Leveraging Software Product Lines for Testing Automated Driving Systems

Stefan Klikovits
Institute for Business Informatics – Software Engineering
Johannes Kepler University Linz
Austria
stefan.kilkovits@jku.at

Alessio Gambi
Institute for Digitalization
IMC University of Applied Sciences Krems
Austria
alessio.gambi@fh-krems.ac.at

Deepak Dhungana
Institute for Digitalization
IMC University of Applied Sciences Krems
Austria
deepak.dhungana@fh-krems.ac.at

Rick Rabiser
CDL VaSiCS, LIT CPS Lab
Johannes Kepler University Linz
Austria
rick.rabiser@jku.at

ABSTRACT

Extensive testing of Automated Driving Systems (ADS), such as Advanced Driver Assistance Systems and Autonomous Vehicles, is commonly conducted using simulators programmed to implement various driving scenarios, a technique known as scenario-based testing. ADS scenario-based testing using simulations is challenging because it requires identifying scenarios that can effectively test ADS functionalities while ensuring that driving simulators' features match the driving scenarios' requirements. This short paper discusses the main challenges of systematically conducting simulation-based testing and proposes leveraging Software Product Line techniques to address them. Specifically, we argue that variability models can be used to support testers in generating test scenarios by effectively capturing and relating the variability in driving simulators, testing scenarios, and ADS implementations. We conclude by outlining an agenda for future research in this important area.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; **Software testing and debugging**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

KEYWORDS

Autonomous Vehicles, Scenario- and Simulation-based Testing, Software Product Lines

ACM Reference Format:
Stefan Klikovits, Alessio Gambi, Deepak Dhungana, and Rick Rabiser. 2024. Leveraging Software Product Lines for Testing Automated Driving Systems. In *18th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2024), February 7–9, 2024, Bern, Switzerland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3634713.3634720

VaMoS 2024, February 7–9, 2024, Bern, Switzerland

ACM ISBN 979-8-4007-0877-0/24/02.
https://doi.org/10.1145/3634713.3634720

1 INTRODUCTION AND MOTIVATION

Automated Driving Systems (ADSs), which include Advanced Driver Assistance Systems and Autonomous Vehicles, carry the promise to improve transportation and mobility. On the one hand, they aim to reduce accidents and their criticality drastically; on the other hand, they aim to improve fuel consumption and passenger comfort.

ADSs are complex, safety-critical systems in which software and hardware cooperate; additionally, they are increasingly embedded with Machine Learning and Artificial Intelligence capabilities, making thorough validation difficult.

Thereby, ADSs are an instance of a class of autonomous safety-critical Cyber-Physical Systems (CPSs), including autonomous drones, pilotless delivery robots, automated maritime vehicles, etc. Consequently, the challenges found in the testing and verifying ADSs are shared and representative of this class of systems.

Testing ADSs is commonly conducted using computer simulators for cost-efficiency and safety reasons: using simulations, ADSs can be tested at different levels of abstraction (Model-, Software- and Hardware-in-the-loop [21]) in nominal and critical scenarios, like car crashes, that are rare to observe [18] and too expensive to replicate in real life.

Scenario-based testing using simulations is challenging because it requires finding a suitable match among ADSs functionalities under test, driving scenarios, and simulators to test them. The possible concrete driving scenarios that can be implemented are infinite; thus, developers need to identify *relevant* driving scenarios, i.e., driving scenarios that effectively stress the target ADSs' functionalities. Additionally, existing driving simulators do not offer a standard interface, do not implement the same range of features, and are not generally well documented. Consequently, developers must manually identify simulators that offer the functionalities necessary to implement the selected scenarios and are also compatible with ADSs' implementation.

Developers would benefit from an approach that can systematically handle such complexity, and, in this short paper, we argue that variability models can be used to effectively capture and relate the variability in driving simulators, driving scenarios, and ADSs' implementations. Leveraging software product line techniques can

enable the design of smart systems that support testers in generating test scenarios by recommending valid scenarios and simulator combinations suitable for testing ADSs.

