

# Machine-learning assisted model-implemented fault injection

Mehrdad Moradi, Joachim Denil

University of Antwerp and Flanders Make, Belgium

## Abstract

Validation and verification of modern safety-critical systems demand an increasing amount of time and effort as systems become more complicated. Fault Injection (FI) is a well-known testing method that stresses the system in an unusual way to examine system's behavior. Traditional FI methods are not preferable in modern Cyber-Physical Systems (CPS) as they require too much effort to locate critical faults in the system. To tackle this problem, we propose an approach where the Machine Learning (ML) algorithm aids FI by efficiently injecting faults in the model under test automatically. The ML algorithm uses domain knowledge and simulation models at different abstraction levels to predict catastrophic faults, which fail the model's properties.

## Keywords

Fault injection, Machine learning, Domain knowledge, Validation and verification

## 1. Introduction

Validation, verification, and safety assessment of modern Cyber-Physical Systems (CPS) are challenging as they become more complicated, large scale, and heterogeneous [1]. In safety-critical industries like the automotive industry, safety assessment is crucial for vendors, and a tremendous amount of time is spent on testing their products [2]. There are many approaches to test and verify the system properties, but this paper focuses on Fault Injection (FI), which is a well-known method that uses simulation and experiments to perform safety analysis.

FI is traditionally applied to hardware and software prototypes. In FI, a tester stresses a small part of the system (physical or virtual) in an unusual way to observe the system behavior [3]. In FI, faults have three main attributes which establish fault space. They are (i) *type* (what should be injected?) (ii) *time* (when should be activated?), and (iii) *location* (where should be injected?) [4]. Traditional FI approaches are usually based on a random-based method [5], as such they are not efficient in terms of fault coverage (the ratio of the number of catastrophic faults to the total number of injected faults) [4] and performance. By increasing the complexity of modern CPS, applying traditional methods is inefficient with regard to the tremendous cost.

To address this inefficiency, we propose a model-based FI approach in which the user provides domain knowledge for a Machine Learning (ML) algorithm to aid FI in finding the potential

---

Woodstock'20: Symposium on the irreproducible science, June 01–05, 2020, Woodstock, NY

EMAIL: Mehrdad.Moradi@uantwerpen.be (M. Moradi); Joachim.Denil@uantwerpen.be (J. Denil)


URL: <https://www.uantwerpen.be/en/staff/mehrdad-moradi/> (M. Moradi);

<https://www.uantwerpen.be/en/staff/joachim-denil/> (J. Denil)

ORCID: 0000-0001-8748-069X (M. Moradi); 0000-0002-4926-6737 (J. Denil)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

critical faults. Model-based FI injects a fault model into the model under test, which can violate safety specifications. The ML algorithms chooses the fault parameter in each simulation, based on the parameter of the injected fault and the simulation result. As the proposed method intelligently and (semi-)automatically determines the faults with less manual effort, fewer simulation, and higher coverage, it mends some of the scalability issues found in traditional approaches. Furthermore, by leveraging multiple levels of abstraction, we also reduce the computational cost of executing computationally intensive simulations which reduces the total experimentation time. Finally, by applying our techniques to the model of the system, the analysis already starts in earlier design phases which decreases the cost of experiment.

The rest of the paper is organized as follows. In section 2 we explain recent advances in FI. Next, we describe the research objective and the proposed approach in section 3. Then, we express the past work in section 4 and the plan in detail in section 5. Finally, we conclude the paper in section 6.

