Springer Journal manuscript No.
(will be inserted by the editor ITNG 2024)

State of Test Optimization for Variability in Industry

Muhammad Abbas[1,2*] · Mehrdad Saadatmand[1] · Eduard Paul Enoiu[2] · Bernd-Holger Schlingloff[3]

Abstract This paper presents an exploratory survey on test optimization practices for variant-intensive systems in the industry, focusing on understanding current challenges and prospects. Testing variant-intensive systems, particularly in domains like automotive, embedded systems, and telecom, involves considerable complexity due to the high number of configurations and variability in hardware and software. The survey responses reveal varying adoption of test categorization, prioritization, and selection techniques across different domains, with some employing feature-based testing approaches and others relying on more ad-hoc methods. Challenges identified include difficulties with change impact analysis, the scalability of test optimization in systems with many configurations, and the ineffective reuse of test cases across product variants. The study highlights the need for automated tools to support test categorization, prioritization, and coverage-based test selection to address these challenges. The findings highlight the importance of developing more scalable solutions tailored to the needs of specific domains for optimizing tests in variant-intensive systems.

Keywords Test Optimization · Variant-Intensive Systems · Test Reuse · Coverage Criteria · Product line engineering · Variability · Test Selection · Test Categorization · Coverage · Survey

1 Introduction

In engineering industrial systems, the testing process introduces complexities far beyond those seen in traditional software testing. Industrial systems must account for an extensive range of configurations, enclosing not only different hardware components but also variant software versions and their integration into a functional system [19,2]. These diverse configurations, combined with the constrained testing infrastructure, present significant challenges for efficient testing. Traditional testing techniques [10] cannot manage the combinatorial explosion of variants required by industrial systems testing. As industrial systems are increasingly deployed in mission-critical domains, ensuring that all possible configurations are tested effectively and efficiently becomes essential.

Several research efforts have highlighted the complexities involved in testing industrial systems due to the variability in hardware and software configurations. For instance, Golagha et al. [21] emphasize the need for test selection techniques that intelligently and automatically determine which system variant to test and where to execute these tests, considering the limitations of hardware testbeds. Variants of industrial systems, whether in hardware tools such as those in controllers for industrial machines, require careful optimization and selection of test cases that can cover the breadth of system behaviors while minimizing the cost and time spent on exhaustive testing [9].

Recent advances in test optimization, particularly in industries such as automotive software development, have shed light on methods to reduce test analysis effort. Golagha et al. [11] present a clustering approach that groups failing test cases based on non-code features to reduce manual analysis time. This kind of failure clustering highlights the importance of optimizing test efforts when multiple configurations are involved, as seen in regression testing environments where large tests may fail due to a few core issues [11,1].

Contributions. While studies focus on test optimization and selection in the presence of variability, it is still unclear if they are standard industry practices. This paper reviews the state of practice in industry in variability management, test

[1]RISE Research Institutes of Sweden, Västerås, Sweden.
[2]Mälardalens University, Västerås, Sweden.
[3]Fraunhofer FOKUS, Berlin, Germany.
*Corresponding author(s). E-mail(s): muhammad.abbas@ri.se
Contributing authors: mehrdad.saadatmand@ri.se; eduard.paul.enoiu@mdu.se; holger.schlingloff@fokus.fraunhofer.de

optimization, and test selection in the presence of variability. The paper further reports perceived practical challenges in the area.

Structure. The rest of the paper is organized as follows. Section 2 introduces related work briefly. Section 3 presents the method, Section 4 presents and discusses the obtained qualitative results. Section 5 presents the potential threats to validity and limitations. Finally, Section 6 concludes the paper with future directions.

2 Related Work

Researchers have addressed some issues of variant selection and test optimization for industrial software systems. How to handle variability testing in product lines has been surveyed by Sinnema et al. [19], Lamancha et al. [14] and Lee et al. [15]. These surveys indicate little work exists on variant handling and selection during software testing compared to other aspects of software development. In recent years, some contributions [7,6,4,8,18] have proposed applications of combinatorial testing concepts for variant and product line handling as well as for variability testing and test optimization.

