Scenario Generation for Testing Automated Driving Systems

Lukas Birkemeyer, Ina Schaefer

{lukas.birkemeyer, ina.schaefer}@kit.edu

Abstract: The SOTIF-standard (ISO 21448) establishes scenario-based testing as state-of-the-art for verifying and validating Advanced Driver Assistance Systems (ADAS) and Automated Driving Systems (ADS). However, the SOTIF standard lacks details for selecting scenarios used as test cases. Consequently, missing details of SOTIF hinder its practical application. In this paper, we analyze existing scenario generation techniques and discuss whether they generate SOTIF-compliant scenario suites. Subsequently, we leverage variability modeling techniques to address two essential challenges that remain open: How to model an overall scenario space and how to cover it practically? We elaborate on sampling strategies and coverage criteria that are relevant for generating SOTIF-compliant scenario suites. Finally, we assess and compare generated scenario suites using mutation testing indicating the ability of scenario suites to detect potential failures.

1 Introduction

Highly automated Advanced Driver Assistance Systems (ADASs) are widely spread in modern vehicles, aiming to increase comfort, safety, and sustainability in road traffic. Increasing the autonomy of ADAS up to Automated Driving Systems (ADSs) potentially improves both aspects even further. However, increasing autonomy simultaneously increases the testing requirements. Testing strategies that are sufficient for testing ADAS become insufficient for verifying and validating ADS. For example, Wachenfeld and Winner [WW16] state that distance-based testing is practically not suitable to verify and validate highly automated ADAS/ADS due to the extremely large number of possible test kilometers required. Instead, the current Safety of the Intended Functionality (SOTIF) standard (ISO 21448) [iso22] establishes *scenario-based testing* as state-of-the-art to test highly automated ADAS/ADS. Scenario-based testing includes the environment of the vehicle in the testing process to evaluate the interaction of the System Under Test (SUT) with the environment. A *scenario* is a test case in scenario-based testing describing the environment of a vehicle, including the road infrastructure, static/dynamic actors, and weather conditions. In practice, to perform scenario-based testing, a *scenario suite*, a set of scenarios, is required. The SOTIF standard [iso22] has two requirements for those scenarios suites: (Req-1) The scenario suite needs to *sufficiently* cover all possible scenarios that potentially occur in the operational design domain of the ADAS/ADS. (Req-2) Scenarios need to minimize the set of hazardous scenarios. A scenario is *hazardous* if it contains triggering conditions turning functional insufficiencies of the SUT into hazardous behavior potentially causing harm [iso22].

Although the SOTIF standard describes requirements for scenario suites, it lacks details on how to generate them. Modeling and testing all possible real-world scenarios is practically not feasible due to the extremely large number of possible scenarios [iso22]. Thus, regarding (Req-1), it is unclear how to define a coverage criterion to *sufficiently* cover the space of possible scenarios. Regarding (Req-2), it is unclear how to determine whether a scenario contributes to minimizing the set of hazardous scenarios. Moreover, the SOTIF standard lacks details for assessing scenarios. Thus, for test engineers, it is not possible to evaluate whether a scenario suite is *good* for testing ADAS/ADS. Without defining those missing details, concrete methodologies to generate scenarios suites, or metrics to assess generated scenarios, it is not possible to apply the SOTIF standard practically.

In this paper, we pursue the overarching research goal: Generation of SOTIF-compliant scenario suites to practically verify and validate ADAS/ADS. We discuss existing scenario generation methods wrt. the SOTIF standard and identify current research gaps towards SOTIF-compliant scenario generation. We introduce a combinatorial scenario generation concept capable of generating scenario suites according to a coverage criterion (Req-1). Subsequently, we contribute two improvements towards closing existing research gaps. The improvements extend combinatorial scenario generation for generating scenario suites that additionally comply with (Req-2). We assess generated scenario suites by their ability to detect potential failures in an SUT and evaluate the proposed improvements for combinatorial scenario generation using a large-scale automotive case study.

