From Fault Injection to Formal Verification: A Holistic Approach to Fault Diagnosis in Cyber-Physical Systems

Drishti Yadav
TU Wien
Vienna, Austria
drishti.yadav@tuwien.ac.at

Abstract

Cyber-Physical Systems (CPSs) face growing complexity, especially in safety-critical areas. Ensuring their correctness is vital to maintain full operational capacity, as undetected failures can be both costly and life-threatening. Therefore, advanced fault diagnosis procedures are essential for thorough CPS testing, enabling accurate fault detection, explanation, and rectification. This doctoral research contributes to the field by developing novel tools and techniques to enhance fault-based testing and diagnosis of CPSs. Our research focuses on testing of CPS dataflow models created in Simulink, validated against strict formal specifications. Our contributions include *(i)* an automated tool for systematic fault injection, *(ii)* a bio-inspired global optimization algorithm, *(iii)* a robust fault localization method, *(iv)* a novel approach to mutation testing for evaluating test suites against formal properties, and *(v)* a new coverage criterion tailored for CPS dataflow models. This comprehensive approach offers significant improvements over existing methods, ensuring thorough testing across various scenarios. We validate the effectiveness of our solutions using publicly available benchmarks from various domains. Our findings open new perspectives on CPS testing, laying the foundation for more robust CPSs.

CCS Concepts

• **Software and its engineering → Software testing and debugging**.

Keywords

Cyber-Physical Systems, Coverage Criteria, Debugging, Fault Injection, Fault Localization, Model-Based Development, Mutation Testing, Optimization, Signal Temporal Logic (STL), Simulink Models, Software Testing

ACM Reference Format:
Drishti Yadav. 2024. From Fault Injection to Formal Verification: A Holistic Approach to Fault Diagnosis in Cyber-Physical Systems. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '24), September 16–20, 2024, Vienna, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3650212.3685552

ISSTA '24, September 16–20, 2024, Vienna, Austria

ACM ISBN 979-8-4007-0612-7/24/09
https://doi.org/10.1145/3650212.3685552

1 Introduction

The rapid and accurate detection and diagnosis of faults are crucial to ensure fail-safe operation of Cyber-Physical Systems (CPSs), particularly in safety-critical sectors. To streamline the development process and mitigate the challenges faced by engineers during the design phase, the embedded software industry is increasingly adopting model-based development. In practice, engineers often utilize commercial toolchains and dataflow modeling languages, such as MathWorks' Simulink [23], to create prototypes of safety-critical CPSs.

As safety-critical CPSs grow in complexity [10], the importance of comprehensive testing of CPS models increases significantly, underscoring the need for advanced fault diagnostics that can quickly identify potential problems. Such advanced diagnostics enable efficient fault correction, enhancing the overall reliability and safety of the system. Despite the growing complexity of CPSs and advancements in testing methodologies, there remains a significant gap in the availability of automated, effective tools and methods for systematic fault-based testing, including fault detection and analysis. This gap hinders engineers from performing thorough fault diagnostics on CPSs, thereby compromising their reliability and safety in critical applications.

This doctoral research introduces innovative tools and techniques aimed at verifying the safety aspects of CPSs, thereby enabling swift and accurate fault diagnostics. The main contribution of this research, as the title suggests, lies in developing a fault-based testing framework specifically designed for safety-critical CPSs. This framework incorporates various activities, including fault injection, global optimization, search-based testing for fault localization, mutation testing in presence of formal properties, and coverage-based testing. These activities form an interconnected network, working synergistically rather than in isolation, to enhance both fault analysis and system testing.

More in details, our research primarily focuses on testing of CPS dataflow Simulink models governed by formal temporal logic specifications expressed in Signal Temporal Logic (STL) [22]. In our work, we first introduce FIM [6], an automated tool for systematic fault injection and mutation in CPS Simulink models, offering advanced features and scalable experimentation compared to current state-of-the-art tools and techniques. Also, we introduce the Blood Coagulation Algorithm (BCA) [32], a novel bio-inspired metaheuristic that excels in local optima avoidance, speed, and convergence, outperforming many state-of-the-art optimizers. Additionally, we investigate failing and passing executions of the system to identify fault locations, resulting in a cost-effective search-based fault localization method [7]. This method accurately localizes multiple faults in a system model, surpassing the performance of state-of-the-art

fault localization tool. As described in the paper, we also present Property-Based Mutation Testing (PBMT) [8], a fresh twist on mutation testing (MT) of software concerning properties. Another area of research emerges where the existing state-of-the-practice coverage metrics prove inadequate for evaluating the efficacy of test suites. This inadequacy is particularly pronounced in the context of CPS dataflow models, where executing a test case to cover one element can often lead to the execution of numerous other elements. To address this shortfall, we intend to propose a novel coverage metric customized for CPS dataflow models, aiming to comprehensively explore the system's internal behavior.

