ICST Tool Competition 2026 - SDC Testing Track

Prakash Aryan
University of Bern
Bern, Switzerland

Christian Birchler
University of Bern
Bern, Switzerland

Tommaso Fulcini
Politecnico di Torino
Turin, Italy

Luigi Libero Lucio Starace
Università di Napoli Federico II
Naples, Italy

Sebastiano Panichella
University of Bern &
AI4I - The Italian Institute of
Artificial Intelligence for Industry
Bern, Switzerland

***Abstract*—This paper reports on the ICST 2026 edition of the tool competition on regression testing of self-driving cars (SDCs). The competition aims to foster research on test prioritization for simulation-based SDC testing, a rapidly growing and practically relevant domain, by providing a common platform for the submission and evaluation of testing tools.**

To facilitate participation, the competition supplies an advanced infrastructure together with representative case studies, enabling participants to develop and assess test generation, selection, and prioritization approaches for SDCs. In its second edition, the competition features four submitted tools, evaluated using regression metrics for test prioritization (in the previous edition, a selection strategy was the target) and compared with baseline approaches, namely a random strategy, and the tools included in the SBFT edition of the same challenge. The paper presents an overview of the competition, including its motivation, framework, evaluation process, participating tools, and key findings.

***Index Terms*—Software engineering, Software testing, Regression testing, Autonomous systems, Simulation-based testing, Cyber-physical systems**

I. Introduction

Testing cyber-physical systems (CPSs) [1], [2], and in particular self-driving cars (SDCs), introduces challenges [3] that go beyond those encountered in conventional unit and integration testing [4]–[6]. A central aspect is the heavy reliance on simulation-based evaluation, where individual test executions are often time-consuming, computationally expensive, and affected by inherent non-determinism [4], [7]. Moreover, the well-known *reality gap*, i.e., the mismatch between simulated behavior and real-world system performance, further complicates the assessment of testing outcomes [8], [9]. These factors make the execution of large test suites costly, both in terms of runtime and required infrastructure [8], [10], [11]. Consequently, there is a strong need for techniques that improve the efficiency of testing processes by extracting more value from a limited number of simulation runs.

The self-driving car (SDC) Testing Competition aims to foster such research by providing a shared experimental setting for the evaluation of automated testing techniques. This edition represents a continuation of a recent line of competitions, following the SBFT Tool Competition [12], and serves as a second iteration that further explores the same problem space with updated tools and participants. By doing so, it supports comparative assessment and encourages incremental advances in testing methodologies for autonomous driving systems, particularly those relying on vision-based navigation.

Within this framework, the 2026 edition of the competition concentrates on the problem of *test prioritization* [13], namely determining an execution order of test cases that increases the likelihood of detecting faults early during testing. In simulation-driven SDC testing, where each test execution can be expensive, prioritization becomes crucial: by executing more informative tests earlier, it is possible to uncover faults sooner and make better use of constrained computational budgets.

II. Methodology

To support participation, we publicly released on GitHub the competition infrastructure, the interface specification, a sample benchmark, and a random prioritization baseline. Participants could use these resources to develop and locally assess their tools before submission. Tool implementations were submitted as pull requests and then evaluated by the organizers on a benchmark that was not disclosed to participants. This setup ensured a fair comparison while providing an open-source and extensible platform for tool development[1].

A. Competition Platform

1) Interface: For this edition, participants were required to implement a standardized interface to interact with the Competition Platform. The interface covers the full lifecycle of a tool, including an optional initialization step and the generation of a complete prioritization of the test suite.

As in previous editions, the interface is formally specified using *Protocol Buffers*[2], which provides a language-neutral and extensible definition of data structures and service endpoints. Communication between the Competition Platform and

[1]https://github.com/christianbirchler-org/sdc-testing-competition
[2]https://protobuf.dev/

TABLE I
OVERVIEW OF PARTICIPATING TOOLS.

Tool	Overview
KSERESNET [14]	Learning-based approach that transforms road waypoint sequences into kinematic derivative features and uses an SE-attention 1D-ResNet to rank test cases.
RoadFury [15]	Transformer-based approach leveraging a 10-channel feature representation of test scenarios. Ranks test cases using a Pre-LLN Transformer Encoder with stochastic weight averaging.
EagleSemble [16]	Ensemble-based approach that extracts geometric road features from interpolated road layouts and combines the predictions of multiple models to rank test cases.
RTE4SDC [17]	Transformer-based approach that uses a pre-trained Ranking Transformer Encoder model to rank self-driving car test cases for regression testing.

the submitted tools is implemented via *gRPC*, enabling participants to use different programming languages and technology stacks.

