Re-Using Concrete Test Scenarios Generally Is a Bad Idea

PrePrint for the proceedings of the IEEE Intelligent Vehicles Symposium 2020
Published version: https://doi.org/10.1109/IV47402.2020.9304678

Florian Hauer, Alexander Pretschner, and Bernd Holzmüller

***Abstract*— Many approaches for testing automated and autonomous driving systems in dynamic traffic scenarios rely on the reuse of test cases, e.g., recording test scenarios during real test drives or creating "test catalogs." Both are widely used in industry and in literature. By counterexample, we show that the quality of test cases is system-dependent and that faulty system behavior may stay unrevealed during testing if test cases are naïvely re-used. We argue that, in general, system-specific "good" test cases need to be generated. Thus, recorded scenarios in general cannot simply be used for testing, and regression testing strategies needs to be rethought for automated and autonomous driving systems. The counterexample involves a system built according to state-of-the-art literature, which is tested in a traffic scenario using a high-fidelity physical simulation tool. Test scenarios are generated using standard techniques from the literature and state-of-the-art methodologies. By comparing the quality of test cases, we argue against a naïve re-use of test cases.**

I. Introduction

Striving for highly automated and autonomous driving systems results in more and more complex and capable systems. The complexity of these systems as well as the complexity and sheer number of possible scenarios makes safety and functional correctness a crucial challenge [17]. Since testing by real test drives alone becomes practically infeasible [15], [33], the focus shifts to virtual test drives. For virtual testing of vehicle safety, scenario-based closed-loop testing in the form of X-in-the-loop settings is used [30]. Such scenarios usually contain dynamic traffic situations to test the behavior of automated and autonomous driving systems. An exemplary test scenario for testing a highway pilot is depicted in Fig. 1.

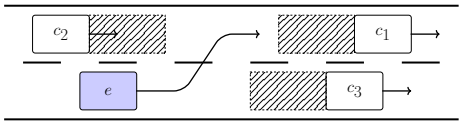


Fig. 1. Example test scenario for testing lane change functionality

The ego vehicle e is driving on a two-lane highway behind the car c_3 and performs a lane change into the gap between the cars c_1 and c_2. It is tested whether the system keeps a sufficient safety distance (shaded areas in Fig. 1) to the surrounding cars during the lane change. In case the system violates the safety distance to e.g. c_1, because the gap between c_1 and c_2 is too small, a faulty behavior of the system is revealed. Now assume that the system is updated by "correcting" this faulty behavior and that the test scenario is re-run to test whether the system behaves correctly now. In this case, it may turn out that the system does not even perform a lane change anymore, since the planning component considers the gap too narrow. Thus, by slightly modifying the system, the previously useful test case now has become essentially useless for testing whether a safety distance is violated during a lane change. Even worse: The system may still violate a minimum safety distance threshold during a lane change. This stays unrevealed unless a new suitable test case is found instead of the re-used one. Since the consequences of a system change for its behavior in dynamic traffic in general cannot be known a-priori, we argue that this problem is of a general nature.

Conventional classical tests can usually be re-used, e.g. for testing the door locking systems. In contrast, the above consideration of driving scenarios has severe implications for the re-use of tests for regression testing as well as for the re-use of recorded test scenarios and tests of so-called "test catalogs." All these approaches rely on the idea of creating test cases once and re-using them later on. They are widely used in industry and literature. Unfortunately, as the simple example conveys, naïvely re-using test scenarios in general cannot directly provide the basis for an argumentation about safety and functional correctness.

The **contribution** of this paper is as follows. We provide a numerical counterexample to the re-usability of concrete test scenarios, heavily relied on by many approaches in industry and literature. Using standard techniques from literature, we generate test scenarios to test an autonomous driving system built according to state-of-the-art approaches. With the experimental results, we can show that naïvely re-using concrete test scenarios cannot guarantee the quality of a test as this test may not even trigger relevant behavior.

The remainder is structured as follows. §II explains scenario-based testing and what constitutes a "good" test case, before the methodology of creating the counterexample is described in §III. §IV explains the experiments and the re-usability issue, and §V provides an overview of related work. We conclude in §VI.