As this paper unfolds, we delve into the problems and objectives of this doctoral research, describe our artifacts (tools and methods), highlight some preliminary results and findings, and outline our plans for future research.

Paper Organization. We begin by outlining the problems and defining the main research objectives in Section 2. Next, Section 3 details our plan, methodologies, and the results achieved so far. Finally, Section 4 presents our concluding remarks.

2 Problem

In the following sections, we outline the main research objectives, offering a brief overview of the current state-of-the-art limitations and the Research Questions (RQs) under investigation.

2.1 Injecting Faults in CPS Simulink Models

The standard method for systematically evaluating a testing strategy involves assessing its effectiveness in identifying undesirable system behaviors in the presence of faults [9]. This is accomplished through fault injection and mutation testing, both of which are recommended by industrial safety standards such as ISO 26262 and IEC 61508, particularly in safety-critical domains [27]. A critical requirement for extensive mutation testing evaluations is an automated, programmatic mechanism to inject various types of faults into the system model without any human intervention while injecting faults. To our knowledge, none of the existing fault injection solutions for Simulink fulfill all these requirements, explaining the absence of systematic experiments in CPS testing approaches.

Several tools have been created for fault injection in Simulink models [14, 26, 28, 31], but each has its own set of limitations and drawbacks. For example, MODIFI [31] and ErrorSim [28] offer limited fault options and are not available publicly. SIMULTATE [26] provides an interactive user interface using Python and MATLAB, which enhances user-friendliness but may hinder scalability for experiments involving the injection of numerous faults. Another model-based technique for fault injection [14] supports typical faults but does not offer automated support for placing the fault blocks within the System-Under-Test (SUT). This brings us to the following research question:

RQ1: How can we develop automated and programmatic fault injection mechanism for CPS Simulink models to facilitate large-scale mutation testing evaluations?

Objective: Our aim is to create a solution that automatically injects faults and mutations into CPS Simulink models. This solution will provide a wide array of faults and comprehensive options for fault configuration, thereby facilitating scalable experiments.

2.2 Enhancing Search-based Testing

Optimization algorithms, especially metaheuristics, play a vital role in effective testing of CPSs through systematized navigation of the complex input spaces, consequently revealing faults and guaranteeing system resilience. *Falsification* (aka optimization-based falsification or *search-based testing*), in particular, is a potent technique for pinpointing specification violations through counterexamples [35]. Despite the utilization of various metaheuristic techniques in search-based testing, such as Simulated Annealing [1] and Tabu Search [13], striking the right balance between exploitation and exploration remains a significant challenge. Thus, it is imperative to develop new algorithms that can address the diverse challenges posed by CPSs. These algorithms must not only effectively balance exploration and exploitation in the search process but also exhibit faster convergence and excel in both constrained and high-dimensional spaces. Continuous advancements of these algorithms can substantially enhance system reliability and safety in real-world applications. This consideration leads us to the following question:

RQ2: How can we refine optimization algorithms for improving the testing of CPSs in real-world scenarios?

Objective: Our aim is to create an advanced strategy for search-based testing that effectively addresses the exploration-exploitation trade-off, while also being able to navigate high-dimensional search spaces and constrained optimization tasks typically found in advanced CPS controllers.

2.3 Localizing Faults in CPS Models

Fault localization is crucial for the design, verification, and debugging of CPSs. However, identifying the precise location of a fault that has caused a failure in a CPS model is a complex task due to the intricate structure and data-flow characteristics of these models. Usually, Run-time monitoring techniques are effective in detecting faulty or abnormal behaviors in systems by employing monitors for STL properties. In the literature, various approaches have been undertaken to localize and explain faults in a SUT, particularly by examining observed violating and falsifying traces [3–5, 20, 29]. Nonetheless, the challenge of fault localization in the presence of multiple faults remains largely unexplored. Furthermore, current fault localization methods tend to be time-consuming and resource-intensive. Consequently, there is a need for methods that can *quickly* and *accurately* pinpoint *multiple* faults. This leads us to the following research question:

RQ3: How can the observed behaviors of a system be utilized to accurately pinpoint multiple faults, potentially in large numbers?