Software product line (SPL) engineering [9] has been proven useful to support systematic, large-scale reuse and customisation of software in many domains [19, 28]. SPL engineering techniques, specifically dynamic software product lines [7], have supported the context-dependent reconfiguration of autonomous vehicles [17]. Also, feature models [10] have been used to model the (physical) variability of autonomous underwater vehicles [8]. However, to our knowledge, only one line of research has focused on using product line techniques to support scenario-based testing of autonomous vehicles, or, more specifically, of advanced driver assistance systems [6]. In their work, Birkemeyer et al. also confirm that it is challenging to select a set of representative test scenarios and to assess the effectiveness of a test scenario suite. They leverage a feature model to select scenarios from a scenario space and assess the resulting scenario suite's effectiveness using a mutation score.

This paper proposes a more general, multi-model approach to support ADSs testing. Specifically, we propose to use multiple variability models to represent the variability of scenarios, simulators, and agents (ADSs' implementations) and relate them via *cross-model* constraints. We envision creating smart recommender systems that can support developers in generating test scenarios based on these related models.

2 THE CHALLENGE OF ADS TESTING

The co-existence of ADSs alongside regular traffic participants brings new challenges for ensuring their reliability and safety. Nowadays, scenario-based testing using simulations is a *de-facto* standard for ADS validation. Thereby, an ADS *agent* is executed in several different *driving scenarios*. Real-world testing, as proposed by safety agencies such as EURO NCAP,[1] is prohibitively expensive and dangerous; therefore, most of the validation is conducted in *simulated environments* that enable testers to fully control every aspect of the execution, generate safety-critical scenarios, and test the agent at different level of abstractions.

For instance, assume that a developer must test the collision avoidance system (CAS) software, which is central for the safety of the self-driving car, using simulations. To do so, the developer has to choose a set of scenarios that stress the component and can expose issues in its implementation. For example, suitable scenarios might involve other traffic participants, such as pedestrians crossing the road or vehicles cutting into the lane, and testing whether the agent can avoid them. Conversely, scenarios that do not involve any other traffic participant might not be helpful to the developers for testing CASs. Those high-level scenarios (i.e., abstract scenarios) must be refined and implemented into running driving simulations (i.e., concrete scenarios). For instance, the developer must choose the number of pedestrians simulated, how they are dressed, how they move, and so on. Likewise, simulating traffic requires placing and moving vehicles of different types and sizes in a realistic fashion.

Over the years, numerous simulators (e.g., BeamNG.tech [4], Carla [12], LGSVL [26]) and driving agents (e.g., Apollo [1], Autonomoose [2]) were developed and tested on a plethora of different scenario sets (e.g., EuroNCAP [15], SafetyPool [11]).

2.1 Variability

Given this variety, the selection of a "good" combination of ADS, simulator and scenario (set) is not always straightforward. For instance, testing a specific capability, such as the CAS of a certain agent, requires a simulator and scenario set that will execute it accordingly.

Agent Variability. This variability may be represented using methods known from the variability and software product lines domain. For instance, an autonomous driving agent could be described using a feature model (FM) as shown in Figure 2a. This (reduced) FM shows that an agent is implemented at a specific Abstraction Level, which defines whether it is a Software component, an abstract Model (such as the kinematic bicycle model [24]) or if it is integrated into a Hardware setup where it has to be tested as e.g. hardware in the loop unit.

Similarly, depending on the type of agent, different Features are available, such as Sensors, Knowledge sources (e.g., access to maps), and control features that allow the vehicle to operate (e.g., Steering, Throttle, Brake).

Scenario Variability. Scenarios may also be represented using FMs, as exemplified in Figure 2b. Here, we see that a scenario specifies the agent under test's Mission (Initial State and Objectives), the layout of the Map and the position, shape and behaviour of various Obstacles, such as other Traffic Participants, and any Environment conditions that might be required for implementing this specific scenario.[2]

Simulator Variability. Finally, the individual simulators' variability might also be expressed as FM (cf. Figure 2c). Note how the simulator FM reflects many features that were similarly described in the previous FMs. For instance, to enable hardware in the loop testing, the simulator has to provide the capability to include such a scenario. Conversely, certain simulators might be "too detailed" for testing abstract kinematic models.

Furthermore, the simulator should also support a scenario's defined specifications. Hence, the prescribed weather and lighting conditions can only be simulated, if the tool supports such an alteration of the simulated environment, which is not always the case.