2) Docker: To lower the entry barrier, we provided a reference implementation of the interface together with an initial dataset. The sample tool implements a simple random test prioritization strategy and serves both as an example of how to use the platform and as a baseline for comparison.

3) Sample Tool & Data: To facilitate participation and lower the entry barrier, we provided a reference implementation of the interface together with an initial dataset. The sample tool implements a simple random test prioritization strategy and serves both as a usage example and as a baseline for comparison in the evaluation.

B. Competition Tools

We received four competing tools, as reported in Table I. For the evaluation, we additionally considered two baseline approaches: the random baseline and ITEP4SDC [18], the tool submitted in the SBFT edition [12]. We built Docker images for all evaluated tools targeting the x86_64 architecture and made them publicly available in the GitHub Container Registry[3].

C. Benchmark Test Suites

Our evaluation relies on pre-executed test cases from the SensoDat dataset [7], which was developed in prior work on simulation-based testing of self-driving cars [19]–[22]. SensoDat contains 36 collections of test cases. Each collection groups tests generated and executed under the same conditions, including the test generator, the driving AI configuration, and the oracle definition. We treat each collection as one benchmark test suite, yielding a total of 36 suites.

The benchmark suites were generated using three test generators: Ambiegen [23], Frenetic [24], and FreneticV [25]. Overall, the benchmark comprises 32,580 pre-executed test cases. Table II summarizes the distribution of benchmark suites and test cases across the three generators.

[3]https://github.com/orgs/christianbirchler-org/packages?repo_name=sdc-testing-competition

TABLE II
SUMMARY OF BENCHMARK TEST SUITES IN SENSODAT [7].

Generator	# Suites	# Test Cases	Avg. Suite Size
Ambiegen	13	12,496	961.2
Frenetic	13	12,135	933.5
FreneticV	10	7,949	794.9
Total	**36**	**32,580**	**905.0**

D. Evaluation Metrics and Procedure

In this edition, the competition task shifts from test *selection* to test *prioritization*. Accordingly, the evaluation focuses on metrics that assess how effectively and efficiently a tool orders a complete test suite with respect to fault revelation.

1) Initialization Time: Before prioritization, each tool receives auxiliary training data that may be used to build internal models or preprocess information. Initialization time measures the duration of this phase.

2) Prioritization Time: After initialization, each tool must produce a complete ordering of the given test suite. Prioritization time measures only the time required to generate this ordering, excluding initialization.

3) Average Percentage of Faults Detected (APFD): APFD is a standard metric for evaluating test prioritization techniques [26]. It measures how quickly failing tests appear in the prioritized order. Formally: $\mathrm{APFD} = 1 - \frac{\sum_{i=1}^{m} TF_i}{n \cdot m} + \frac{1}{2n}$, where n denotes the number of test cases, m the number of failing test cases, and TF_i the position of the i-th failing test in the prioritized suite. Higher APFD values indicate more effective prioritization.

4) Cost-Aware APFD (APFD_C): In simulation-based testing of self-driving cars, test cases may have substantially different execution costs, for example due to different route lengths or vehicle dynamics. As a result, two prioritizations with the same APFD may differ in cost-effectiveness. APFD_C extends APFD by incorporating the execution cost of each test case, which in our setting corresponds to simulation time.

Let C_j denote the execution cost of the j-th test in the prioritized order, $CF_i = \sum_{j=1}^{i} C_j$ the cumulative cost up to the i-th failing test, and $C_{\text{total}} = \sum_{j=1}^{n} C_j$ the total execution cost of the test suite. Then: $\mathrm{APFD}_C = 1 - \frac{\sum_{i=1}^{m} CF_i}{C_{\text{total}} \cdot m} + \frac{1}{2m}$. Higher values indicate that failures are revealed earlier with respect to cumulative execution cost.

5) Time to First Fault: Time to first fault measures the cumulative simulation time required to encounter the first failing test case in the prioritized order. Lower values indicate that fault-revealing tests are ranked earlier.

6) Time to Last Fault: Time to last fault measures the cumulative simulation time required to reach the last failing test case in the prioritized order. This metric captures how fault-revealing tests are distributed across the full ranking.

E. Experimental Setup

All experiments were conducted on a virtual machine equipped with an NVIDIA RTX A6000 GPU. To support

GPU-enabled tools, we used the NVIDIA Container Runtime extension for Docker.