Objective: Our goal is to create a precise and cost-effective fault localization method that integrates seamlessly with testing. This method should remain robust even when dealing with numerous and diverse types of faults in the SUT.

2.4 Mutation Testing against Properties

MT, a well-established technique in software quality assurance, serves as a valuable tool for evaluating test suites in terms of their fault-detection capability [2, 11, 12, 15]. However, its utility diminishes when software validation necessitates adherence to specific

requirements. This situation commonly arises in embedded software, where validation occurs against rigorous safety properties. In such contexts, the *relevance* of a mutant lies solely in its potential to impact the fulfillment of the tested properties, and a mutant is considered *meaningfully killed* only if it leads to the violation of those properties. Consequently, conventional MT becomes impractical and inefficient for fault detection in these cases. This leads us to the following question:

RQ4: How can we modify and improve conventional MT to reliably expose faults in embedded software undergoing rigorous validation of safety properties?

Objective: Our goal is to establish a formal framework for investigating mutants in relation to properties. Additionally, we aim to develop a test generation strategy tailored for CPS models to create test cases capable of effectively killing mutants with respect to formal properties.

2.5 Coverage Criteria for Testing CPS Models

In the field of embedded software testing, the effectiveness of test suites is typically assessed through their code coverage and fault-revealing capabilities. Practitioners use a variety of metrics, including input coverage [21], output coverage [24], and structural coverage to gauge coverage adequacy. Structural coverage metrics, particularly prevalent in both research and industry [16, 19, 33], provide valuable insights into how comprehensively a test suite examines system elements, revealing potential weaknesses in the test suite. However, existing research argue that relying solely on structural coverage criteria may be inadequate for uncovering faults [17, 18, 25, 30]. This issue is especially pronounced in CPS dataflow models, where the interconnected nature of elements means that covering one element with a test case can inadvertently cover many others. Consequently, depending exclusively on these metrics for testing of CPS dataflow models may not fully reveal the system's internal behavior. For thorough testing of CPS models, more advanced and nuanced coverage metrics, that can capture the complex behaviors and interactions within these systems, need to be developed. This consideration brings us to the following research question:

RQ5: What methods can be employed to refine and enhance coverage metrics to thoroughly capture the intricate behaviors and interactions in CPS models?

Objective: We seek to develop a new coverage metric that rigorously examines the internal behavior of a CPS model, and compare with other established coverage criteria. Furthermore, we intend to create a strategy to generate test cases that maximize the coverage, and evaluate the effectiveness of the resulting test suite in uncovering faults.

3 Work Plan and Preliminary Results

The objectives outlined in this paper are structured into five key milestones, each corresponding to the research questions detailed in Section 2. This doctoral research commenced in October 2020 and is anticipated to conclude in October 2024. As of June 2024, at the time of writing this paper, we are nearing the final stages of completion. We have successfully completed the first four milestones and are currently focusing on the fifth milestone.

Milestone 1: Advanced Fault Injection and Mutation Framework for Simulink Models. This milestone corresponds to RQ1 and culminates in the creation of **FIM** [6], an automated tool for systematic **F**ault **I**njection and **M**utation in CPS Simulink models. Before developing the tool, we recognized the complexities of working with Simulink models and gained a thorough understanding of programmatic editing, from lines to blocks. We also learned the basics of masking in Simulink, which was essential for creating a comprehensive *fault injection library* with parameterized blocks. Various injection strategies were tested, and we ultimately chose to implement the injection mechanism by adding new blocks, ensuring maximum control and the ability to observe and modify variable values without issues. To facilitate large-scale experiments, we designed an interface that operates without requiring user interaction during the experiments. Additionally, we explored multiple strategies for managing a large number of injected faults. We decided to support two complementary approaches: *(i)* MULTI-MODEL: generating numerous models with a single fault each and, *(ii)* SINGLE-MODEL: creating a single model that includes all the faults.

