbook: A guide for system life cycle processes and activities." Incose San Diego, CA (US), 2011.

[4] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *2017 IEEE International conference on systems, man, and cybernetics (SMC)*. IEEE, 2017, pp. 2075–2080.

[5] E. Stoddart, S. Chebolu, and S. Midlam-Mohler, "System engineering of an advanced driver assistance system," SAE Technical Paper, Tech. Rep., 2019.

[6] S. K. Yetkin, M. Eren, N. Zengin, E. A. Rencüzoğullan, M. Tomruk, and H. Yılmaz, "V2X communication based system development: Application on intersection assist with co-simulation," in *2022 IEEE 21st International Ccnference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE, 2022, pp. 138–145.

[7] B. Kaiser, B. Dion, I. Tolchinsky, T. Le Sergent, and M. Najork, "An AEBS use case for model-based system design integrating safety analyses and simulation," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2022, pp. 3–20.

[8] A. Terry Bahill and S. J. Henderson, "Requirements development, verification, and validation exhibited in famous failures," *Systems Engineering*, vol. 8, no. 1, pp. 1–14, 2005.

[9] J. D'Ambrosio, A. Adiththan, E. Ordoukhanian, P. Peranandam, S. Ramesh, A. M. Madni, and P. Sundaram, "An MBSE approach for development of resilient automated automotive systems," *Systems*, vol. 7, no. 1, p. 1, 2019.

[10] A. Audi and A. Volkswagen. The PEGASUS Method. https://www.pegasusprojekt.de/en/pegasus-method. Accessed: 2023-08-01.

[11] S. Kriebel, J. Richenhagen, C. Granrath, and C. Kugler, "Systems engineering with SysML the path to the future?" *MTZ worldwide*, vol. 79, no. 5, pp. 44–47, 2018.

[12] J. A. Estefan *et al.*, "Survey of model-based systems engineering (MBSE) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[13] T. Weilkiens, A. Scheithauer, M. Di Maio, and N. Klusmann, "Evaluating and comparing MBSE methodologies for practitioners," in *2016 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016, pp. 1–8.

[14] C. Sippl, F. Bock, C. Lauer, A. Heinz, T. Neumayer, and R. German, "Scenario-based systems engineering: An approach towards automated driving function development," in *2019 IEEE International Systems Conference (SysCon)*. IEEE, 2019, pp. 1–8.

[15] ASAM. Asam openscenario. https://www.asam.net/standards/detail/openscenario/. Online; accessed: 01.10.2022.

[16] M. Kremer, S. Christiaens, C. Granrath, and M.-A. Meyer, "Scenario-and model-based systems engineering for highly automated driving," *ATZ worldwide*, vol. 122, no. 12, pp. 16–21, 2020.

[17] M.-A. Meyer, S. Silberg, C. Granrath, C. Kugler, L. Wachtmeister, B. Rumpe, S. Christiaens, and J. Andert, "Scenario-and model-based systems engineering procedure for the sotif-compliant design of automated driving functions," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1599–1604.

[18] E. Architect, "Sparx systems," 2010.

[19] ISO34501, "Road vehicles — Terms and definitions of test scenarios for automated driving systems," *International Organization for Standardization, Geneva, Switzerland*, 2021.

[20] ISO34502, "Road vehicles — Scenario-based safety evaluation framework for automated driving systems," *International Organization for Standardization, Geneva, Switzerland*, 2021.

[21] U. R. No, "157—Automated Lane Keeping Systems (ALKS)," *Nations Economic Commission for Europe: Geneva, Switzerland*, pp. 75–137, 2021.