Engstrom et al. [10] provides a comprehensive view of variability testing and identifies several challenges regarding the testing of variant-intensive software. This survey particularly mentions i) how to handle many tests, ii) how to balance the effort between reusable components and concrete products, and iii) how to manage variability. Additionally, there is a lack of substantial empirical studies on actual industrial systems and the applicability of these optimization techniques for handling variability in industrial scenarios.

Test case selection and prioritization play a critical role in test optimization and regression testing. Previous research has explored various approaches for selecting and prioritizing the most effective test cases, especially when executing the entire test suite is impractical [3]. These approaches typically use data from variants or prior system executions, including fault history, execution time, test coverage, and previous failing tests [17,21]. Yoo and Harman [22] provide a comprehensive review of regression test minimization, selection, and prioritization strategies, underscoring their importance in test optimization. Despite over 40 years of maturing research, the industrial application of these techniques still needs to be improved. Several surveys on regression test selection methods [12,22] highlight that most techniques lack industrial application. As the complexity of software systems increases, particularly with the introduction of variability, there is a growing demand for advanced test optimization techniques that can efficiently handle both the scale and variety of industrial systems.

3 Method

The survey conducted for this study was designed to assess the current state of test optimization practices in variant-intensive systems.

Context and Research Questions: This work can be classified as an *exploratory* questionnaire-based case study that focuses on documenting the current state of practices in effectively testing variant-intensive systems in industry. We define the following research questions (RQs) that guide our data collection for the study.
RQ1: What is the current state-of-practice in test optimization for variant-intensive systems in industry?
This RQ collects general data about the nature of the system under test (SUT), the test suite, and the process for testing variants in the industry.
RQ2: What challenges exist in testing and test optimization for variant-intensive systems?
In this RQ, we aim to gather qualitative insights on the perceived challenges in testing and test optimization of variant-intensive systems.

We collect data for both research questions using a questionnaire containing closed and open-ended questions. For RQ1, we analyze the responses and provide an overview of the results directly based on them. For RQ2, we employ qualitative data analysis (thematic analysis) to identify themes and sub-themes regarding perceived challenges from the reactions.

Our method was based on guidelines for surveys, case studies, and ethical data collection [13,16,20]. The rest of this section details the data collection procedure.

Data collection procedure: As shown in Figure 1, the procedure is divided into three main phases, i.e., Ⓐ, Ⓑ, and Ⓒ, focusing on the preparation of the questionnaire, conducting the survey, and analysis of the collected data, respectively. Below, we provide a brief overview of each phase.

A. Questionnaire: We followed a team-based approach to design and execute the questionnaire-based survey. The team included subject matter experts from industry and researchers involved in a European project. During various meetings with the experts, an initial questionnaire was drafted with open and close-ended questions, as shown in phase Ⓐ of Figure 1. After an internal review, the questions were refined, re-ordered, and simplified. Following the guidelines for survey instrument evaluation [16], we used expert reviews to evaluate our refined questionnaire. The consortium reviewed the refined questionnaire in an online meeting, and based on the feedback, a final questionnaire was obtained. Following this collaborative approach, the survey questions were formulated based on discussions with project stakeholders, including use case providers and technology/research partners, to address key aspects of test management, test reuse, priori-