More in details, FIM consists of three main components: (1) the *Fault Library*, which includes 15 customized blocks for various types of block and line mutations, each with user-configurable fault parameters; (2) the *Fault Injection* Module, which injects faults based on a user-defined list of mutations in either SINGLE-MODEL or MULTI-MODEL mode; and (3) the *Fault Configuration* component, which allows users to activate fault blocks and configure them according to specified fault parameters. Additionally, FIM provides testers with full flexibility to control the fault injection space, the fault parameters and dynamically activate/deactivate faults according to their testing requirements. It is important to note that FIM serves as the primary tool for fault seeding and mutant generation in the experiments conducted to address RQ3, RQ4, and RQ5.

Milestone 2: Development of a Bio-inspired Global Optimizer: The Blood Coagulation Algorithm. This milestone aligns with RQ2 and involves the creation of a new metaheuristic optimization algorithm called the **B**lood **C**oagulation **A**lgorithm (**BCA**) [32]. BCA is inspired by the natural biological process of blood coagulation in the human body, specifically the cell-centric model where the movement of thrombocytes, analogous to particles in classical Particle Swarm Optimization, is guided by different mathematical equations. We evaluated BCA's efficiency on various mathematical benchmarks, including unimodal and multimodal functions, and compared it with several state-of-the-art algorithms. Additionally, we assessed its convergence, scalability, and performance in high-dimensional tasks, as well as its application in engineering design optimization challenges and CPS falsification. Our experimental results demonstrate BCA's superior ability to tackle real-world optimization problems more effectively. Note that we will employ the BCA optimizer to manage all search-based tasks necessary for tackling RQ3, RQ4, and RQ5.

Milestone 3: Enhancing Fault Localization Using Search-based Testing. This milestone addresses RQ3 by developing a cost-effective search-based fault localization algorithm [7]. Our method involves examining pairs of system executions—one failing and one passing—to accurately identify fault components. The underlying idea is that each pair should include two similar executions, with

their differences likely caused solely by the activation of the fault(s). This point-by-point comparison yields valuable insights for precise fault localization. In detail, we begin by presenting a novel search-based method for automatically generating a *passing* test that is *similar* to a *failing* test for data-flow Simulink models. We then use this failing test and its closely related passing counterpart to identify and rank suspicious variables based on their time of violation and degree of deviation, linking these variables to their respective model blocks. The outcome is a customized list of suspicious model variables and blocks, specifically designed to aid engineers in their debugging efforts. We evaluated the effectiveness of our approach on 240 faulty variants (involving one-fault, two-fault and three-fault models) of three benchmark Simulink models tested against STL specifications. Additionally, we applied our method with equivalence testing when explicit STL specifications were unavailable and compared its performance with `CPSDebug` [4], a state-of-the-art fault localization tool. Our experimental results demonstrate our approach's superior fault localization accuracy, scope reduction, computational efficiency, and robustness, even as the number of faults in the system models increases.

Milestone 4: Improving Test Suite Assessment with Property Based Mutation Testing. This milestone addresses RQ4 and introduces **P**roperty-**B**ased **M**utation **T**esting (**PBMT**) [8], a novel approach to MT that focuses on software properties. We argue that traditional MT is less significant and insightful than PBMT, particularly when software must be validated against stringent properties, as is often the case with safety-critical CPSs. In these scenarios, software is tested against formal safety properties. PBMT redefines the concept of mutant killability by considering the satisfaction and violation of a property for both the original and mutated versions of a program. Specifically, our approach ensures that a mutant is killed if its execution against a test case produces an output that violates the property ϕ, while the original program passes. This implies that the test case is effective in exposing faults, causing significant behavioral differences that lead to the violation of ϕ. Furthermore, we introduce the concept of *ϕ-trivially different mutants*, which are mutants that cannot be differentiated from the original program in terms of the property. In this context, a mutant is irrelevant not only if it is *equivalent* (showing no behavioral differences with respect to the original program), but also if the behavioral differences introduced are *not relevant* to the property ϕ.

We establish a formal framework for PBMT and the problem of mutant killing, facilitating the use of a global optimizer to generate test cases through a search-based approach. Our study on two Simulink models from the safety-critical CPS domain demonstrates that testing software against properties is more challenging and relevant than traditional MT, where mutants[1] can be more easily killed. Our evaluation reveals that state-of-the-art techniques such as Adaptive Random Testing and Falsification Testing are still weak in generating test suites that can effectively kill mutants when tested against properties.