2 State of the Art: Scenario Generation

To determine whether existing scenario generation approaches can already generate SOTIF-compliant scenario suites, we performed an Systematic Literature Review (SLR) [BKS23] according to the guidelines of Kitchenham [Kit04]. We defined a search string that we applied to search engines of scientific databases and defined objective inclusion/exclusion to collect relevant literature proposing scenario generation approaches for testing ADAS/ADS. In total, the following findings are based on 150 selected articles. We observe that existing scenario generation approaches are systematic, i.e., not random. Prime examples for systematic scenario generation approaches that can automatically generate large-scale scenario suites are data-driven, optimization-based, or combinatorial approaches.[1]

To analyze existing scenario generation approaches for SOTIF-compliance, we determined which entities, potentially occurring in a scenario, are covered (Req-1). Moreover, we determined whether existing scenario generation approaches consider knowledge of the SUT. The SUT-knowledge is relevant for generating scenario suites that comply with (Req-2); holistically considering the scenario space independently of the SUT, in turn, is relevant for generating scenario suites that comply with (Req-1). Additionally, we determined whether the behavior of the SUT for scenario generation has to be *known* or *unknown* and whether a generated scenario is *hazardous* or *not hazardous*, as the SOTIF standard [iso22] requires to minimize the set of (*unknown*) *hazardous* scenarios (Req-2).

Although existing scenario generation approaches can in principle address all *types* of

---

[1]Techniques to generate scenarios are ordered according to their share in current literature.

scenario entities,[2] they do not cover each possible entity potentially occuring in the real world. Optimization-based approaches contribute to minimizing the set of *hazardous* scenarios (Req-2) since they consider the knowledge of the SUT to exploit the space of possible scenarios. These approaches assume that critical scenarios will likely trigger hazardous behavior of the SUT. Combinatorial scenario generation approaches, in turn, are beneficial to generate scenario suites covering the scenario space (Req-1) since they holistically explore the scenario space independently of the SUT. We conclude that none of the existing scenario generation approaches generates SOTIF-compliant scenario suites. Existing scenario generation approaches either address (Req-1) by exploring the possible scenario space or (Req-2) by exploiting the possible scenario space. Based on our findings, we identify two research gaps that remain open: (1) combining existing scenario generation techniques to address both requirements of the SOTIF standard, and (2) balancing exploration vs. exploitation.

3 Combinatorial Scenario Generation

In this paper, we combine scenario generation approaches addressing both SOTIF-requirements to close existing research gaps hindering the practical application of the SOTIF standard. As a basis, we describe a combinatorial scenario generation approach complying with (Req-1). Moreover, we describe a method for assessing scenario suites in accordance with SOTIF.

Scenario Generation: In [BPV$^+$22], we propose a combinatorial scenario generation approach leveraging variability modeling techniques for generating scenario suites. Variability modeling techniques use *feature models* to capture the space of possible configurations [ABKS13]. In simulation tools used to test ADAS/ADS, a *feature* of the feature model describes an atomic scenario entity. Thus, the feature model represents the space of all possible scenarios that might be simulated using the simulation tool. Selecting a set of features from the feature model results in a scenario in the simulation tool. We apply established sampling strategies (namely uniform random sampling and feature interaction coverage sampling) to derive suites of scenarios from the scenario space feature model. Uniform random sampling selects scenarios randomly, while feature interaction coverage sampling covers all interactions of features systematically, i.e., the interaction of a specific number of atomic scenario entities.

Scenario Assessment: In [BPV$^+$22], we propose mutation testing to assess generated scenario suites. Mutation testing is an established method in software engineering to assess test cases according to their ability to detect potential failures in an SUT [JH10]. In mutation testing, artificial errors are seeded in the SUT to represent faults potentially occurring in the real world. A *mutation score* indicates the share of killed *mutants* (i.e., number of detected faulty SUT). To decide whether a mutant is killed or alive, mutation testing refers to a *kill criterion*. In accordance with the SOTIF standard, we develop a kill criterion explicitly focusing on safety-critical incidents [BPV$^+$22].

Our results in [BPV$^+$22] demonstrate that feature models are suitable for representing scenario spaces. Moreover, feature interaction coverage sampling can reduce the number

---

[2]We define *types* of scenario entities, according to the classification proposed by Bagschik et al. [BMM18]

of test scenarios up to 96% while maintaining their fault detection capabilities in terms of a high mutation score. We conclude that variability modeling techniques are beneficial for generating scenario suites that sufficiently cover a scenario space (Req-1).

4 SOTIF-compliant Scenario Generation

We propose two concepts to improve the combinatorial scenario generation approach towards SOTIF-compliant scenario generation by additionally addressing (Req-2).