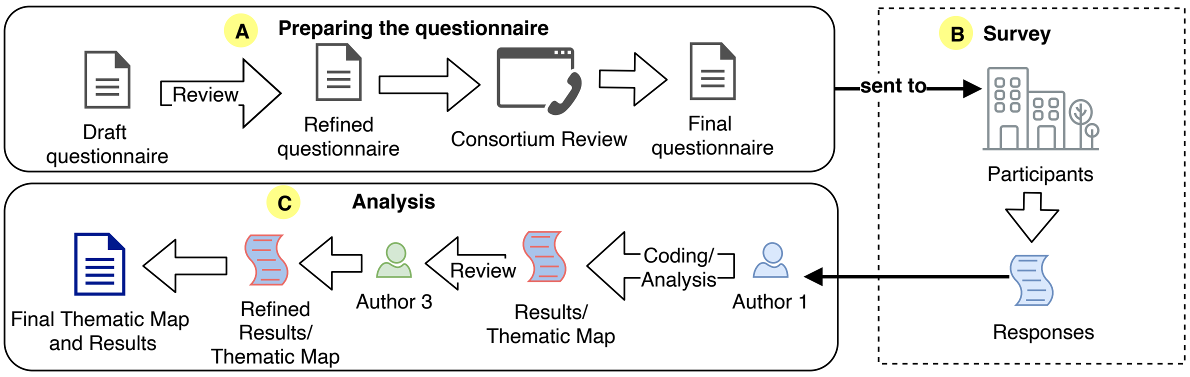

Fig. 1: Data collection and analysis procedure

tization, and optimization for product variants. The questionnaire development process was iterative, involving internal reviews by industry experts and stakeholders within the consortium to ensure the relevance of the questions to practical, real-world testing scenarios.

The final questionnaire included open and close-ended questions covering the following main areas.

1. General questions about the nature of SUT, test suite, variability in the SUT, and the current test process (RQ1).
2. Questions to collect data about the state of test optimization, test optimization for variability, test reuse, and test categorization and prioritization (RQ1).
3. Questions to collect data about perceived challenges and areas of improvement (RQ2).

The first group of questions aims to document the demographics of the SUT, the testing process, and the test suite. The second group focuses on collecting practices in test optimization for variability and related topics that enable test optimization.

B. Survey and participants: After drafting the final questionnaire, we selected representatives from various companies in diverse domains based on convenience. Therefore, we chose representative industrial professionals involved in the European project. Some of these participants also helped review and refine the initial version of the questionnaire. Every participant had a representative role in their company and was distributed across Europe and North America. The survey was distributed to the participants using a collaborative word-processing tool, and they were encouraged to answer all the questions. We also encouraged the participants to add comments while answering the questions they thought could be necessary for the survey.

Ethical considerations: The survey targets representative software development and testing professionals to gather data about the current state of practices in test optimization for variant-intensive systems. Via a European project, we accessed seven representative companies from which we selected one participant each. However, participation in the survey was strictly voluntary. Furthermore, we ensured that the data collected during the survey was only used to draw general conclusions about the topic and that the confidentiality of the data and anonymity of the participants were respected.

C. Analysis and interpretation: The collected data is analyzed, reported, and discussed in Section 4. As is typical for surveys, we report and discuss the participants' responses to close-ended questions, supporting the discussions with statistics extracted from the responses. On the other hand, for open-ended questions, we use qualitative content analysis primarily to document and report high-level themes about the topic and the perceived challenges. In particular, the responses were analyzed, and open-ended questions were coded by Author 1 (as shown in step Ⓒ of Figure 1), and a document reporting the results and thematic map was produced. Author 3 of this paper reviewed and refined the results and thematic map, producing a refined document for the results with a thematic map. Finally, all authors reviewed the analysis and the thematic map reported in this paper.

Table 1: Background on representative companies

Domain	# of Part.	Region	Nature
Automotive	2	Germany, Canada	Safety/Security-critical (1/2)
Embedded Observability	1	Sweden	N/A
Production & Manufacturing	1	Germany	Safety-critical
Railway control	1	Sweden	Safety-critical
Telecom	1	Türkiye	N/A
TV OS	1	Türkiye	N/A

4 Results and Discussions

In this section, we present and discuss the results of the analysis and interpretation of the responses. Where needed, we also support the results with supporting quotes— rewritten for better readability—from the responses presented in *"italic text"* for emphasis.

4.1 Background on SUT and Test suite (RQ1):

Table 1 summarizes the background of the representative companies and their domains. As seen in Table 1, the participants ranged from various domains, such as automotive, embedded software observability, and TV OS, to name a few. The `# of Part.` column shows the total number of participants from the domain and the region column shows the country of operation of the representative company. Finally, the `Nature` column shows the nature of the SUT under consideration for the survey. One of the representative companies' products for the automotive domain is safety and security-critical. The other automotive SUT (from Germany) provides supporting software to enable the engineering of automotive systems and is, therefore, not a safety or security-critical product. Finally, the SUTs in production, manufacturing, and the railway domain are classified as safety-critical.