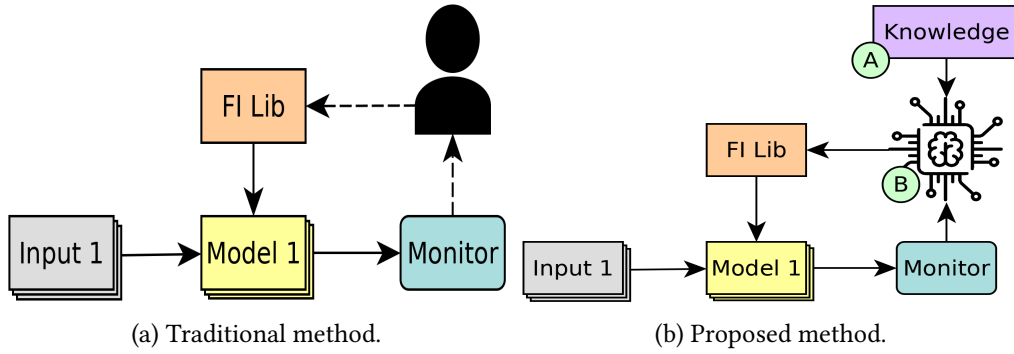
## 2. State-of-the-Art

In this section, we briefly address some of the recent advances in the Fault Injection (FI) technique in different applications and domains. There is a large body of literature about software-based and hardware-based FI; however, researchers in academia and industry pay less attention to model-implemented FI compare to the others [6]. The model-based techniques facilitate data analysis, validation, and verification of the system in an earlier phase of system design [7]. For example, authors in [8] modeled common faults in hardware and applied FI in Matlab Simulink, and authors in [9] proposed a domain independent tool that utilizes model-based mutation.

There are some researches, using Machine Learning (ML) algorithm to increase fault coverage. Authors in [10] exploit an ML algorithm to classify relevant and non-relevant fault campaigns and found a correlation between injected fault and system characteristics. In [11], the authors utilized a learn-based method to create a simplified model of the system under test by applying inputs and observing outputs. Then, they validate and verified the model under test using model checking techniques. Besides, in [12], authors use an ML algorithm to predict accident conditions of a vehicle and inject fault around accident time.

Some literature used fault space pruning to explore limited fault space that have higher potential to fail the system. The model-based FI utilizes this technique through injection simulation or before simulation. Authors in [13] prune the fault space by using probability analysis to score and rank potential fault locations. They applied their technique to sequential logic, and they increased their speed and scalability. In [14], authors divided the fault space into equivalence classes and chose one fault from each equivalence class to inject. They also use domain knowledge to cut unnecessary parts of the fault space.

Authors in [5] used a statistical approach to extract information about critical faults. They run numerous simulations to find fault propagation path and fault combinations that can affect system specifications.



**Figure 1:** Comparing FI in industrial practice to our proposed approach.

### 3. Research Objectives and Methodological Approach

In this study, the research objective is to improve the model-implemented FI process in terms of fault coverage and simulation time in CPSs. Traditional approaches are less effective in modern CPS as these systems are tremendously complex. In the traditional approaches of FI in Fig. 1a, the tester designs the fault campaign and continuously checks the results and analyzes them. However in the proposed technique in Fig. 1b, we utilize domain knowledge (marked by A) and an ML-based (marked by B) approach to choose the fault campaign in a way to increase the fault coverage and performance in the earlier design phase. Furthermore, in typical model-based systems engineering processes, the system is described at different levels of detail. We leverage the different levels to efficiently train the ML algorithm as it requires lots of data. Models at a higher level of abstraction are typically less computationally expensive to simulate.

The domain knowledge provides the ML algorithm some primary knowledge about the model under test such as discontinuity, input boundaries, different modes of system, etc. The ML algorithm exploits this knowledge to find critical faults which can violate system specification and efficiently explore the fault space. In addition, in our approach the tester is less involved in the testing procedure which reduces tester error and increases efficiency.

### 4. Past Work and Preliminary Results

Our past research results consist of three contributions to the state of art in FI. In our first contribution, we created a generic framework to apply FI at the model level [6]. Our approach uses model transformation to inject faults into models [15]. The user defines injection rules, and the framework automatically generates faulty models and runs the model-in-the-loop simulation. The framework gets the results and compares it with the result of the normal simulation to check the system's specification. Furthermore, we also generate source code for running the hardware-in-the-loop simulation. We validated the technique by creating a prototype tool to inject faults into the well known Mathworks Simulink tool.

A second contribution aims to efficiently apply FI in black-box models. In [16], we worked on FI in Functional Mock-up Interface (FMI) standard [17]. The FMI standards define a container and an interface for black-box simulation of CPS. With black-box models, we have less knowledge

about the system under test and thus less freedom to inject faults. We proposed to use sensitivity analysis before the injection process to gain knowledge about where the most effective signals are, how strong the amplitude of fault must be, and when the fault must be injected. We validated the technique on a model of an automotive power window. Other work accomplished has focused on the configuration of FMI simulations, with the aim of future integration of FI [18].