Selective Sampling: The first improvement addresses the coverage criterion. Since the SOTIF standard lacks details of a concrete coverage criterion, in [BPV+22], we applied feature interaction coverage sampling using a combinatorial interaction coverage criterion. However, Kaltenecker et al. [KGS+19] state that feature interaction coverage sampling does not consider the number of features included in a configuration. In the scenario generation context, the number of features correlates to the complexity of a scenario. A crowded scenario is more complex than a scenario without road traffic. Thus, a scenarios' complexity is relevant for defining *sufficient* coverage.

To realize a complexity-aware coverage criterion, in [BPRS24], we adapt and modify the distance-based sampling approach provided by Kaltenecker et al. [KGS+19]. We introduce *selective sampling* considering the complexity distribution of an overall scenarios space. The intention is to generate a scenario suite with the same complexity distribution as the overall scenario space. Additionally, similar to distance-based sampling [KGS+19], we combine selective sampling with feature coverage criterion by prioritizing scenarios containing atomic scenario entities (i.e., features) that are not covered in the current scenario suite [BPRS24].

Our results in [BPRS24] indicate that selective sampling maintains the testing quality of generated scenario suites in terms of mutation scores compared to feature interaction coverage and distance-based sampling. Additionally, selective sampling generates scenario suites that cover the complexity distribution of an overall scenario space.

Parameter Sampling: The second improvement expands combinatorial scenario generation regarding (Req-2) by combining combinatorial scenario generation with parameter sampling [BFGS23]. We observe that scenarios contain discrete and continuous parameters. In combinatorial scenario generation [BPV+22], we discretize continuous parameters using equivalence classes based on expert knowledge assuming all parameter values of an equivalence class trigger equivalent behavior of the SUT. However, in [BFGS23], we demonstrate that equivalence classes are not sufficient.

Instead, in [BFGS23], we propose splitting the scenario sampling process into two consecutive steps: First, we select *discrete* scenario entities using feature interaction coverage sampling, and second, we select concrete parameter values of *continuous* parameters using parameter sampling. To realize parameter sampling, different sampling techniques such as random, equidistant, or optimization-based sampling can be applied. A feedback mechanism can improve parameter sampling by producing critical scenarios potentially minimizing the set of (unknown) hazardous scenarios. Depending on the definition of the parameter ranges and sampling strategy, test engineers can balance exploration vs. exploita-

tion of the scenario space. For example, subdividing the full possible parameter range into small subparameter ranges enables a strong coverage criterion.

Our results in [BFGS23] demonstrate that random parameter sampling outperforms expert-defined equivalence classes. Concerning the requirements of the SOTIF standard, parameter sampling enables to optimize continuous scenario parameters for minimizing the set of (unknown) hazardous scenarios (Req-2) while sufficiently covering a scenario space wrt. discrete scenario entities (Req-1). Thus, parameter sampling extends the combinatorial scenario generation approach towards SOTIF-compliant scenario generation. The feedback mechanism and the definition of subparameter ranges enable testers to balance exploration (Req-1) vs. exploitation (Req-2) while generating scenario suites.

5 Evaluation

In this section, we evaluate combinatorial scenario generation and its improvements [Bir25]. The fundamental experiment setup is as follows: We generate scenario suites with different strategies and assess generated scenario suites using the mutation score.

Subject System: In this evaluation, we use an Adaptive Cruise Control (ACC) case study. An ACC is an ADAS controlling the longitudinal velocity of a vehicle considering the traffic ahead. We implement a large-scale scenario space feature model covering highway scenarios relevant to test ACC functionality. To discretize continuous scenario parameters such as velocity, rain rate, or temperature, we define three equivalent classes each. The resulting feature model consists of 197 features representing a space of $152 \cdot 10^{12}$ concrete scenarios.[3]

Sampling Strategies: We generate scenario suites with feature interaction coverage sampling (namely the YASA algorithm [KTS+20]) using feature-wise (t=1) and pair-wise (t=2) coverage. We also generate scenario suites using selective sampling replicating the complexity distribution of the overall scenario space. To apply parameter sampling, we consider the continuous range of possible scenario parameter values. We generate scenario suites with purely random parameter sampling and in combination with the feedback mechanism. The feedback mechanism resamples scenario parameters until a scenario contains road traffic triggering ACC functionality. As baselines, we generate scenario suites using uniform random sampling and distance-based sampling [KGS+19]. Since most scenario sampling strategies are non-deterministic, we average over ten versions of generated scenario suites each. Inspired by the two feature interaction coverage criteria feature-wise and pair-wise, we generate scenario suites of 7 and 80 scenarios with all considered approaches.