The questionnaire also covered data collection for existing test suites for the products. When asked if the SUT has an existing test suite and its size, all the respondents highlighted that they have a test suite at the product level. The size of the test suite varies across all domains. In the automotive domain, one respondent exemplified infotainment systems with a test suite that captures different parts of the road map data; thus, the test suite captures around 220000 miles of road. In addition, radar and stereo camera data are also used for testing, which results in additional tests. In the railway domain, the system contains over 180 sub-components, each of which is tested with at least four to five test scenarios. In addition, there are 20 test scenarios in the railway domain at the integration level. A respondent from the telecom domain highlighted that their product has a test suite of 1500 test scenarios for their product, where each test scenario might contain multiple test cases. The TV OS domain has a test suite of 101K test scenarios, 100K requiring manual execution. The respondents from embedded observability and production domains did not specify the size of their test suites.

4.2 Variability, testing and test optimization practices (RQ1):

The survey responses provided valuable insights into the current practices in testing variant-intensive systems across different domains. The results revealed significant variability in the adoption of test optimization techniques. While some domains, such as automotive and railway, have well-established processes for feature-based testing, others use specific practices due to the nature of their systems.

Feature-Based Testing and Reuse: A key finding was the widespread use of feature-based testing in safety-critical domains, where testing is closely tied to documented features and system requirements. Automotive and railway domains, in particular, showed a reliance on structured requirements and models to guide their testing efforts. However, the survey also revealed inconsistent test reuse across variants, with some domains successfully reusing test cases. In contrast, others, such as embedded observability, struggle due to the high variability of their platforms. Test reuse is often not feasible in these cases due to the dynamic configuration changes introduced at runtime.

As most of the SUTs have to work with varying hardware configurations and regional regulations, we also asked questions to collect data about the variability in the systems and how they are handled. We asked how new features

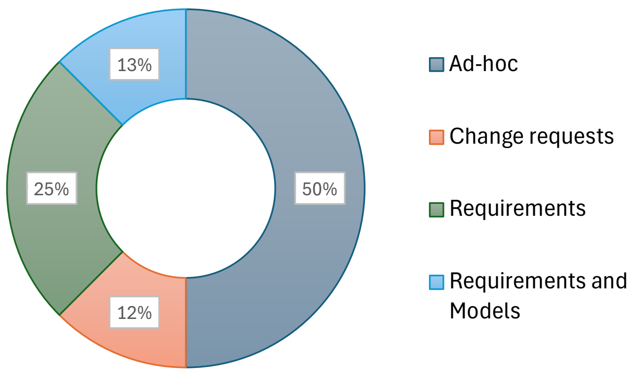

Fig. 2: New feature introduction process

are documented and introduced to the systems, if they are modeled, and how. In addition, we asked if existing artifacts such as requirements, tests and source code is being reused when deriving new variants.. As shown in Figure 2, ad-hoc approaches for documenting features are widespread. The most representative respondents of the domains define an ad hoc process that can vary. They can use requirements, feature requests, or other documentation sources when introducing new features or modeling variability. Following ad-hoc feature modeling and documentation, requirements are also a primary source for introducing and documenting new features. In some cases (13%), requirements are combined with models to document new features and their variability. Finally, change requests are more frequently used in the automotive domain when new features are introduced. For artifact reuse, except the embedded observability domain, all other domains reuse existing artifacts when deriving new variants. The representative expert from the embedded observability domain cited that their SUT is one big platform that has to work with varying hardware configurations, often at run-time. Due to the nature of the SUT, they do not derive their systems' variants, making reuse irrelevant.

Test Prioritization and Selection. Classifying the test cases into different categories might be necessary to effectively apply test selection for test optimization. For example, one classification would be whether or not a test case is testing a particular product configuration. Such categorizations and classifications improve the test selection process and, in turn, enable test optimization. In addition, coverage criteria for test cases also play a crucial role in test selection. Therefore, we asked our respondents if they categorize test cases. We also asked about the categorization scheme used, whether they keep track of the variants and features the test cases are testing, and whether they use coverage criteria.

Five of the seven representative companies from their domains categorize test cases based on various classification schemes. In addition, another company plans to introduce test categorization into their process. Only one of the seven companies does not have test categorization in place and has no plans for it in the near future. When asked what classification schemes are being used for test categorization, a respondent from the automotive domain mentioned that tests are classified based on their perceived difficulty levels. Respondents from the embedded observability and telecom domain use functional and various non-functional quality labels to categorize test cases. In the production and manufacturing domain, the mode of operation of the tests is used to categorize them (e.g., automatic or human input required). Finally, the respondent in the TV domain responded that the company uses a classification scheme unknown to the respondent.