Our final contribution aims to reduce the fault space (what, where and when) of FI. We investigated the use of Reinforcement Learning (RL) [19]. We applied RL to efficiently search the fault space and to find the critical faults that violate the safety specifications. The technique was applied to Adaptive Cruise Control (ACC) application<sup>1</sup>. In our limited validation, we showed that our approach is more effective than traditional Monte-Carlo FI in terms of fault coverage and the number of simulations to find the first critical faults. This approach can be applied to broader applications such as hardware-based and software-based FI.

## 5. Future Work and Expected Results

As briefly described in section 3, we work on a ML-assisted model-implemented FI using domain knowledge. We plan to use probabilistic automata as a formalism to model the domain knowledge and system properties such that both the algorithm and tester can interact with. The current choice is Bayesian Networks (BN) that comprise both states and transitions and can describe the behavior of the model at a higher abstraction level. For example, signal's ranges and different modes of the system can be modeled as states and transitions. In the proposed algorithm, we have three main steps as (i) modeling the domain knowledge in BN, (ii) continual learning, and (iii) searching for the critical faults by FI.

In the first step, the user models the system under test, based on BN along with its specifications, limitations, developer assumption, etc. This model describes the system and corresponding constraints or properties in a high-level abstraction. For example, in the ACC use case, we can model different modes of the state machine (which controls the acceleration of vehicle) as states in BN, and we can map the sensors data to transition between these states. The ML algorithm uses this model as a behavioral model and interacts with it at the later steps. In future work, we will also investigate semi-automated techniques to create the model. The resulting model is extended by the ML algorithm in the second phase.

In the second phase, we must train the BN, but we lack a static data set for training. Therefore, the ML algorithm must run simulations and interact with both the BN and the model under test. It uses different predefined input scenarios to update or extend the underlying BN. The user needs to assist the training by defining rules in the domain knowledge and using multiple abstraction levels. For example, the tester can add some equations describing relations between probabilities in different transitions. For instance, in the ACC, we relate the probability of having an accident to probability of stopping the lead vehicle (the vehicle which is in front of the current vehicle) in the street, and these two probabilities have a positive relation with each other. Formulating those relationship aid the ML algorithm to converge to the ideal value faster.

---

<sup>1</sup><https://www.mathworks.com/help/driving/examples/adaptive-cruise-control-with-sensor-fusion.html>

Finally, the ML algorithm predicts the fault parameter based on the trained BN, which is trained in the previous step and generates the fault campaign. The ML ranks the faults based on their corresponding probability and starts the injection process. This prediction calculation is based on the *conditional probability*, as specified for two events  $A$  and  $B$  in the BN in Eq. 1.

$$P(A|B) = P(B|A) \times P(A)/P(B) \quad (1)$$

The ML algorithm iterates on the FI and evaluating its results to optimally creates the next fault campaign, until finding the critical faults. We will compare our method with random-based and statistical-based method to measure its effectiveness. We expect the proposed method to find more critical faults (higher fault coverage) than other methods, with less simulation and computational time, because the injection process is based on reasoning and probability analysis. The decision-making ability of the ML algorithm has been proven in many applications such as planning and prediction in autonomous vehicle [20]. Furthermore, preliminary experiments by different authors show a higher fault coverage compared to random-based FI techniques [12].

## 6. Conclusions

This work explores the combination of Machine Learning (ML) and domain knowledge to increase the efficiency of Fault Injection (FI) simulation at the model level. The ML algorithm uses domain knowledge to find critical faults that have the most destructive effect on system behavior and violate safety requirements. The proposed technique can cover broader applications with higher complexity and larger scale as it applies to the system's model with multiple level of abstraction and can tackle both heterogeneous and hybrid models. Besides, this approach is generic and can be applied not only to the white-box models, but also to the black-box as well as the gray-box models as the domain knowledge compensates the lack of knowledge about the model under test.

## Acknowledgments

This work was partly funded by Flanders Make project aSET (grant no. HBC.2017.0389) of the Flanders Innovation and Entrepreneurship agency (VLAIO). The authors thank Bentley James Oakes for his useful suggestions.