Scenario Assessment: We generate a set of 62 unique mutants of the ACC case study that implement exactly one artificial error using the SIMULTATE mutation framework [PRWN16]. We use the same set of mutants to assess each scenario suite. As a domain-specific kill criterion, we focus on safety-critical incidents [BPV+22]

Results and Findings: In Figure 1, we present the mutation scores we determine for scenario suites generated using several sampling strategies: (a) for the smaller scenario

---

[3]Counted with SharpSAT: `https://github.com/marcthurley/sharpSAT`
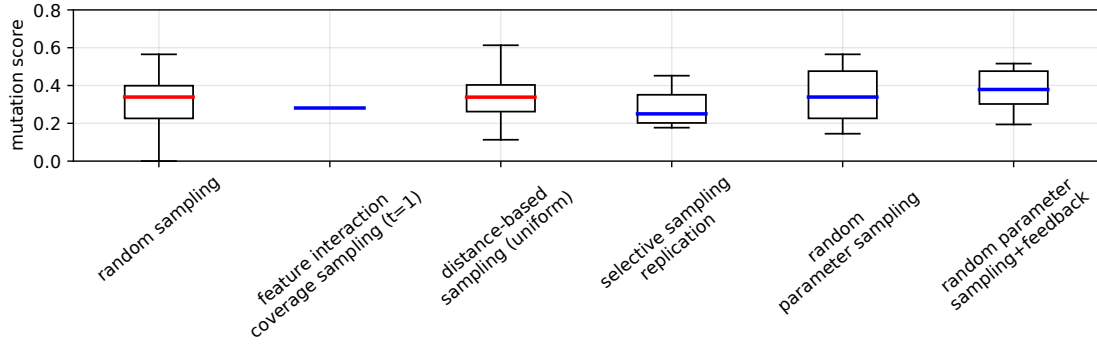
(a) Small scenario suites containing 7 scenarios
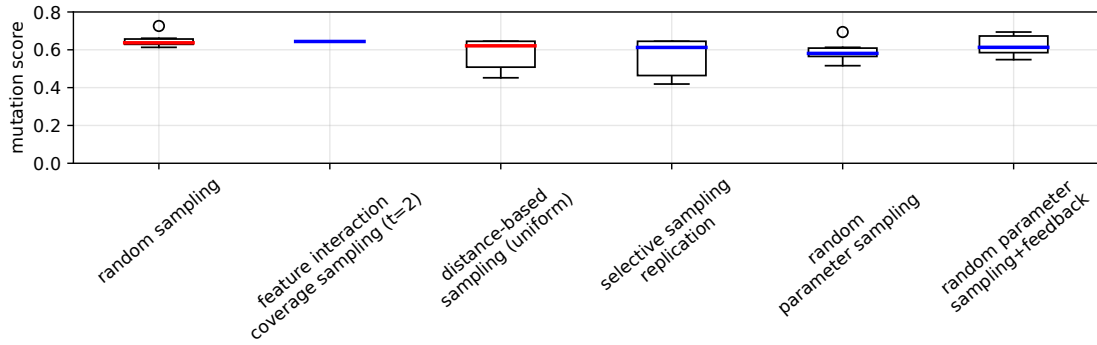
(b) Large scenario suites containing 80 scenarios

Figure 1: Mutation scores for scenario suites generated using different sampling strategies. The higher the mutation score, the better is the scenario suite in detecting potential failures.

suites of 7 scenarios; (b) for the larger scenario suites of 80 scenarios. The higher the median mutation score, the *better* is the scenario suite in detecting potential failures. A mutation score of 1 means that the scenario suite is capable of killing all mutants, while a mutation score of 0 means none of the potential failures is detected. A mutation score of 0.5 means that the scenario suite kills 50% of all mutants, which is as good as chance. For *feature interaction coverage sampling*, we observe that scenario suites generated using pair-wise coverage lead to *better* scenario suites than feature-wise coverage. Indeed, random sampling leads to similar median mutation scores, but the mutation scores scatter more, which impacts the reliability of a single test run. For *complexity-aware sampling*, the results reveal that distance-based sampling and selective sampling lead to similar median mutation scores compared to feature interaction coverage sampling. Standard distance-based sampling slightly improves the median mutation score for the small scenario suites. These results indicate that considering the complexity of scenarios is relevant for generating scenario suites. The additional benefit of selective sampling is an additional complexity-aware coverage criterion, which is in accordance with (Req-1). For *parameter sampling*, our results indicate that random parameter sampling improves scenario suites for small scenario suites compared to feature interaction coverage sampling and selective sampling. For large scenario suites, the median mutation scores are similar. Adding a feedback mechanism to parameter sampling further improves the generated scenario suite.