3 MULTI-MODEL APPROACH

In the context of testing ADSs, reusable assets of SPLE can translate into reusable driving agents (e.g., vehicles), simulation environments, and test scenarios. We propose to model the variability of these three artefacts in three different models, as they can be seen as three individual product lines. An overview of the proposed approach is depicted in Figure 1. As reported by Reiser and Weber [25], attempting to model various viewpoints and distinct product lines using a global feature model is often unfeasible in practice.

[1] https://www.euroncap.com/

[2] Note that an exhaustive scenario description exceeds the scope of this work.

Many multi-product line approaches have thus been proposed to address this issue [16].

Agent Feature Model outlines the various attributes, components, and options available for a specific ADS, which is the subject of the test. It includes typical hardware characteristics and software modules that can be customised or configured (see example in Figure 2a).

Scenarios Feature Model outlines the various attributes, functionalities, and customisation options available in a software artefact describing the main variables of a driving experience. It includes road geometry and map, driving tasks (i.e., the mission to accomplish), traffic and pedestrian behaviour, etc. (see example in Figure 2b).

Simulator Feature Model outlines the various attributes, functionalities, and customisation options available in a software application designed to simulate the experience of driving a vehicle. It includes the simulator's capabilities, e.g., driving physics, camera views, control options, location/terrain, time and weather conditions, etc. (see example in Figure 2c).

Even though the three models represent independent product lines, when it comes to generating test cases, all these models must be configured correctly and must consider the options selected in other models. Consequently, there exist constraints across the different models, which must be considered and captured formally in a model-based approach. In our approach, these *Cross-Model Constraints* are similar to the invar approach Galindo et al. [14] and can also be seen as "cross-discipline constraints" as reported by Fadhlillah et al. [13]. The formulation of the Cross-Model Constraints allows us to have a unified view of all the different product lines – a Unified Configuration Model is created. Figure 2d shows some examples of Cross-Model Constraints.

The *Unified Configuration Model* is the basis for the generation of the *Test Cases*. Instead of creating new test scenarios for each vehicle model or software version, our approach enables the generation of test cases based on configured scenarios, agents, and simulation environments. The tester provides the test oracles, i.e., assertions defining the system under test's expected behaviour. Next, an automated reasoning engine completes the configuration and generates the required testing artefacts, which include the test scripts for setup, execution, and verification (i.e., test cases), and the configuration parameters for the agent and the simulator.

In the long run, as the variability models become stable, test cases can be designed to cover different feature combinations and configurations, leading to more thorough testing. With a model-based approach, collaboration and knowledge sharing within the testing community can be enhanced as the reusable scenarios and test cases can be shared and adapted by different teams working on various aspects of autonomous vehicle testing.

4 DISCUSSION

Our method suggests the joint use of several variability models as a basis to find appropriate tools and scenarios for ADS testing. Indeed, as outlined in the previous section, if used correctly, our approach could relieve a big burden on the ADS industry. Nonetheless, we note that several challenges still remain. In this section, we outline some of them.