We also asked if separate test suites exist for the variants and the standard product. We found that only the automotive and railway domains have separate test suites for the standard product and the derived variants. When asked if the test cases could be traced to the standard product and its product variants, we found that all domains except for two (production and one representative company from the automotive domain) have mechanisms to trace test cases to variants and standard products.

When new features are introduced or changes are made to the system under development, regression testing is common. In such cases, redundant and repeated testing of shared features can make the testing process inefficient. Furthermore, not all the tests in a suite are the same in effectiveness and might need ranking, prioritization, and selection. In addition, the execution of all test suites might not be possible in some cases. For example, one billion test cases may be needed to test all the possible variants of the product in the embedded observability domain. This is due to the varying configurations that could be enabled at compile time. *"A single test program, where 10-20 sanity checks are applied on the output. This, however, needs to be executed on every combination of 30-50 static build flags that affect the code in undocumented ways, meaning at least 2^30 = 1 billion tests if using a 'brute force' approach with no test case prioritization."* Different test criteria are used to select a subset of tests from the test suite in such cases. We asked if the shared code and features are tested across different variants and versions. In addition, we asked for test optimization enablers. In particular, we asked if they order, prioritize, or categorize tests based on some criteria before execution in regression scenarios.

Four of the seven representative companies from their domains order test cases. In the automotive domain, the representative companies do not order test cases for execution, but one representative company does select sub-sets based on the test suite size. For example, due to time constraints, they might select a subset of map data from a large test suite to test the navigation system. In the embedded observability domain, engineers could order the test randomly or systematically based on exploring various build flags. In addition, they only test the critical sub-set of build configurations—identified manually— and select a sub-set from the test suite that targets the most critical build configurations.

On the other hand, all tests are executed at least once in the representative company from the production and manufacturing domain. In the rail domain, the test cases are independent and, thus, do not require any ordering. However, test selection is performed to target the changes in the code. Nevertheless, due to the safety-critical nature of the product, the entire test suite is once again executed before the release of the software. In the telecom domain, test cases are ordered, but no test selection is made, and all the tests are executed. Finally, test cases might depend on each other in the TV domain, and thus, ordering is required based on their dependency. Executing the entire test suite is preferred, but it is sometimes not possible, and an ordered subset is selected to meet time constraints.

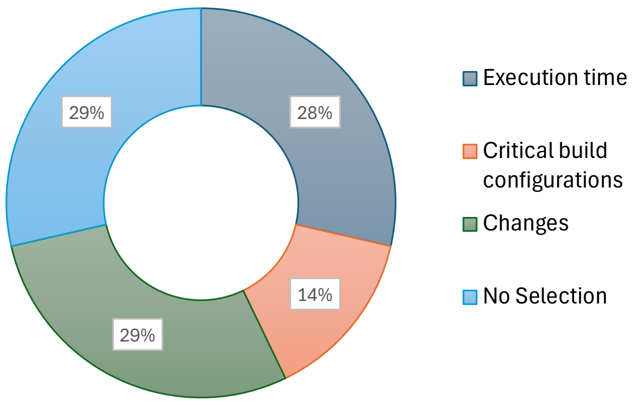

Fig. 3: Test selection criteria

In summary, four representative companies (57%) do not order the test cases for execution. The ones that do order tests are often based on test dependencies. Also, five of the representative companies (70%) have some test selection in place with varying selection criteria, shown in Figure 3. The automotive and TV domains select tests that could finish execution in a pre-defined time window. The representative company from the embedded observability domain identifies critical build configurations based on experience and selects a subset of tests targeting those configurations. Finally, the telecom and railway domains select a subset of tests targeting changes.

Use of Coverage Criteria. The survey also explored the use of coverage criteria in test optimization. Results indicated that coverage criteria, such as feature coverage, code coverage, and combinatorial interaction testing, are applied across domains. Automotive and railway control systems, in particular, utilize feature coverage to ensure that new variants or features are sufficiently tested without redundant testing of common components.