F. Procedure

Unlike previous editions, we did not treat each SensoDat testing campaign as a fixed evaluation sample, as this would make the benchmark structure predictable. Instead, we constructed samples by randomly drawing test cases from the full SensoDat dataset, thereby creating test suites from a broader pool of available tests.

Each experiment uses one sample composed of multiple subjects, where each subject corresponds to a test suite to be prioritized. Sample construction is governed by two parameters: (i) the sample size, that is, the number of subjects in the sample, and (ii) the subject size, that is, the number of test cases contained in each subject. We sampled 20 sample sizes and 20 subject sizes. In both cases, the sampled values ranged from 10 to 200, with increments of 10.

Before prioritizing the sample, each tool was given one subject for initialization. The remaining subjects in the sample were then prioritized and evaluated.

G. Data Collection & Analysis

The evaluator provides each tool with unsorted test suites and collects the resulting prioritized suites. Based on these prioritized suites, it computes the metrics described in Section II-D for each test suite.

For the final analysis, we record summary statistics over each sample, namely the minimum, maximum, mean, and standard deviation of the measured metrics, and store them in a dedicated database. This database will be made publicly available to support further analysis.

III. Experiments and Results

We ran each tool on multiple samples under different configurations, varying both sample and subject sizes. Overall, each tool was evaluated on 400 samples. In Table III, we report the statistics of the samples after applying the participants' tools. For comparison, we also include the results of two baseline approaches: the random baseline tool provided in the competition platform and ITEP4SDC [18], the tool submitted in the SBFT edition [12]. The reported statistics consider only successful treatments, i.e., cases in which the tool returned a valid prioritization of the full test suite.

The results indicate that all four submitted tools outperform the random baseline in terms of mean APFD. RoadFury [15] obtains the highest mean APFD (0.80), followed by EagleSemble [16] (0.78), RTE4SDC [17] (0.77), and KSERESNET [14] (0.62). Regarding mean $APFD_C$, RoadFury and EagleSemble both reach 0.83, which is comparable to ITEP4SDC, while RTE4SDC obtains 0.82. KSERESNET, while being the fastest tool in terms of prioritization time (0.08 s on average), yields the lowest effectiveness among the submissions.

It is worth noting that EagleSemble exhibits a reliability issue similar to that of ITEP4SDC: out of 3,600 treatments, only 521 (14.5%) resulted in a valid prioritization. This is comparable to ITEP4SDC, which succeeded on 5,941 out of 41,600 treatments (14.3%) in the SBFT evaluation. In contrast, the remaining three tools—RoadFury, RTE4SDC, and KSERESNET—achieved a 100% success rate across all treatments.

TABLE III
Average Performance Metrics Statistics (successful treatments only)

Tool	Statistic	Metrics				
		apfd	apfdc	t_prioritize_tests	t_first_fault	t_last_fault
RoadFury	max	0.84	0.90	0.43	85.44	5,005.27
	mean	**0.80**	**0.83**	0.42	**52.21**	**3,273.49**
	std	0.03	0.04	0.01	27.03	1,067.11
	min	0.75	0.78	0.41	11.45	1,946.21
EagleSemble	max	0.81	0.88	1.02	60.40	2,246.57
	mean	**0.78**	**0.83**	0.97	45.99	2,053.17
	std	0.04	0.06	0.09	21.80	316.75
	min	0.75	0.79	0.92	30.90	1,872.13
RTE4SDC	max	0.82	0.88	0.47	82.83	5,067.84
	mean	**0.77**	**0.82**	0.40	47.73	3,803.70
	std	0.03	0.04	0.04	27.55	761.18
	min	0.73	0.75	0.33	8.93	2,751.26
KSERESNET	max	0.69	0.73	0.09	193.62	6,188.57
	mean	0.62	0.65	**0.08**	77.83	5,593.73
	std	0.04	0.05	0.00	59.71	414.17
	min	0.56	0.57	0.08	13.29	4,909.35
ITEP4SDC	max	0.84	0.92	0.80	87.90	3,706.99
	mean	**0.79**	**0.83**	0.78	**54.77**	**3,016.25**
	std	0.03	0.05	0.02	28.31	553.80
	min	0.74	0.76	0.76	22.69	2,451.42
Random Baseline	max	0.62	0.67	0.52	668.30	6,613.83
	mean	0.50	0.52	0.46	145.75	5,781.33
	std	0.05	0.06	0.03	132.02	349.89
	min	0.38	0.39	0.37	6.65	4,898.84

IV. Discussion & Conclusion

The results of this edition highlight several findings. First, three of the four submitted tools—RoadFury, RTE4SDC, and KSERESNET—demonstrate full reliability, successfully prioritizing all test suites across all experimental configurations. This stands in contrast to both ITEP4SDC and EagleSemble, which failed to produce valid prioritizations for the majority of test suites (approximately 86% failure rate in both cases). The evaluation results reported in Table III consider only those treatments that succeeded; the high failure rates of these two tools limit their practical applicability and warrant further investigation.