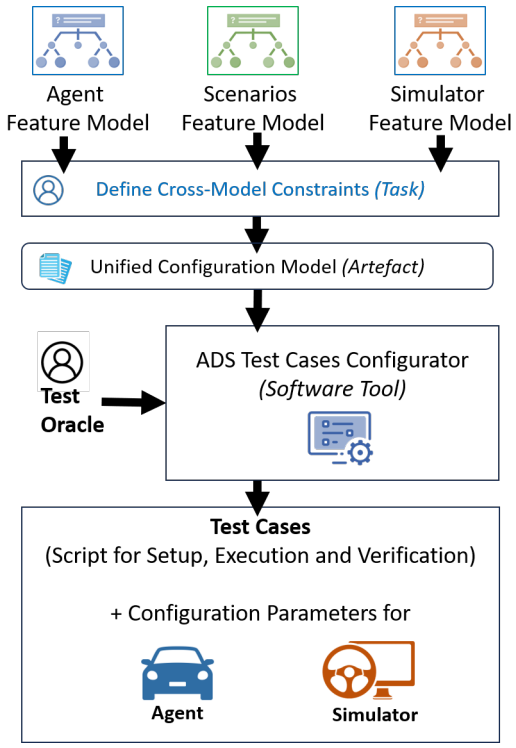


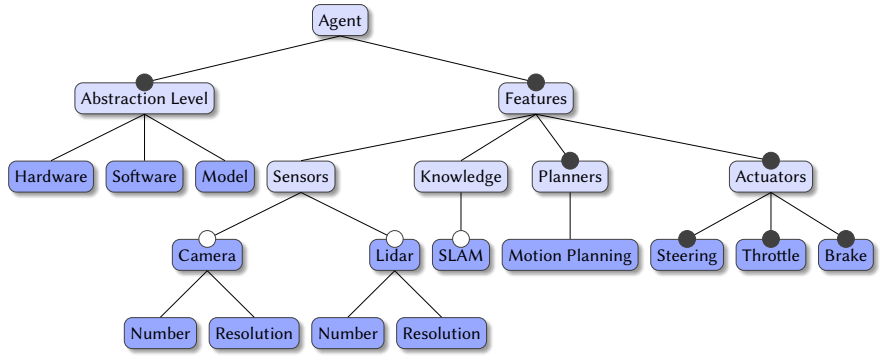
Figure 1: Overview of the proposed approach.

Information Sources. The FMs presented in Figure 2b, Figure 2a, and Figure 2c are limited examples that do not encompass the full complexity of ADS agents, scenarios and simulators. Additionally, some features are specific to individual tools or have certain requirements (e.g., proprietary APIs) that render them tool-specific. One difficulty might be obtaining and integrating this information into a common FM format and maintaining this information (cf. the challenge on automating variability management below).

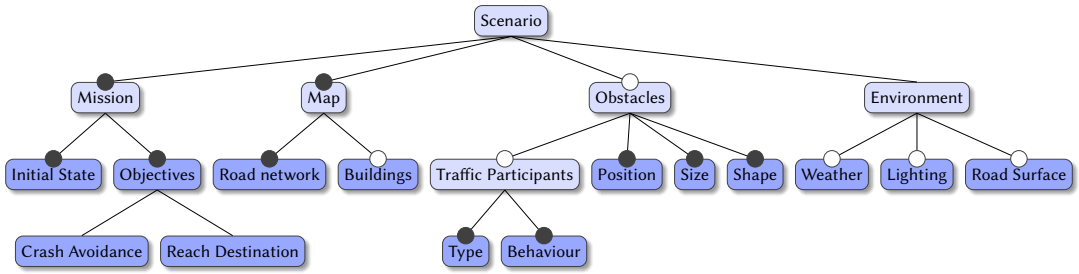
Scenarios are typically stored in XML or JSON format and follow commonly available standards (e.g., OpenScenario [23]), allowing them to be parsed rather easily. Furthermore, over the years, the academic community has described the use of various sources as the basis for the design of scenarios, including legal documents, traffic rules, accident reports, etc.

On the other hand, finding a parsable representation that allows a complete description of agent or simulator features might be a bit more difficult. One potential solution might be the use of (internal) APIs of open-source software. Industrial or closed-source tools, however, would still require manual editing.

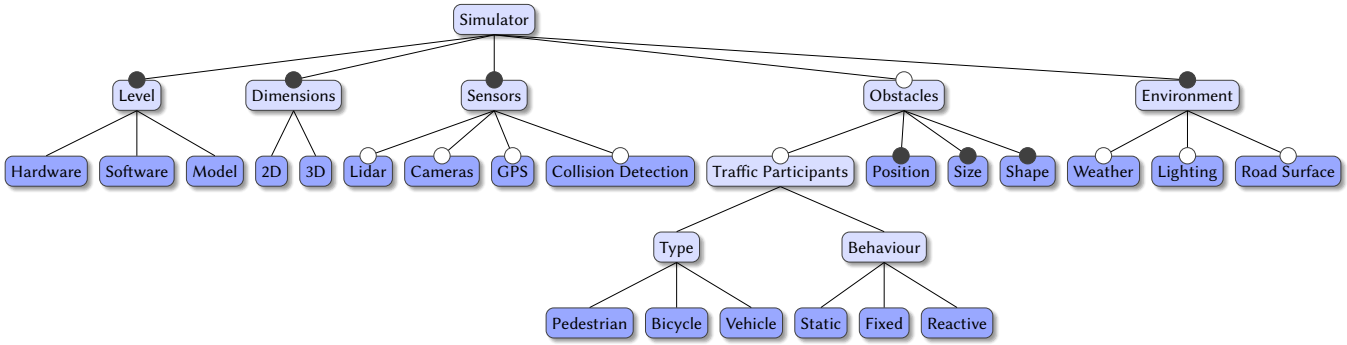
Heterogeneity. Another challenge is the heterogeneity of the developed tools, which produces inherent complexity, naming conflicts and similar issues. Even though recent years saw the rise of several standardisation agencies trying to tame this situation by suggesting standards, the proposed standards remain volatile and subject to frequent changes, requiring continuous efforts. Additionally, the simulators and agents might express features at different levels of abstraction (e.g. weather could be modelled as "rainy" vs