Milestone 5: New Coverage Metric to Improve CPS Testing. This final milestone addresses RQ5. We believe that developing a coverage criterion focused on the *features* of *internal signals* will be highly effective in fault detection. Our goal is to create a coverage metric that comprehensively captures system behavior by analyzing the time and frequency domain *features* of *internal signals*. This approach aligns with research advocating the use of signal features [24, 34] and internal signals [17, 21] for fault identification and localization. As part of this doctoral research, we aim to establish an initial meta-definition of *feature-oriented coverage*, applicable to specific user-defined scenarios. Additionally, we plan to develop a test generation strategy that maximizes this feature-oriented coverage. Ultimately, we will evaluate the efficiency of these test suites in fault detection, comparing them to state-of-the-art test generation approaches. Specifically, we will utilize the concept of PBMT that we introduced in response to RQ4 to assess the effectiveness of test suites in identifying faults. As of now, the research work addressing RQ5 is still in progress and is yet to be considered for publication.

[1] Our focus is restricted to *first-order mutants* (FOMs). Given that the majority of current research in MT addresses FOMs in software artifacts, we evaluate our technique using models with single faults. The exploration of *higher-order mutants* (HOMs) is reserved for future research.

4 Concluding Remarks

In this paper, we detail our significant contributions to the field of CPS testing by providing robust solutions for fault diagnosis. Primarily, we offer a solution for automated fault injection in Simulink models. Additionally, we introduce a new global optimizer that outperforms several state-of-the-art metaheuristics in terms of avoiding local optima, speed, and convergence. Our results establish this optimizer as a viable and competitive solution for tackling real-world optimization challenges, including those found in CPS testing. We also propose a cost-effective, search-based fault localization technique that uses two similar system model executions with opposite outcomes to accurately pinpoint multiple faults. Furthermore, we challenge the relevance of traditional mutation testing (MT) in scenarios where software must be validated against stringent properties, suggesting new directions for MT. We also highlight our ongoing efforts to enhance test cases for comprehensive system evaluation by introducing a new coverage criterion that rigorously tests the features of internal signals of the system. This research aims to explore the fault-revealing capabilities of test suites designed to maximize our feature-oriented coverage. Overall, our work opens new perspectives on fault-based testing and analysis of CPSs. We believe that our contributions will empower engineers and testers to thoroughly evaluate CPS models, laying the groundwork for further exploration and development.

Acknowledgments

The author is advised by Prof. Ezio Bartocci, Professor at TU Wien, Austria. This work has been conducted under the supervision of the author's advisor, in collaboration with Dr. Dejan Ničković, Senior Scientist at the AIT Austrian Institute of Technology, Austria, and Prof. Leonardo Mariani, Professor at the University of Milano-Bicocca, Italy. This work has been supported by the Doctoral College Resilient Embedded Systems, which is run jointly by the TU Wien's Faculty of Informatics and the UAS Technikum Wien.

References

[1] Houssam Abbas and Georgios Fainekos. 2012. Convergence proofs for Simulated Annealing falsification of safety properties. In *50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012, Allerton Park & Retreat*

Center, Monticello, IL, USA, October 1-5, 2012. IEEE, USA, 1594–1601. https://doi.org/10.1109/ALLERTON.2012.6483411

[2] Allen Troy Acree, Timothy Alan Budd, Richard A. DeMillo, Richard J. Lipton, and Frederick Gerald Sayward. 1979. *Mutation Analysis*. techreport GIT-ICS-79/08. Georgia Institute of Technology, Atlanta, Georgia.