Second, in terms of prioritization effectiveness, RoadFury and RTE4SDC consistently outperform the random baseline by a substantial margin, with mean APFD values of 0.80 and 0.77, respectively. EagleSemble achieves a comparable mean APFD (0.78) on successful treatments, but its low success rate reduces its overall utility. KSERESNET, while fully reliable, obtains a lower mean APFD of 0.62, indicating that its prioritization strategy, though better than random (0.50), is less effective than the other submitted approaches.

Third, the time-to-first-fault metric reveals that RTE4SDC and EagleSemble tend to surface faults slightly earlier (47.73 s and 45.99 s, respectively) than RoadFury (52.21 s), while KSERESNET requires notably more simulation time before the first fault is encountered (77.83 s).

The SDC Testing Competition, co-organized with ICST and SBFT 2026, reached its second edition in 2026. In this edition, we introduced (regression) metrics for *test prioritization* and evaluated them against a random baseline approach. Future

editions will incorporate additional metrics [27] and broaden the scope to include a wider range of obstacle types, such as trees and buildings, as well as environmental factors.

V. Data Availability

The challenge pipeline, submitted tools, and evaluation data are publicly available on GitHub: https://github.com/christianbirchler-org/sdc-testing-competition.

Acknowledgement

We thank the participants of the competition for their invaluable contribution. We thank the Horizon 2020 (EU Commission) support for the project InnoGuard, Marie Sklodowska-Curie Actions Doctoral Networks (HORIZON-MSCA-2023-DN), and the SNSF for the project entitled "SwarmOps: Human-sensing based MLOps for Collaborative Cyber-physical systems" (Project No. 200021_219732). We also acknowledge that this work was partially supported by the Italian PNRR MUR project PE0000013-FAIR (Future AI Research). Furthermore, we also thank CHOOSE, the Swiss Group for Original and Outside-the-box Software Engineering, for their financial support.

References

[1] S. Khatiri, F. M. Amin, S. Panichella, and P. Tonella, "When uncertainty leads to unsafety: Empirical insights into the role of uncertainty in unmanned aerial vehicle safety," *Empir. Softw. Eng.*, vol. 30, no. 6, p. 166, 2025. [Online]. Available: https://doi.org/10.1007/s10664-025-10697-z

[2] S. Panichella, S. Khatiri, and C. Birchler, *Deployment for Cyber-Physical Systems: The Relations Between the Testing and Monitoring DevOps Phases*. Singapore: Springer Nature Singapore, 2026, pp. 103–141. [Online]. Available: https://doi.org/10.1007/978-981-95-1786-2_6

[3] S. Panichella, P. Arcaini, M. B. Cohen, and A. Arrieta, Eds., *Roadmap for DevOps in Cyber-Physical Systems*, ser. Communications of Shonan Meetings. Springer Singapore, 2026. [Online]. Available: https://doi.org/10.1007/978-981-95-1786-2

[4] C. Birchler, S. Khatiri, P. Rani, T. Kehrer, and S. Panichella, "A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, pp. 152:1–152:9, 2025. [Online]. Available: https://doi.org/10.1145/3711906

[5] F. Zampetti, R. Kapur, M. D. Penta, and S. Panichella, "An empirical characterization of software bugs in open-source cyber-physical systems," *J. Syst. Softw.*, vol. 192, p. 111425, 2022. [Online]. Available: https://doi.org/10.1016/j.jss.2022.111425

[6] V. Crespo-Rodriguez, C. Birchler, Neelofar, A. Aleti, and S. Panichella, "The role of road features and vehicle dynamics in cost-effective autonomous vehicles safety testing: Insights from instance space analysis," *Information and Software Technology*, vol. 195, p. 108123, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584926001126

[7] C. Birchler, C. Rohrbach, T. Kehrer, and S. Panichella, "Sensodat: Simulation-based sensor dataset of self-driving cars," in *21st IEEE/ACM International Conference on Mining Software Repositories, MSR 2024, Lisbon, Portugal, April 15-16, 2024*, D. Spinellis, A. Bacchelli, and E. Constantinou, Eds. ACM, 2024, pp. 510–514. [Online]. Available: https://doi.org/10.1145/3643991.3644891