When asked what coverage metrics are recorded that can be used in test selection, we found various traditional and variability-aware metrics used to record coverage, summarized in Figure 4. In particular, one respondent from the automotive domain and one from the railway domain mentioned the use of code, feature, and change coverage as recorded metrics for optimization. One of the partners in the automotive domain plans to have these metrics in place in the near future. The coverage of static build flags is of more importance for test optimization for the embedded systems observability domain. Furthermore, the TV and production domains use feature coverage as a metric that can enable test optimization. Finally, the representative company from the telecom domain uses the traditional code coverage metric for test optimization.

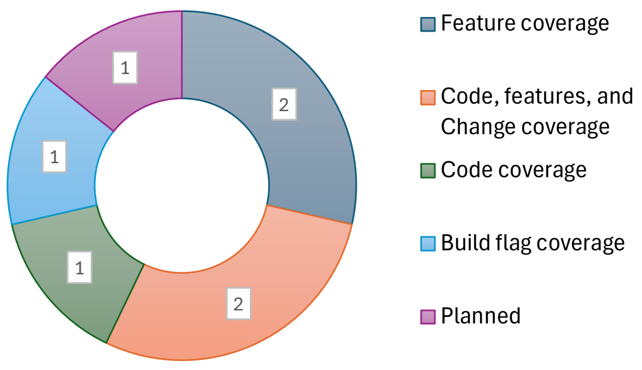

Fig. 4: Prominent coverage metrics recorded

Table 2: Perceived challenges in the area (RQ2)

Theme	Sub-theme and codes
1. Configurations to (not) test	1.1 Detection of redundant configurations 1.2 Effect of configuration change on code 1.3 Testing composed features
2. Impact analysis	2.1 Flagging changes that can break tests 2.2 Flagging changes that can fail tests
3. Variance and testing	3.1 Test script derivation for variants 3.2 Test selection for changes and variants 3.3 Test classification based on criticality 3.4 Critical scenarios identification
4. Coverage	4.1 Dead code effects coverage 4.2 Measuring feature coverage from component-level coverage

Takeaway: RQ1

In practice, variability in systems are still captured informally, often with natural language requirements. The number of variants and their combinations can explode and test selection and optimization for variability is a practical concern. Most of the studied representative companies do use test categorization and a variety of coverage metrics to enable test optimization for variability. Nevertheless, the respondents still see this as a challenging problem.

4.3 Challenges and improvement opportunities (RQ2):

Our RQ2 focuses on collecting and reporting potential areas of improvement and opportunities for research by collecting practical challenges in test optimization for variability. In this regard, we asked the representative companies from their domains how the current testing process can be improved. As mentioned in Section 3, we use a qualitative data analysis method known as thematic analysis to collect and document the findings for RQ2. Table 2 reports the area's high-level themes and concrete challenges (sub-themes and codes), briefly summarized as follows.

The thematic analysis of the survey responses highlighted several recurring challenges in test optimization. One major issue was the difficulty of effectively dealing with configuration parameters and variability. (1) Respondents found it challenging to detect redundant configurations of their product that could result in equivalent behavior. *"Many configurations have been tested, but many of these are believed to produce identical code (the same variant) as the relevance of many parameters depends on other configuration parameters. These dependencies are not documented. It could be useful for finding which configuration combinations are redundant."* The industry could benefit from approaches and tooling that can extract and capture configuration parameter dependencies. This will help enable the detection of equivalent variants and, in turn, optimize variability testing processes. In addition, the uncertain effects of a small change to configuration parameters on the resultant software hinder the maintenance and evolution of both the software and its tests. Finally, the thematic analysis also highlighted the challenges with test optimization in the presence of large and composed features (features that use other features). *"Some features are reused [...] The granularity of features is very low for us, and we don't model; we re-test [...]"*

(2) Another reoccurring theme in the survey responses was the challenges in applying change impact analysis to identify breakages. Most companies reported that while they could identify changes in code or requirements, they lacked tools to trace these changes to specific test cases, making it harder to prioritize tests. The analysis further highlighted the need for the detection of small changes that may break or fail tests. Additionally, the manual nature of test categorization and selection in some domains, such as embedded observability, was a barrier to optimizing test efforts. *"We don't have a tool for change management of requirements, features, and test cases. [...] We don't know anything about new features before we see failing test cases. Also, we have repeating scenarios with one parameter/flag change in tests. If we can analyze the effects of these parameters and remove repetition, we can save a lot of time."*