[3] Ezio Bartocci, Thomas Ferrère, Niveditha Manjunath, and Dejan Nickovic. 2018. Localizing Faults in Simulink/Stateflow Models with STL. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2018, Porto, Portugal, April 11-13, 2018*. ACM, USA, 197–206. https://doi.org/10.1145/3178126.3178131

[4] Ezio Bartocci, Niveditha Manjunath, Leonardo Mariani, Cristinel Mateis, and Dejan Nickovic. 2019. Automatic Failure Explanation in CPS Models. In *Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11724)*. Springer, USA, 69–86. https://doi.org/10.1007/978-3-030-30446-1_4

[5] Ezio Bartocci, Niveditha Manjunath, Leonardo Mariani, Cristinel Mateis, and Dejan Nickovic. 2021. CPSDebug: Automatic failure explanation in CPS models. *Int. J. Softw. Tools Technol. Transf.* 23, 5 (2021), 783–796. https://doi.org/10.1007/S10009-020-00599-4

[6] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, and Drishti Yadav. 2022. FIM: fault injection and mutation for Simulink. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*. ACM, USA, 1716–1720. https://doi.org/10.1145/3540250.3558932

[7] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, and Drishti Yadav. 2022. Search-based Testing for Accurate Fault Localization in CPS. In *IEEE 33rd International Symposium on Software Reliability Engineering, ISSRE 2022, Charlotte, NC, USA, October 31 - Nov. 3, 2022*. IEEE, USA, 145–156. https://doi.org/10.1109/ISSRE55969.2022.00024

[8] Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, and Drishti Yadav. 2023. Property-Based Mutation Testing. In *IEEE Conference on Software Testing, Verification and Validation, ICST 2023, Dublin, Ireland, April 16-20, 2023*. IEEE, USA, 222–233. https://doi.org/10.1109/ICST57152.2023.00029

[9] Marco Bozzano and Adolfo Villafiorita. 2007. The FSAP/NuSMV-SA Safety Analysis Platform. *Int. J. Softw. Tools Technol. Transf.* 9, 1 (2007), 5–24. https://doi.org/10.1007/S10009-006-0001-2

[10] Johan Cederbladh, Romina Eramo, Vittoriano Muttillo, and Per Erik Strandberg. 2024. Experiences and challenges from developing cyber-physical systems in industry-academia collaboration. *Software: Practice and Experience* 56, 6 (2024), 1193–1212. https://doi.org/10.1002/spe.3312

[11] Oscar Cornejo, Fabrizio Pastore, and Lionel C. Briand. 2022. Mutation Analysis for Cyber-Physical Systems: Scalable Solutions and Results in the Space Domain. *IEEE Trans. Software Eng.* 48, 10 (2022), 3913–3939. https://doi.org/10.1109/TSE.2021.3107680

[12] Richard A. DeMillo, Richard J. Lipton, and Frederick Gerald Sayward. 1978. Hints on Test Data Selection: Help for the Practicing Programmer. *Computer* 11, 4 (April 1978), 34–41.

[13] Jyotirmoy V. Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. 2015. Stochastic Local Search for Falsification of Hybrid Systems. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 9364)*. Springer, Switzerland, 500–517. https://doi.org/10.1007/978-3-319-24953-7_35

[14] Tagir Fabarisov, Ilshat Mamaev, Andrey Morozov, and Klaus Janschek. 2021. Model-based Fault Injection Experiments for the Safety Analysis of Exoskeleton System. *CoRR* abs/2101.01283 (2021). arXiv:2101.01283 https://arxiv.org/abs/2101.01283

[15] Daniel Fortunato, Jose Campos, and Rui Abreu. 2022. Mutation Testing of Quantum Programs: A Case Study With Qiskit. *IEEE Transactions on Quantum Engineering* 3 (2022), 1–17.

[16] Gordon Fraser and Andrea Arcuri. 2011. EvoSuite: automatic test suite generation for object-oriented software. In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*. ACM, USA, 416–419. https://doi.org/10.1145/2025113.2025179

[17] Gregory Gay, Ajitha Rajan, Matt Staats, Michael W. Whalen, and Mats Per Erik Heimdahl. 2016. The Effect of Program and Model Structure on the Effectiveness of MC/DC Test Adequacy Coverage. *ACM Trans. Softw. Eng. Methodol.* 25, 3 (2016), 25:1–25:34. https://doi.org/10.1145/2934672

[18] Laura Inozemtseva and Reid Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*. ACM, USA, 435–445. https://doi.org/10.1145/2568225.2568271

[19] Marko Ivankovic, Goran Petrovic, René Just, and Gordon Fraser. 2019. Code coverage at Google. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*. ACM, USA, 955–963. https://doi.org/10.1145/3338906.3340459

[20] Bing Liu, Lucia, Shiva Nejati, and Lionel C. Briand. 2017. Improving fault localization for Simulink models using search-based testing and prediction models. In *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017*. IEEE Computer Society, USA, 359–370. https://doi.org/10.1109/SANER.2017.7884636