II. "Good" Test Cases in Scenario-Based Testing

In scenario-based testing the goal is to mimic real traffic scenarios in simulation to test the safety of the driving

F. Hauer and A. Pretschner are with Department of Informatics, Technical University of Munich, Germany, e-mail: {florian.hauer,alexander.pretschner}@tum.de
B. Holzmüller is with ITK Engineering, Germany, e-mail: bernd.holzmueller@itk-engineering.de

behavior automated and autonomous driving systems. A variety of different scenario types (lane change, emergency brake, etc.) take place in real traffic. Lists of such scenario types are derived from experience [31] and real data [11], and the completeness of such lists is determined with statistical models [13]. For each scenario type, a parameterized scenario is created, called *logical scenario* [21]. The intention is to capture the variability of the real world with n parameters P and their domains $D_j \in D, j = 1..n$, e.g. the initial velocity of other traffic participants in a scenario is not set to a specific value of $100km/h$, but is assigned a parameter v_{other} with domain $[80, 130]$. The domains span a space $A = D_1 \times D_2 \times ... \times D_n \subset \mathbb{R}^n$ of test cases. Assigning to each parameter a value from its domain yields a single, executable test case, which is called *concrete test scenario* [21]. Not every candidate in such a space A is of the correct form (e.g. the ego vehicle should perform a lane change, but does not) and among those that have the correct form not all are interesting (e.g. all other vehicles are several hundred meters away from the ego vehicle) [12]. In other words, many test scenarios in such a search space are not the "good" test scenarios we are searching for. The concrete scenario describes the input and environment conditions of a test case. The expected behavior of continuous systems is described with the help a safe operating envelope (cf. Fig. 2). Inside the envelope, the system is allowed to freely optimize its performance [17], and as long as it does not leave the envelope it is considered safe. A variety of works present such safety envelopes [24], [26], [28].

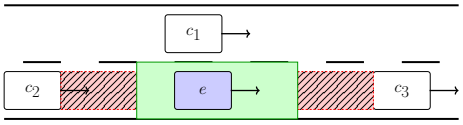


Fig. 2. Example of a safe operating envelope (green plain rectangle) bounded by the necessary safety distances (red shaded rectangle) and lane markings

In the spirit of limit testing, we define **"good" test cases** to test vehicle safety as follows [12], [25]: *A "good" test case can reveal* potential *faulty system behavior. In a "good" test scenario, a correct system approaches the limits of the safe operating envelope, and a faulty system violates them.*

III. Creation of the Counterexample

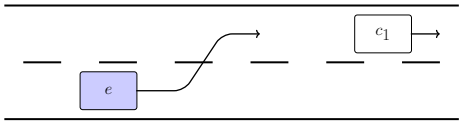
To argue against naïve re-use, we generate test cases for different variants of a system and then simulate "re-use" of these tests by applying each test to all variants. If concrete test scenarios were re-usable, the quality of a test case should not drop when re-used for another system. This requires that "good" test cases are generated and that the quality of the test cases can be measured.

Existing works suggest the use of search-based techniques for the selection of "good" test scenarios (detailed information in Sec V). Such techniques try to identify the best candidate (here: concrete test scenario) in a search space (here: logical scenario) with the help of a fitness function, which provides a quality measure for a test case of how good it is. In the following, the logical scenario, the fitness function, and the systems that are used in the experiments are explained.

A. Logical Scenario

The logical scenario used for the experiments is shown in Fig. 3. It is a simplification of the exemplary scenario in Fig. 1. The cars c_2 and c_3 have been removed. When re-using concrete test scenarios for such a simple lane change like in Fig 3, we can expect that at least the form of the concrete scenario is still correct, i.e. the ego vehicle performs a lane change behind c_1. For more complex scenario types this is not the case as illustrated for the example in Fig. 1.