## References

- [1] E. A. Lee, Cyber physical systems: Design challenges, in: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), IEEE, 2008, pp. 363–369.
- [2] S. McConnell, Code complete, Pearson Education, 2004.
- [3] M.-C. Hsueh, T. K. Tsai, R. K. Iyer, Fault injection techniques and tools, Computer 30 (1997) 75–82.
- [4] A. Benso, P. Prinetto, Fault injection techniques and tools for embedded systems reliability evaluation, volume 23, Springer Science & Business Media, 2003.

- [5] H. Asadi, M. B. Tahoori, M. Fazeli, S. G. Miremadi, Efficient algorithms to accurately compute derating factors of digital circuits, *Microelectronics Reliability* 52 (2012) 1215 – 1226. URL: <http://www.sciencedirect.com/science/article/pii/S0026271411005646>.
- [6] M. Moradi, B. Van Acker, K. Vanherpen, J. Denil, Model-implemented hybrid fault injection for simulink (tool demonstrations), in: *Cyber Physical Systems. Model-Based Design*, Springer, 2018, pp. 71–90.
- [7] P. Micouin, *Model Based Systems Engineering: Fundamentals and Methods*, John Wiley & Sons, 2014.
- [8] R. Svenningsson, J. Vinter, H. Eriksson, M. Törngren, Modifi: a model-implemented fault injection tool, in: *International Conference on Computer Safety, Reliability, and Security*, Springer, 2010, pp. 210–222.
- [9] P. Gómez-Abajo, E. Guerra, J. de Lara, M. G. Merayo, A tool for domain-independent model mutation, *Science of Computer Programming* 163 (2018) 85 – 92.
- [10] F. R. da Rosa, R. Garibotti, L. Ost, R. Reis, Using machine learning techniques to evaluate multicore soft error reliability, *IEEE Transactions on Circuits and Systems I: Regular Papers* 66 (2019) 2151–2164. doi:10.1109/TCSI.2019.2906155.
- [11] H. Khosrowjerdi, K. Meinke, A. Rasmusson, Virtualized-fault injection testing: A machine learning approach, in: *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2018, pp. 297–308.
- [12] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, R. K. Iyer, ML-based fault injection for autonomous vehicles: A case for Bayesian fault injection, in: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 112–124. doi:10.1109/DSN.2019.00025.
- [13] H. Sabaghian-Bidgoli, P. Behnam, B. Alizadeh, Z. Navabi, Reducing search space for fault diagnosis: A probability-based scoring approach, in: *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, pp. 545–550.
- [14] X. Meng, Q. Tan, Z. Shao, N. Zhang, J. Xu, . Zhang, Optimization methods for the fault injection tool seinjector, in: *2018 International Conference on Information and Computer Technologies (ICICT)*, 2018, pp. 31–35. doi:10.1109/INFOCT.2018.8356836.
- [15] J. Denil, P. J. Mosterman, H. Vangheluwe, Rule-based model transformation for, and in simulink, in: *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative, DEVS '14*, Society for Computer Simulation International, 2014.
- [16] M. Moradi, C. Gomes, B. J. Oakes, J. Denil, Optimizing fault injection in FMI co-simulation through sensitivity partitioning, in: *Proceedings of the 2019 Summer Simulation Conference, SummerSim '19*, Society for Computer Simulation International, San Diego, CA, USA, 2019. doi:10.5555/3374138.3374170.
- [17] FMI, *Functional Mock-up Interface for Model Exchange and Co-Simulation*, Technical Report, FMI development group, 2014.
- [18] C. Gomes., B. J. Oakes., M. Moradi., A. T. Gámiz., J. C. Mendo., S. Dutré., J. Denil., H. Vangheluwe., Hintco – hint-based configuration of co-simulations, in: *Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications - Volume 1: SIMULTECH.*, 2019, pp. 57–68.
- [19] M. Moradi, B. J. Oakes, M. Saraoğlu, A. Morozov, K. Janschek, J. Denil, Exploring fault parameter space using reinforcement learning-based fault injection-a preprint, In *Safety*