[21] Bing Liu, Shiva Nejati, Lucia, and Lionel C. Briand. 2019. Effective fault localization of automotive Simulink models: achieving the trade-off between test oracle effort and fault localization accuracy. *Empir. Softw. Eng.* 24, 1 (2019), 444–490. https://doi.org/10.1007/s10664-018-9611-z

[22] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3253)*. Springer, Berlin, Heidelberg, 152–166. https://doi.org/10.1007/978-3-540-30206-3_12

[23] Mathworks. 2024. *Simulink Documentation*. Mathworks. Retrieved June 4, 2024 from https://in.mathworks.com/help/simulink/

[24] Reza Matinnejad, Shiva Nejati, Lionel C. Briand, and Thomas Bruckmann. 2019. Test Generation and Test Prioritization for Simulink Models with Dynamic Behavior. *IEEE Trans. Software Eng.* 45, 9 (2019), 919–944. https://doi.org/10.1109/TSE.2018.2811489

[25] Akbar Siami Namin and James H. Andrews. 2009. The influence of size and coverage on test suite effectiveness. In *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis, ISSTA 2009, Chicago, IL, USA, July 19-23, 2009*, Gregg Rothermel and Laura K. Dillon (Eds.). ACM, USA, 57–68. https://doi.org/10.1145/1572272.1572280

[26] Ingo Pill, Ivan Rubil, Franz Wotawa, and Mihai Nica. 2016. Simultate: A toolset for fault injection and mutation testing of simulink models. In *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, USA, 168–173. https://doi.org/10.1109/ICSTW.2016.21

[27] Ludovic Pintard, Jean-Charles Fabre, Karama Kanoun, Michel Leeman, and Matthieu Roy. 2013. Fault Injection in the Automotive Standard ISO 26262: An Initial Approach. In *Dependable Computing - 14th European Workshop, EWDC 2013, Coimbra, Portugal, May 15-16, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7869)*. Springer, Berlin, Heidelberg, 126–133. https://doi.org/10.1007/978-3-642-38789-0_11

[28] Mustafa Saraoğlu, Andrey Morozov, Mehmet Turan Söylemez, and Klaus Janschek. 2017. ErrorSim: A tool for error propagation analysis of simulink models. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Cham, 245–254. https://doi.org/10.1007/978-3-319-66266-4_16

[29] Nikhil Kumar Singh and Indranil Saha. 2020. Specification-Guided Automated Debugging of CPS Models. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39, 11 (2020), 4142–4153. https://doi.org/10.1109/TCAD.2020.3012862

[30] Matt Staats and Corina S. Pasareanu. 2010. Parallel symbolic execution for structural test generation. In *Proceedings of the Nineteenth International Symposium on Software Testing and Analysis, ISSTA 2010, Trento, Italy, July 12-16, 2010*. ACM, USA, 183–194. https://doi.org/10.1145/1831708.1831732

[31] Rickard Svenningsson, Jonny Vinter, Henrik Eriksson, and Martin Törngren. 2010. MODIFI: a MODel-implemented fault injection tool. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Berlin, Heidelberg, 210–222. https://doi.org/10.1007/978-3-642-15651-9_16

[32] Drishti Yadav. 2021. Blood coagulation algorithm: A novel bio-inspired meta-heuristic algorithm for global optimization. *Mathematics* 9, 23 (2021), 3011. https://doi.org/10.3390/math9233011

[33] Qian Yang, J. Jenny Li, and David M. Weiss. 2006. A Survey of Coverage Based Testing Tools. In *Proceedings of the 2006 International Workshop on Automation of Software Test, AST 2006, Shanghai, China, May 23, 2006*. ACM, USA, 99–103. https://doi.org/10.1145/1138929.1138949

[34] Justyna Zander-Nowicka. 2009. *Model-based testing of real-time embedded systems in the automotive domain*. Technical University Berlin, Berlin, Germany.

[35] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2021. Effective Hybrid System Falsification Using Monte Carlo Tree Search Guided by QB-Robustness. In *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12759)*. Springer, Switzerland, 595–618. https://doi.org/10.1007/978-3-030-81685-8_29

Received 2024-07-07; accepted 2024-07-22