(a) Agent FM



(b) Scenario FM



(c) Simulator FM

$$\text{Agent}:\text{Level} \implies \text{Simulator}:\text{Level}$$
$$\text{Agent}:\text{Cameras} \implies \text{Simulator}:\text{Cameras}$$
$$\text{Scenario}:\text{Environment} \implies \text{Simulator}:\text{Environment}$$
$$\text{Scenario}:\text{Crash Avoidance} \implies \text{Simulator}:\text{Collision Detection}$$
$$\text{Scenario}:\text{Crash Avoidance} \implies \text{Simulator}:\text{Pedestrian OR Scenario}:\text{Vehicle}$$
$$\text{Scenario}:\text{Traffic Participants} \implies \text{Simulator}:\text{Traffic Participants}$$

(d) Cross-Model Constraints

Figure 2: Multi-model approach: examples of Agent, Scenario, and Simulator Feature Models and Cross-Model Constraints.

"sunny", or in full detail, including the amount of precipitation, raindrop size, humidity, etc). This means that it is necessary to find an adequate way to represent these abstraction levels across the individual FMs.

Thus, while the creation of FMs is straightforward on a theoretical level, practical concerns might cause the approach to be more involved than expected.

Automating variability management. A main reason why variability modelling approaches and tools are often not adopted in practice is the manual effort required to create and maintain variability models [5]. The proposed multi-view modelling does not resolve this situation but might make it more manageable by following a divide-and-conquer approach. If the ADS testing community succeeds in standardising terminology as well as scenario representation and format, one could develop a tool that can (semi-)automatically

populate and maintain variability models describing scenarios. Similarly, it might be possible to develop such tools to populate and maintain simulator feature models and agent feature models. A key challenge that will be important to address is also to automate (at least partly) the definition and maintenance of the cross-model dependencies. Automatically populating and maintaining variability models is a general challenge in the variability modelling community, one that has seen some proposals – e.g., feature identification and extraction approaches [3, 20] and constraint extraction approaches [22, 27] – but yet needs to be solved on a more general level.

Additional Challenges. Next to the above challenges, we also note that several challenges remain that are closer to the operational aspects of PLs. These include achieving traceability in the presence of model composition, efficiently solving cross-model constraints, and maintaining the consistency of the multiple models.

5 NEXT STEPS

In this short paper, we presented our idea of pursuing a multi-model approach to capture and relate the variability in driving simulators, testing scenarios, and autonomous vehicle implementations to enable the design of smart recommender systems that support testers in generating test scenarios for ADSs. We have described several challenges that yet remain to be solved in future work. As a next step, we will develop the system, which, based on the related variability models, presents options to testers and allow them to generate test cases. Existing tools such as FeatureIDE [3] or pure::variants [4] with their support for creating and relating, as well as configuring multiple feature models, will be a good starting point.