(3) Another challenge was the scalability of test optimization in systems with high variability. In this regard, the analysis highlights the difficulty in reusing test cases across product variants, particularly in domains where system architectures or configurations vary significantly between product variants. In embedded systems, testing becomes complex due to dynamic configurations, making test reuse ineffective, manual, and time-consuming. Therefore, test scripts for individual variants have to be manually adapted and evolved for variants. While there are design-time approaches for deriving tests for variants [5], there is still a lack of tooling for automated test evolution for product variants. Furthermore, some participants, particularly from the embedded systems domain, reported that the number of potential test cases could reach billions due to varying configurations. In such cases, effective test selection based on coverage criteria becomes critical but is often not fully automated. This highlights the challenge of balancing exhaustive testing against constraints, such as tests that cover changes. Finally, automated critical scenario identification from configuration could help avoid exhaustive testing. *"The existing product tests are used and modified as needed [for variants] [...] We don't have a tool for change management of requirements, features, and test cases. By identifying only the changes performed, then the tester can target only needed areas. [...] We don't know the combination of configurations that are used in practice by our customers. We naturally test the default settings and other 'likely' settings, and then as many other variants as possible."*

Test prioritization is widely used, but determining the most effective criteria for test case selection and ordering is challenging. In safety-critical domains like railway, although test selection is performed to target changes in code, the entire test suite still needs to be executed before release, making testing inefficient. *"[We target] the changes performed within the components, we only perform tests which affect those components. However, running all tests before releasing software is a common practice. "*

(4) Finally, the analysis highlighted several challenges on the integration between code coverage and variability. In particular, variability in systems can introduce dead code that handles a configuration that is not relevant in a particular variant, resulting in low code coverage. *"Testers need to know what is possible given the current design [...] is the dead code really dead, or is it a bad test case?"* In this regard, participants also highlighted the lack of tooling that can link component/variant-level coverage results to derive or compute feature/product-level code coverage. *"[It is challenging] to feed test results from variants into the product."*

The survey results indicate significant interest in further developing automated tools to support test optimization. Many participants highlighted the need for tools to automatically categorize and prioritize test cases based on coverage, risk, and criticality.

Takeaway: RQ2

A major issue is the lack of tools to trace code changes to specific test cases, hindering test prioritization. Manual test categorization and scalability are also problematic, especially in embedded systems with high variability, where test selection is often not automated. Reusing test cases across product variants is difficult due to dynamic configurations. While test prioritization is common, selecting the most effective criteria remains challenging, especially in safety-critical embedded domains.

5 Validity Threats

Participant selection for the survey posed a concern, as using non-probability sampling could introduce selection bias. Although a diverse range of industrial practitioners was included, they may not fully represent the broader population of industry professionals. This limitation was acknowledged, and efforts were made to include participants from different domains and regions to enhance the survey's representativeness.

Another potential threat to internal validity when using questionnaires is the risk of respondents misinterpreting the questions. However, this risk was significantly minimized since all industrial participants were involved in the same research project and shared a similar understanding of the terminology used.

6 Conclusion and Future directions

In conclusion, this study reveals that while test optimization practices for variant-intensive systems are common in some domains, such as automotive and railway systems, current practices often need to address the scalability and complexity of high variability, especially in domains like embedded systems. Many companies use manual processes for test categorization, and there is a need for more tooling to trace changes in code to specific test cases, which hampers efficient test prioritization. Reusing test cases across product variants remains challenging, particularly in systems with dynamic configurations. Future work should enhance automation for test categorization, prioritization, and selection based on coverage and risk, explicitly scaling these solutions for highly configurable systems.