The ego car will start at the longitudinal position 0m and accelerate to the initial velocity v_e. As soon as starting time t_{start,c_1} is reached, the other car starts accelerating from a longitudinal starting position s_{0,c_1} to an initial velocity v_{c_1}. As soon as both cars reach their initial velocities, the lane change request is triggered after time t_{trg}. Those parameters provide the necessary search space to allow c_1 with v_{c_1} to be located arbitrarily on the road at the point in time when e is triggered to perform a lane change.



Parameter	Lower Bound	Upper Bound
Ego vehicle e:		
Initially reached velocity v_e [m/s]	22.22 (80km/h)	36.11 (130kmh)
lane change trigger time t_{trg} [s]	0	5
Other car c_1:		
Longitudinal starting position s_{0,c_1} [m]	0	500
Starting time t_{start,c_1} [s]	0	5
Initially reached velocity v_{c_1} [m/s]	22.22 (80km/h)	36.11 (130km/h)

Fig. 3. Parameterized scenario used for the experiments

B. Fitness Function

The fitness function assigns a quality to each concrete test scenario, which allows the search-based technique to select the "good" test cases. The goal is to test whether the system keeps sufficient distance during a lane change behind another car. Thus, the fitness function has to guide the search-based technique to identify test cases of the correct form, i.e. ego vehicle performs a lane change behind the other car. Among those with the correct form, the biggest violation of the safety distance is searched. We created the fitness function according to the literature [12] (the optimizer minimizes):
The fitness function with $t \in [t_{start}, t_{end}]$:

$$f = \begin{cases} \infty; & \text{no lane change happens} \\ \begin{cases} s_e(t_{start}) - s_{c_1}(t_{start}); & s_{c_1}(t_{start}) \leq s_e(t_{start}) \\ \min\{d - safeDist(t)\}; & s_{c_1}(t_{start}) > s_e(t_{start}) \end{cases} \end{cases}$$

The first part of the fitness function assigns ∞ as a very bad, high value to scenarios in which the ego vehicle does not perform a lane change. Instead, if such a lane change takes place, the longitudinal position of the ego vehicle at the start of the lane change $s_e(t_{start})$ is compared with the one of the other vehicle $s_{c_1}(t_{start})$. If the ego vehicle is in

front of the other car, the distance between the cars at this moment $d(t_{start}) = s_e(t_{start}) - s_{c_1}(t_{start})$ is assigned as bad fitness value. The smaller this distance is, the better the fitness value. In the third case, the lane change takes place behind the other car. The difference of distance $d(t)$ and the the safety distance $safeDist(t)$ is the remaining distance until the safety distance is violated. By computing the minimum of it throughout a lane change ($t \in [t_{start}, t_{end}]$), the least safe remaining distance until violation during this lane change is determined. During the search through the search space this yields the least safe remaining distance in the whole search space. If the system behavior is faulty, it will yield the biggest violation in the search space instead. The safety distance is computed according to a formalization [26] of the Vienna Road Convention. However, other safety models may be used as well, e.g. [24], [28].

C. Driving Systems

The driving systems for the experiments are different versions and variants of a single system type for two reasons: First, we want to show that even small changes in the configuration already cause issues with the re-usability of concrete test scenarios. Changing the whole system design has a far bigger impact than small configuration changes. Second, testing different versions of a single system better represents regression testing, which is the common use case for re-using test cases.

The architecture of our demonstration system is shown in Fig. 4. It follows the general control paradigm, which contains the notions of sensing, long-term planning (or decision-making), short-term planning, tracking and actuation [4]. To ease the interpretation of the experimental results, we excluded sensing from the experiment setup. Also for the sake of simplicity, the car c_3 from the introductory example scenario has been removed as a trigger to the decision making to change lanes. Thus, also the decision making is removed. To preserve the variability of decision making caused by different relative positions of the ego vehicle and this c_3, the desire to change is triggered by a time trigger (cf. t_{trg} in Sec. III-A). For the short-term planning, a state-of-the-art approach for lane change maneuvers of automated vehicles [23] was used. This system was chosen to closely resemble automated and autonomous driving systems of SAE levels 4 & 5 [27], e.g. a Highway Pilot of SAE level 4.