6 Conclusion

In this paper, we consider the research challenge of generating SOTIF-compliant scenario suites. To close the research gaps highlighted in [BKS23], we combine and evaluate existing concepts for generating scenario suites regarding the SOTIF standard. The combined scenario generation concept is based on a combinatorial approach proposed in [BPV+22]. Two improvements extend the combinatorial scenario generation approach towards SOTIF-compliance: 1) We improve the coverage criterion (Req-1) for sufficiently covering a scenario space using selective sampling [BPRS24] taking into account scenario complexity. 2) We combine combinatorial scenario sampling with parameter sampling [BFGS23] for generating (unknown) hazardous scenarios (Req-2) and enabling balancing exploration (Req-1) vs. exploitation (Req-2). Using a mutation score as a metric, we evaluated and compared the proposed concepts based on a large-scale ACC case study [Bir25]. Our results indicate that higher feature interaction coverages lead to *better* scenario suites. Moreover, a complexity-aware coverage criterion generates scenario suites leading to similar median mutation scores, but realizes an additional coverage criterion that is relevant wrt. (Req-1) of the SOTIF standard. Parameter sampling that uses a feedback mechanism improves the median mutation scores while generating SOTIF compliant scenario suites by addressing both requirements (Req-1) and (Req-2).

We conclude: Combining existing scenario generation approaches leads to scenario suites compliant to both requirements of SOTIF. Thus, the proposed improvements close the research gap addressed in [BKS23] towards SOTIF compliant scenario generation. In the future, we plan to determine the impact of balancing (Req-1) vs. (Req-2), i.e., focusing on covering the scenario space vs. focusing on scenarios that minimize the set of hazardous scenarios. Moreover, we aim to transfer the proposed concepts to further ADAS/ADS case studies to determine their generality. Lastly, we plan to identify the space of scenarios relevant for specific (types of) SUT to further minimize the number of test scenarios required to verify and validate ADAS/ADS.

7 Acknoledgment

References

[ABKS13] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. Feature-oriented software product lines. 2013.

[BFGS23] Lukas Birkemeyer, Julian Fuchs, Alessio Gambi, and Ina Schaefer. SOTIF-Compliant Scenario Generation Using Semi-Concrete Scenarios and Parameter Sampling. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2023.

[Bir25]   Lukas Birkemeyer. *Scenario Generation for Testing Advanced Driver Assistance Systems and Automated Driving Systems*. PhD thesis, Technische Universität Braunschweig, 2025. (forthcoming).

[BKS23]   Lukas Birkemeyer, Christian King, and Ina Schaefer. Is Scenario Generation Ready for SOTIF? A Systematic Literature Review. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2023.

[BMM18]   Gerrit Bagschik, Till Menzel, and Markus Maurer. Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1813–1820. IEEE, 2018.

[BPRS24]   Lukas Birkemeyer, Tobias Pett, Tobias Runge, and Ina Schaefer. Selective Sampling: A Complexity-Aware Sampling Strategy for Combinatorial Scenario-based Testing. 2024. (under review).

[BPV$^+$22] Lukas Birkemeyer, Tobias Pett, Andreas Vogelsang, Christoph Seidl, and Ina Schaefer. Feature-Interaction Sampling for Scenario-based Testing of Advanced Driver Assistance Systems. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 1–10, 2022.

[iso22]   ISO 21448:2022 Road vehicles — Safety of the intended functionality, 2022.

[JH10]   Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, 37(5):649–678, 2010.

[KGS$^+$19] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. Distance-based sampling of software configuration spaces. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1084–1094. IEEE, 2019.

[Kit04]   Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[KTS$^+$20] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. YASA: yet another sampling algorithm. In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 1–10, 2020.

[PRWN16] Ingo Pill, Ivan Rubil, Franz Wotawa, and Mihai Nica. Simultate: A toolset for fault injection and mutation testing of simulink models. In *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 168–173. IEEE, 2016.

[WW16]   Walther Wachenfeld and Hermann Winner. The release of autonomous vehicles. *Autonomous Driving: Technical, Legal and Social Aspects*, pages 425–449, 2016.