References

1. Abbas, M., Hamayouni, A., Moghadam, M.H., Saadatmand, M., Strandberg, P.E.: Making sense of failure logs in an industrial devops environment. In: S. Latifi (ed.) ITNG 2023 20th International Conference on Information Technology-New Generations, pp. 217–226. Springer International Publishing, Cham (2023)
2. Abbas, M., Jongeling, R., Lindskog, C., Enoiu, E.P., Saadatmand, M., Sundmark, D.: Product line adoption in industry: An experience report from the railway domain. In: Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A, SPLC '20. ACM, New York, NY, USA (2020)
3. Ali, N., Engström, E., Taromirad, M., Mousavi, M.R., Minhas, N.M., Helgesson, D., Kunze, S., Varshosaz, M.: On the search for industry-relevant regression testing research. Journal of Empirical Software Engineering (2018)
4. Bryce, R.C., Colbourn, C.J.: Prioritized interaction testing for pair-wise coverage with seeding and constraints. Information and Software Technology **48**(10), 960 – 970 (2006). DOI https://doi.org/10.1016/j.infsof.2006.03.004. Advances in Model-based Testing
5. Bucaioni, A., Di Silvestro, F., Singh, I., Saadatmand, M., Muccini, H.: Model-based generation of test scripts across product variants: An experience report from the railway industry. Journal of Software: Evolution and Process **34**(11), e2498 (2022)
6. Cohen, M.B., Dwyer, M.B., Shi, J.: Interaction testing of highly-configurable systems in the presence of constraints. In: Proceedings of the 2007 International Symposium on Software Testing and Analysis, ISSTA '07, p. 129–139. Association for Computing Machinery, New York, NY, USA (2007). DOI 10.1145/1273463.1273482
7. Cohen, M.B., Snyder, J., Rothermel, G.: Testing across configurations: Implications for combinatorial testing. SIGSOFT Softw. Eng. Notes **31**(6), 1–9 (2006). DOI 10.1145/1218776.1218785
8. Colbourn, C.J., Cohen, M.B., Turban, R.: A deterministic density algorithm for pairwise interaction coverage. In: IASTED Conf. on Software Engineering, pp. 345–352 (2004)
9. Eljasik-Swoboda, T., Demuth, W.: Leveraging clustering and natural language processing to overcome variety issues in log management. In: ICAART (2), pp. 281–288 (2020)
10. Engström, E., Runeson, P.: Software product line testing–a systematic mapping study. Information and Software Technology **53**(1), 2–13 (2011)
11. Golagha, M., Lehnhoff, C., Pretschner, A., Ilmberger, H.: Failure clustering without coverage. In: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 134–145 (2019)
12. Hao, D., et al.: Test-case prioritization: achievements and challenges. Frontiers of Computer Science **10**(5), 769–777 (2016)
13. Kasunic, M.: Designing an effective survey. Carnegie Mellon University, Software Engineering Institute Pittsburgh, PA (2005)
14. Lamancha, B.P., Usaola, M.P., de Guzman, I.G.R.: Model-driven testing in software product lines. In: International Conference on Software Maintenance, pp. 511–514. IEEE (2009)
15. Lee, J., Kang, S., Lee, D.: A survey on software product line testing. In: International Software Product Line Conference, pp. 31–40. ACM (2012)
16. Linåker, J., Sulaman, S.M., Höst, M., de Mello, R.M.: Guidelines for Conducting Surveys in Software Engineering v. 1.1. Tech. rep., Lund University, Sweden (2015)
17. Memon, A., Gao, Z., Nguyen, B., Dhanda, S., Nickell, E., Siemborski, R., Micco, J.: Taming google-scale continuous testing. In: International Conference on Software Engineering, pp. 233–242. IEEE Press (2017)
18. Petke, J., Cohen, M.B., Harman, M., Yoo, S.: Practical combinatorial interaction testing: Empirical findings on efficiency and early fault detection. IEEE Transactions on Software Engineering **41**(9), 901–924 (2015). DOI 10.1109/TSE.2015.2421279
19. Sinnema, M., Deelstra, S.: Classifying variability modeling techniques. Information and Software Technology **49**(7), 717–739 (2007)
20. Strandberg, P.E.: Ethical interviews in software engineering. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–11. IEEE (2019)
21. Strandberg, P.E., Ostrand, T.J., Weyuker, E.J., Sundmark, D., Afzal, W.: Automated test mapping and coverage for network topologies. In: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 73–83 (2018)
22. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability **22**(2), 67–120 (2012)