This model predictive control approach first predicts the positions x_i and velocities v_i of surrounding cars c_i for each

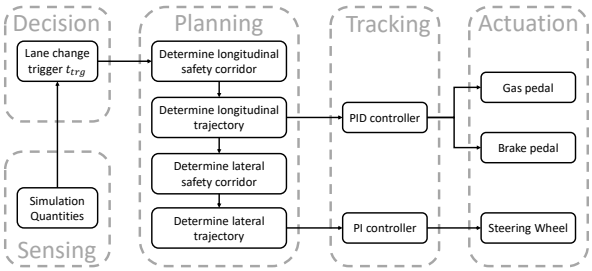


Fig. 4. System architecture overview

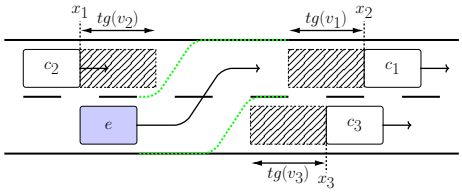


Fig. 5. Schematic simplified depiction of the predicted positions x_i and time gaps $tg(v_i)$ at a specific prediction time t_k of the system

sample time step t_k over a short time horizon (cf. Fig. 5). For safety distance planning, a time gap $\tau = 0.5s$ is used as proposed in the paper series of [23]. The safety corridor for the lane change (dashed green lines) is bounded by the distance $tg(v_i) = \tau \cdot v_i$ to the predicted positions x_i of other cars c_i. First, a safe longitudinal and afterwards a safe lateral trajectory is computed within these bounds. The objective is to keep the velocity at each time step of the trajectory as close to the velocity before the lane change (given by the scenario parameter "initially reached velocity" v_e) and the acceleration as low as possible. The tracking of the trajectories is done by classic control of the gas and brake pedals as well as the steering wheel. Both are tuned for a physical model of a sports car provided by the widely used physical simulation software CarMaker by IPG Automotive, which served as the simulation environment for this work.

IV. Experiments

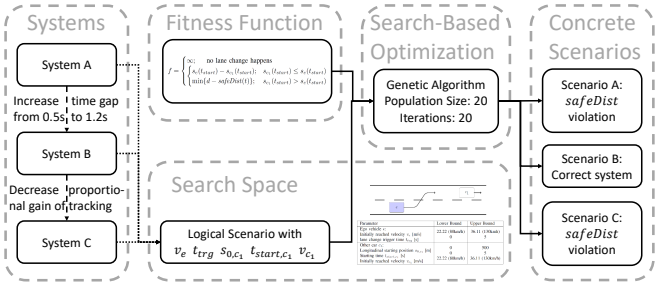


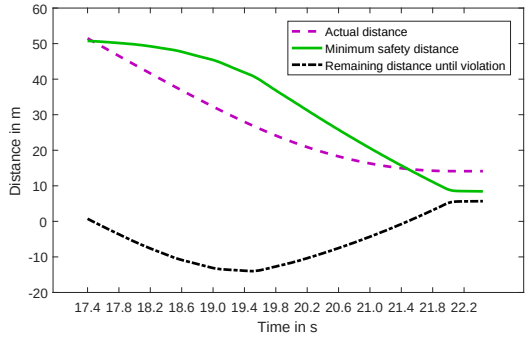
Fig. 6. Experiment Overview

In Fig. 6, an overview over the experiments is given. Three systems configurations are created (systems A,B,C). System A is the system described above, while system B is created by increasing the time gap τ from 0.5s to 1.2s, and system C is yielded by decreasing the proportional gain of the longitudinal PID controller in system B. The fitness function from Sec. III-B and the logical scenario as the search space from Sec. III-A are used for all three experiments. For the optimization, a single-objective genetic algorithm is applied. Note that the technological aspect is not the focus of this work and more advanced techniques could be used, as described in [9]. The population size and the number of generations were both set to 20, resulting in 400 simulation executions, which means each system is tested in 400 concrete scenarios during the optimization. The experiments were executed multiple times to rule out randomization effects. The scenario with the best fitness value of each experiment is presented in detail in the images of respective Fig. 7, 8, 9: (1) distances during lane change,