REFERENCES

[1] [n. d.]. Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving. https://github.com/ApolloAuto/apollo. Accessed: 2019-02-11.
[2] Michał Antkiewicz, Maximilian Kahn, Michael Ala, Krzysztof Czarnecki, Paul Wells, Atul Acharya, and Sven Beiker. 2020. Modes of Automated Driving System Scenario Testing: Experience Report and Recommendations. *SAE International Journal of Advances and Current Practices in Mobility* 2, 4 (apr 2020), 2248–2266. https://doi.org/10.4271/2020-01-1204
[3] Noor Hasrina Bakar, Zarinah M Kasirun, and Norsaremah Salleh. 2015. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software* 106 (2015), 132–149.
[4] BeamNG GmbH. [n. d.]. *BeamNG.tech.* https://www.beamng.tech/
[5] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *Proc. of the 7th International Workshop on Variability Modelling of Software-intensive Systems*. 1–8.
[6] Lukas Birkemeyer, Tobias Pett, Andreas Vogelsang, Christoph Seidl, and Ina Schaefer. 2022. Feature-Interaction Sampling for Scenario-based Testing of Advanced Driver Assistance Systems. In *Proc. of the 16th Int'ernationa'l Working Conf. on Variability Modelling of Software-Intensive Systems*. 1–10.
[7] Rafael Capilla, Jan Bosch, Pablo Trinidad, Antonio Ruiz-Cortés, and Mike Hinchey. 2014. An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry. *Journal of Systems and Software* 91 (2014), 3–23.
[8] Carlos Cares, Daniel Lühr, Sandro Mora, Claudio Navarro, Leonardo Olivares, Samuel Sepúlveda, and Gastón Vidal. 2022. Architecting Autonomous Underwater Vehicles by Adapting Software Product Lines. In *Proc. of the Conf. on Integrated Computer Technologies in Mechanical Engineering–Synergetic Engineering*. Springer, 719–730.
[9] Paul C. Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.
[10] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool features and tough decisions: a comparison of variability modeling approaches. In *Proc. of the 6th Int'l Workshop on Variability Modelling of Software-intensive Systems*. 173–182.
[11] Deepen AI and WMG University of Warwick, UK. [n. d.]. Safety Pool Scenario Database. https://www.safetypool.ai/
[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.
[13] Hafiyyan Sayyid Fadhlillah, Kevin Feichtinger, Kristof Meixner, Lisa Sonnleithner, Rick Rabiser, and Alois Zoitl. 2022. Towards multidisciplinary delta-oriented variability management in cyber-physical production systems. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*. 1–10.
[14] José A Galindo, Deepak Dhungana, Rick Rabiser, David Benavides, Goetz Botterweck, and Paul Grünbacher. 2015. Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. *Information and Software Technology* 62 (2015), 78–100.
[15] C Adrian Hobbs and Paul J McDonough. 1998. Development of the European new car assessment programme (Euro NCAP). *Regulation* 44, 3 (1998), 2439–2453.
[16] Gerald Holl, Paul Grünbacher, and Rick Rabiser. 2012. A systematic review and an expert survey on capabilities supporting multi product lines. *Information and Software Technology* 54, 8 (2012), 828–852.
[17] José Miguel Horcas, Julien Monteil, Mélanie Bouroche, Mónica Pinto, Lidia Fuentes, and Siobhán Clarke. 2018. Context-dependent reconfiguration of autonomous vehicles in mixed traffic. *Journal of Software: Evolution and Process* 30, 4 (2018), e1926.
[18] Nidhi Kalra and Susan M. Paddock. 2016. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice* 94 (2016), 182–193. https://doi.org/10.1016/j.tra.2016.09.010
[19] Jabier Martinez, Wesley KG Assunção, and Tewfik Ziadi. 2017. Espla: A catalog of extractive spl adoption case studies. In *Proc. of the 21st Int'ernationa'l Systems and Software Product Line Conference-Volume B*. 38–41.
[20] Jabier Martinez, Tewfik Ziadi, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. 2017. Bottom-up technologies for reuse: automated extractive adoption of software product lines. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 67–70.
[21] S. Moten, F. Celiberti, M. Grottoli, A. van der Heide, and Y. Lemmens. 2018. X-in-the-loop advanced driving simulation platform for the design,development, testing and validation of ADAS. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. 1–6. https://doi.org/10.1109/IVS.2018.8500409
[22] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. 2014. Mining configuration constraints: Static analyses and empirical results. In *Proc. of the 36th Int'l Conference on Software Engineering*. 140–151.
[23] ASAM OpenSCENARIO. 2020. ASAM OpenSCENARIO.
[24] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. 2017. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. 812–818. https://doi.org/10.1109/IVS.2017.7995816
[25] Mark-Oliver Reiser and Matthias Weber. 2007. Multi-level feature trees: A pragmatic approach to managing highly complex product families. *Requirements Engineering* 12 (2007), 57–75.
[26] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. https://doi.org/10.1109/ITSC45102.2020.9294422
[27] Paul Temple, José A Galindo, Mathieu Acher, and Jean-Marc Jézéquel. 2016. Using machine learning to infer constraints for product lines. In *Proc. of the 20th Int'ernationa'l Systems and Software Product Line Conference*. 209–218.
[28] Frank J Van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.

[3] https://featureide.github.io/
[4] https://www.pure-systems.com/de/purevariants