[8] S. Khatiri, S. Panichella, and P. Tonella, "Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights," in *IEEE Conference on Software Testing, Verification and Validation*. IEEE, 2023, pp. 281–292. [Online]. Available: https://doi.org/10.1109/ICST57152.2023.00034

[9] P. Aryan and S. Panichella, "Physics-informed machine learning for precision unmanned aerial vehicle control: Adaptive transformers with safety guarantees," *Eng. Appl. Artif. Intell.*, vol. 172, p. 114379, 2026. [Online]. Available: https://doi.org/10.1016/j.engappai.2026.114379

[10] S. Khatiri, S. Panichella, and P. Tonella, "Simulation-based testing of unmanned aerial vehicles with aerialist," in *International Conference on Software Engineering (ICSE)*, 2024.

[11] S. Khatiri, F. E. V. Barrientos, M. Wulf, P. Tonella, and S. Panichella, "Bridging research and practice in simulation-based testing of industrial robot navigation systems," 2025.

[12] P. Aryan, C. Birchler, T. Fulcini, L. L. L. Starace, and S. Panichella, "SBFT Tool Competition 2026 – CPS-SDC regression testing track," in *2026 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)*. IEEE, 2026.

[13] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Softw. Test. Verification Reliab.*, vol. 22, no. 2, pp. 67–120, 2012. [Online]. Available: https://doi.org/10.1002/stv.430

[14] V. K. Swain, S. Godboley, P. R. Krishna, and A. Das, "KSERESNET at the ICST 2026 Tool Competition – self-driving car testing track," in *2026 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2026.

[15] C.-N. Tran, S. D. M. Dao, T. K. Huynh, P.-H. Pham, and L. P. Q. Nguyen, "RoadFury at the ICST 2026 Tool Competition – self-driving car testing track," in *2026 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2026.

[16] C. Aquilino, E. Ritacco, and V. Riccio, "EagleSemble at the ICST 2026 Tool Competition – self-driving car testing track," in *2026 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2026.

[17] M. Maugeri, P. Lin, and A. Pasikhani, "Rte4sdc at the icst 2026 tool competition – self-driving car testing track," in *2026 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2026.

[18] A. Güllü, F. A. Shah, and D. Pfahl, "ITEP4SDC at the SBFT 2026 Tool Competition – CPS-SDC regression testing track," in *2026 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)*. IEEE, 2026.

[19] C. Birchler, S. Khatiri, P. Derakhshanfar, S. Panichella, and A. Panichella, "Single and multi-objective test cases prioritization for self-driving cars in virtual environments," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2022.

[20] C. Birchler, S. Khatiri, B. Bosshard, A. Gambi, and S. Panichella, "Machine learning-based test selection for simulation-based testing of self-driving cars software," *Empirical Software Engineering*, 2022.

[21] C. Birchler, N. Ganz, S. Khatiri, A. Gambi, and S. Panichella, "Cost-effective simulation-based test selection in self-driving cars software," *Science of Computer Programming (SCP)*, 2022.

[22] ——, "Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor," in *the 29th IEEE International Conference on Software Analysis, Evolution, and Reengineering*, 2022.

[23] D. Humeniuk, G. Antoniol, and F. Khomh, "Ambiegen tool at the SBST 2022 tool competition," in *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022*. IEEE, 2022, pp. 43–46. [Online]. Available: https://doi.org/10.1145/3526072.3527531

[24] E. Castellano, A. Cetinkaya, C. H. Thanh, S. Klikovits, X. Zhang, and P. Arcaini, "Frenetic at the SBST 2021 tool competition," in *14th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2021, Madrid, Spain, May 31, 2021*. IEEE, 2021, pp. 36–37. [Online]. Available: https://doi.org/10.1109/SBST52555.2021.00016

[25] E. Castellano, S. Klikovits, A. Cetinkaya, and P. Arcaini, "Freneticv at the SBST 2022 tool competition," in *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022*. IEEE, 2022, pp. 47–48. [Online]. Available: https://doi.org/10.1145/3526072.3527532

[26] F. Altiero, A. Corazza, S. Di Martino, A. Peron, and L. Libero Lucio Starace, "Regression test prioritization leveraging source code similarity with tree kernels," *Journal of Software: Evolution and Process*, vol. 36, no. 8, p. e2653, 2024.

[27] C. Birchler, T. K. Mohammed, P. Rani, T. Nechita, T. Kehrer, and S. Panichella, "How does simulation-based testing for self-driving cars match human perception?" *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, pp. 929–950, 2024. [Online]. Available: https://doi.org/10.1145/3643768