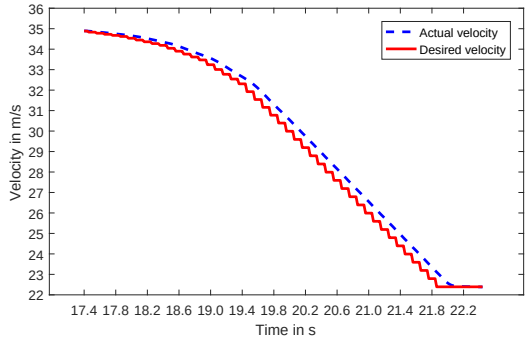
and (2) velocities during lane change. Experiment A shows that faulty behavior of the trajectory planner can be revealed, B presents a case where the system behavior is seemingly correct, and in C faulty behavior caused by slow tracking is detected.

A. Experiment Results

In **scenario A**, which is the best concrete test scenario that could be found for system A, the ego vehicle performs a lane change behind the other car, which drives at lower velocity. In Fig. 7(1), the actual distance between both cars $d(t)$, the minimum safety distance $safeDist(t)$, as well as the remaining distance until this safety distance is violated are shown (a negative remainder means violation). Approximately between scenario time 17.5s and 21.5s, the safety distance is violated (see 7(1)). The biggest violation of 14m takes place at approximately 19.5s. Even though tracking will never be perfect, here it is considerably fast (cf. Fig. 7(2)), which leads to the conclusion that the planned trajectory does not consider sufficient safety distance in this scenario with respect to the safe distance computation of the fitness function. This faulty behavior gets addressed by increasing τ in the safety distance estimation $tg(v_i)$ of the planning algorithm to 1.2s to yield system B. The concrete value of 1.2s was determined experimentally. The results of the **scenario B** are shown in Fig. 8. It can be seen that even in the best concrete test scenario, the updated time gap causes the system to keep sufficient distance to the other vehicle to not violate the minimum safety distance. Therefore, it is considered to be safe with respect to this search space. To yield system C, the proportional gain of the velocity

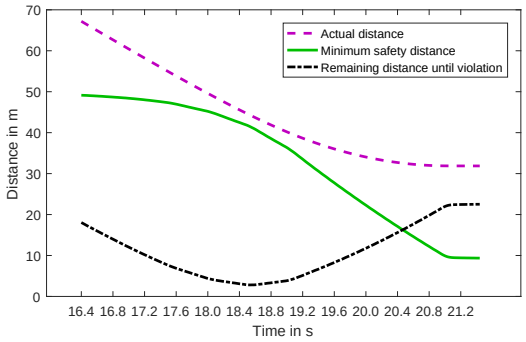


(1) Distances during the lane change of test scenario A



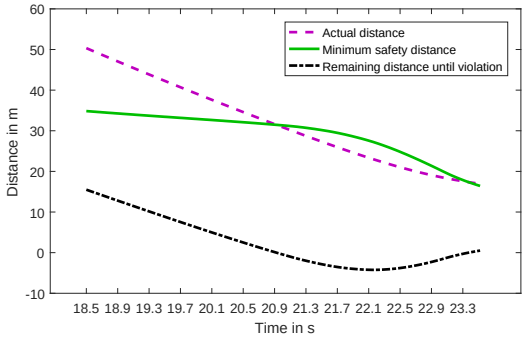
(2) Velocities during the lane change of test scenario A

Fig. 7. Results of scenario A: Faulty trajectory planner

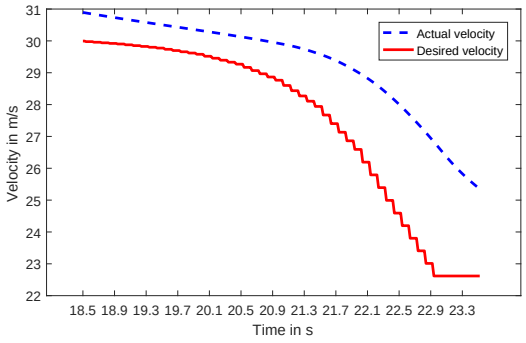


(1) Distances during the lane change oftest scenario B

Fig. 8. Results of scenario B: Seemingly safe system



(1) Distances during the lane change of test scenario C



(2) Velocities during the lane change of test scenario C

Fig. 9. Results of scenario C: Tracking too slow

tracking PID controller of system B is decreased to make the tracking more smooth, e.g. for passenger comfort. However, this also increases the rise time of the controller. The results of **scenario C** (cf. Fig. 9(2)) show that the tracking does not follow the planned trajectory close enough to decrease the ego vehicle's speed as fast as necessary, which causes a violation of the safety distance despite the increased time gap τ.

B. Why Re-Using Concrete Scenarios Is a Bad Idea

Each system (e.g. system A) is additionally tested using the best concrete test scenarios identified for the other systems (e.g. scenario B/ C). The resulting fitness values, which measure both the test case quality and the remaining buffer until violation of the safety distance in meters, are depicted in Tab. I.

	System A	System B	System C
Scenario A	-14.001	5.604	-3.065
Scenario B	2.595	2.812	-3.037
Scenario C	20.153	20.484	-4.238

TABLE I

FITNESS VALUES OF THE THREE IDENTIFIED BEST CONCRETE TEST SCENARIOS FOR THE THREE SYSTEMS

Every concrete test scenario has the best fitness value for testing the system it was created for (red cells). This confirms the intuition regarding the re-usability issue implied with the introductory example in Sec. I. The smallest remaining buffer until violation (positive fitness value) or biggest violation (negative fitness value) for distinct systems is found in different scenarios. Especially the first column of Tab. I illustrates the following. If the best concrete test scenarios B and C are "re-used" to test system A, the faulty behavior of system A is not revealed. In other words: By naïvely reusing test scenarios that have been created, or recorded, for other system versions or variants, the faulty behavior of a system might not be revealed. In this example, even the best concrete test scenario for another system is not sufficient, let alone scenarios of lower quality. Note that for this observation, the provenance of these concrete scenarios does not matter, e.g. they could have originated from test drive recordings or "test catalogs" of concrete test scenarios, showing the same result in terms of re-usability. This emphasizes the need for the identification of system-specific concrete test scenarios, since re-used test cases may not provide a strong foundation for a safety argument for the release of an automated or autonomous driving system.

Our argument does *not* imply that *no test case can ever be re-used*. Our experiments show, instead, that *there exist new system variants* for which the quality of existing test cases is *provably bad*. Our conclusion is hence that *in general*, the quality of test cases considered for re-use is *unknown*. If there are external means to ensure that the quality of existing test cases continues to be good, for instance because it is known that the driving behavior of the system under test has not been modified, then these tests may indeed be re-used. It is then the responsibility of the test engineer to argue that the quality of tests for an earlier version of a system carries over to a newer version.

V. RELATED WORK

Several existing works are based on the idea of extracting concrete test scenarios from data according to a variety of different selection and filtering criteria. This data is collected with real test drives [5], [19], [22] or simulation setups [29]. Similar to manual test case generation based on experience, the result of those approaches are "test catalogs" of concrete test scenarios. These are subject to the presented re-usability issue. Additionally, the extracted scenarios are randomly encountered scenarios, which is problematic for safety argumentations, as such are hardly possible based on randomly encountered scenarios. An infeasible amount of driving hours or driven kilometer is necessary for each version of the system [33], [15] for each version and variant of the system.

A very popular idea to circumvent this issue is to extract all concrete test scenarios that occur in real data and filter for the "critical" ones, where a multitude of distinct metrics of criticality exist [14], [20], [32]. The resulting "test catalogs" are relatively small as they contain only the critical scenarios. When the amount of data is huge, there is the hope that the resulting "test catalog" overcomes the randomness of encountering certain concrete scenarios in real traffic. However, it is still subject to the re-usability issue. The critical concrete test scenarios are critical with respect to the behavior of the recording test vehicle or the driver maneuvering the recording vehicle. When such concrete test scenarios are re-used for testing another vehicle, e.g. the next generation automated or autonomous vehicle, it is not guaranteed that the concrete test scenarios are still critical. It might be an easy non-critical test scenario for the new system. Note that such criticality metrics might be used for targeted generation of concrete test scenario, e.g. as fitness function for the application of search-based techniques [10]. In this case, the usage of such criticality metrics is not an instance of the re-usability issue.

A variety of existing works suggest the use of search-based techniques for the selection of concrete test scenarios. We followed this idea to generate system-specific concrete test scenarios. Some of those works focus on the technical aspects of the scenario search [1], [2], [3], [6], some describe fitness functions for different specific purposes [7], [8], [16], and another one provides fitness function templates for testing against safe operating envelopes [12]. Even though these works implicitly advocate the generation of system-specific concrete test scenarios, neither of them discusses the re-usability issue or provides a numerical counterexample to the re-usability.

VI. CONCLUSION

We started by sketching that re-using concrete test scenarios for the testing of automated and autonomous driving systems in general is problematic, since the quality of a test case is system-dependent and may become bad when the system changes. However, many approaches in industry and literature rely on this re-usability. We provided a numerical counterexample to the re-usability of concrete test scenarios. We used standard techniques to generate concrete test scenarios for three different configurations of an exemplary system. As scenario type, we chose an easy lane change. The experimental results showed that even the best concrete test scenario for one system configuration may not be a "good" concrete test scenario for the other. Note that even for a simple lane change and configurations of the identical system design, the re-usability issue could be shown. Systems with different designs (e.g. systems from different vendors) differ much more than configurations of a single design. Thus, re-usability may even be worse. While for this lane change scenario the re-used concrete test scenarios at least are of the correct form, i.e. the ego vehicle performs a lane change

behind the other car, for more complex scenario types, re-used scenarios might not even be of the correct form anymore. In the case of the introductory example of Fig. 1, almost certainly the lane changes will not be performed into the gap anymore as different systems decide to do their lane changes at different moments in time. In terms of re-usability, the provenance of the concrete test scenarios does not matter, e.g. they could be recorded test cases or reference tests from "test catalogs."

We have argued that this shows that the quality of existing tests *in general* cannot be predicted for new versions of a system. While this shows that there cannot be "canonical" reference tests for a product line *in general*, this does *not* mean that the quality of existing test cases *never* carries over to new versions of the system. There may well be situations where an engineer can argue that the re-use of tests is justified.

As a direct consequence, regression testing of automated and autonomous driving systems needs to be reviewed, and new methodologies are necessary. As a solution, we recommend the re-use of logical scenarios instead of concrete test scenarios. Then, for a new system, system-specific concrete test scenarios are generated for each logical scenario instead of re-using concrete test scenarios—which is the automated technique we used to generate all test cases in this paper. The generation of tests obviously is more costly than simple re-use, but our arguments suggest that this cost cannot be avoided unless one can argue that tests for earlier versions are safe to re-use. In addition to using recorded scenarios directly as test cases when this can be explicitly justified, we deem these recorded scenarios useful for the derivation of logical scenarios for test case generation, e.g. as done in [18], [34].

REFERENCES

[1] R. B. Abdessalem, S. Nejati, L. Briand, and T. Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *International Conference on Software Engineering*. ACM, 2018.

[2] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 63–74, 2016.

[3] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *ACM/IEEE International Conference on Automated Software Engineering*, pages 143–154, 2018.

[4] M. Aeberhard et al. Experience, results and lessons learned from automated driving on germany's highways. *IEEE Intelligent Transportation Systems Magazine*, 7(1):42–57, 2015.

[5] J. Bach, M. Holzäpfel, S. Otten, and E. Sax. Reactive-replay approach for verification and validation of closed-loop control systems in early development. Technical report, SAE Technical Paper, 2017.

[6] H. Beglerovic, M. Stolz, and M. Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *IEEE 2017 Intelligent Transportation Systems Conference*, pages 1–6, 2017.

[7] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Generating avoidable collision scenarios for testing autonomous driving systems. In *IEEE Int. Conf. on Software Testing, Verification and Validation*, 2020. to appear.

[8] A. Calo, P. Arcaini, S. Ali, F. Hauer, and I. Fuyuki. Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems. In *Genetic and Evolutionary Computation Conference*, 2020. to appear.

[9] R. Feldt and S. Poulding. Broadening the search in search-based software testing: It need not be evolutionary. In *IEEE/ACM International Workshop on Search-Based Software Testing*, pages 1–7, 2015.

[10] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu. Testing scenario library generation for connected and automated vehicles, part i: Methodology. *arXiv:1905.03419*, 2019.

[11] F. Hauer, I. Gerostathopoulos, T. Schmidt, and A. Pretschner. Clustering traffic scenarios using mental models as little as possible. In *IEEE Intelligent Vehicles Symposium (IV)*, page to appear, 2020.

[12] F. Hauer, A. Pretschner, and B. Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 69–84, 2019.

[13] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *IEEE Intelligent Transportation Systems Conference*, pages 2950–2955, 2019.

[14] P. Junietz, F. Bonakdar, B. Klamann, and H. Winner. Criticality metric for the safety validation of automated driving using model predictive trajectory optimization. In *IEEE Intelligent Transportation Systems Conference*, pages 60–65, 2018.

[15] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[16] M. Klischat and M. Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *IEEE Intelligent Vehicles Symposium*, 2019.

[17] P. Koopman and M. Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[18] F. Kruber, J. Wurst, and M. Botsch. An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *IEEE Intelligent Transportation Systems Conference*, pages 2811–2818, 2018.

[19] U. Lages, M. Spencer, and R. Katz. Automatic scenario generation based on laserscanner reference data and advanced offline processing. In *IEEE Intelligent Vehicles Symposium Workshops*, pages 146–148, 2013.

[20] M. Lehmann, M. Bäumler, G. Prokop, and D. Hamelow. Use of a criticality metric for assessment of critical traffic situations as part of sepia. In *19. Internationales Stuttgarter Symposium*, pages 1154–1167. Springer, 2019.

[21] T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles. *arXiv:1801.08598*, 2018.

[22] P. Minnerup, T. Kessler, and A. Knoll. Collecting simulation scenarios by analyzing physical test drives. In *IEEE Intelligent Transportation Systems Conference*, pages 2915–2920, 2015.

[23] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson. Lane change maneuvers for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1087–1096, 2017.

[24] D. Nister, H.-L. Lee, J. Ng, and Y. Wang. The safety force field. `https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/the-safety-force-field.pdf`, retrieved 22nd January 2020.

[25] A. Pretschner. Defect-based testing. In: Dependable Software Systems Engineering, 2015.

[26] A. Rizaldi et al. Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In *International Conference on Integrated Formal Methods*, pages 50–66. Springer, 2017.

[27] SAE. Definitions for terms related to on-road motor vehicle automated driving systems. *J3016, SAE International Standard*, 2014.

[28] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv:1708.06374*, 2019.

[29] C. Sippl et al. From simulation data to test cases for fully automated driving and adas. In *IFIP International Conference on Testing Software and Systems*, pages 191–206. Springer, 2016.

[30] S. Ulbrich et al. Testing and validating tactical lane change behavior planning for automated driving. In *Automated Driving*, pages 451–471. Springer, 2017.

[31] U.S. National Highway Traffic Safety Administration. A framework for automated driving systems. `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/`

13882-automateddrivingsystems_092618_v1a_tag.pdf, retrieved 22nd January 2020.

[32] W. Wachenfeld, P. Junietz, R. Wenzel, and H. Winner. The worst-time-to-collision metric for situation identification. In *IEEE Intelligent Vehicles Symposium*, pages 729–734, 2016.

[33] W. Wachenfeld and H. Winner. The release of autonomous vehicles. In *Autonomous Driving*, pages 425–449. Springer, 2016.

[34] J. Zhou and L. del Re. Identification of critical cases of adas safety by fot based parameterization of a catalogue. In *Control Conference (ASCC), 2017 11th Asian*, pages 453–458. IEEE, 2017.