Lecture Notes in Computer Science **16504**

Advanced Research in Computing and Software Science

Subline of Lecture Notes in Computer Science

More information about this series at https://link.springer.com/bookseries/558

Elvira Albert · Corina Pasareanu
Editors

Fundamental Approaches to Software Engineering

29th International Conference, FASE 2026
Held as Part of the International Joint Conferences
on Theory and Practice of Software, ETAPS 2026
Turin, Italy, April 11–16, 2026
Proceedings

Springer

Editors
Elvira Albert
Complutense University of Madrid
Madrid, Spain

Corina Pasareanu
Carnegie Mellon University
Pittsburgh, PA, USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-032-22773-7 ISBN 978-3-032-22774-4 (eBook)
https://doi.org/10.1007/978-3-032-22774-4

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

ETAPS Foreword

Welcome to the 29th edition of ETAPS, which took place as an on-site event in Turin, Italy during April 11–16, 2026!

ETAPS 2026 was the 29th instance of the International Joint Conferences on Theory and Practice of Software (ETAPS). ETAPS is an annual federated conference established in 1998, and consists of four main conferences: ESOP, FASE, FoSSaCS, and TACAS. Each conference has its own Program Committee (PC) and its own Steering Committee (SC). The ETAPS main conferences cover various aspects of software systems, ranging from theoretical computer science to foundations of programming languages, tools and algorithms for system analysis, and formal approaches to software engineering. Organizing these conferences in a coherent, highly synchronized conference programme enables researchers to participate in an exciting event, having the possibility to meet many colleagues working in different directions in the field, and to easily attend talks of different conferences. In addition to its four main conferences, ETAPS 2026 also hosted fifteen satellite workshops and two colocated events, which together further attracted many researchers from all over the globe.

ETAPS 2026 received 456 submissions in total, 138 of which were accepted, yielding an overall acceptance rate of 30%. Out of the 138 accepted papers, 16 papers were selected as ETAPS distinguished papers. I thank all the authors of submitted papers for their interest in ETAPS, all the reviewers for their reviewing efforts, the PC members for their contributions, and in particular the PC (co-)chairs for their hard work in running this entire intensive process in a constructive, objective and timely manner. I congratulate all authors of the ETAPS 2026 accepted papers!

ETAPS 2026 featured the unifying invited keynotes by

- Monika Henzinger (Institute of Science and Technology Austria, Austria), delivering a talk about "Guarding Privacy Over Time: Challenges and Solutions in Continuous Data Observation",
- Einar Broch Johnsen (University of Oslo, Norway), discussing "Formal Methods Meet Digital Twins: Challenges and Opportunities".

ETAPS 2026 hosted the invited keynote speakers

- Christel Baier (Technische Universität Dresden, Germany) for FoSSaCS, presenting "Verification of Infinite-horizon Properties of Dynamic Bayesian Networks",
- Guy Van den Broeck (University of California, Los Angeles, USA) for TACAS, introducing "Symbolic Reasoning in the Age of Large Language Models".

The ETAPS 2026 invited tutorials were provided by

- Mieke Massink (CNR-ISTI Pisa, Italy) on "Model Checking in Space with Applications to Medical Image Analysis",
- Leonardo de Moura (Amazon Web Services, USA) surveying "The Lean Programming Language and Theorem Prover".

The ETAPS 2026 programme also featured a lively Ask-Me-Anything session, interactive tool demos, a Diversity, Equity, and Inclusion session, SV-Comp and Test-Comp community building events, and the ETAPS industry day. The goal of the ETAPS industry day is to bring industrial practitioners into the heart of the research community and to catalyze the interaction between industry and academia. The ETAPS 2026 industry day was organized by Giorgio Audrito (University of Turin, Italy), Sean Kauffman (Queen's University, Kingston, Canada), and Nikolai Kosmatov (Thales Research and Technology, Palaiseau, France).

ETAPS 2026 was organized by the Department of Computer Science of the University of Turin, which is the center for coordinating research, teaching, dissemination and technological transfer in computer science in Turin, Italy. The department covers both methodological and application oriented aspects of computer science, and performs research in several interdisciplinary areas. This is reflected in the collaborations with other research centers and companies in many scientific areas and in its participation in national, European and international projects.

ETAPS 2026 was further supported by the following associations and societies: ETAPS e. V. (the ETAPS Association), EATCS (European Association for Theoretical Computer Science), EAPLS (European Association for Programming Languages and Systems), and EASST (European Association of Software Science and Technology).

The ETAPS Steering Committee consists of an Executive Board, and representatives of the individual ETAPS conferences, as well as representatives of EATCS, EAPLS, and EASST. The Executive Board consists of Laura Kovács (TU Wien, chair), Andrzej Wąsowski (IT University of Copenhagen, vice-chair), Thomas Noll (RWTH Aachen, treasurer), Arnd Hartmanns (University of Twente, artifact evaluation coordinator), Barbara König (University of Duisburg-Essen, proceedings coordination), Caterina Urban (Inria, PhD activities), Elizabeth Polgreen (University of Edinburgh, social media), Jan Kofroň (Charles University Prague, organisational support, website), Jan Křetínský (Masaryk University Brno and TU Munich, diversity & inclusion), and Marieke Huisman (University of Twente, blog, awards). Further members of the ETAPS Steering Committee committee are: Robbert Krebbers (Radboud University Nijmegen), Azalea Raad (Imperial College London), Luís Caires (Tecnico ULisboa), Elvira Albert (Universidad Complutense de Madrid), Corina Păsăreanu (Carnegie Mellon University), Erika Ábrahám (RWTH Aachen), Marsha Chechik (University of Toronto), Marie-Christine Jakobs (LMU Munich), Nathalie Bertrand (Inria Rennes), Stefan Milius (Friedrich-Alexander Universität Erlangen-Nürnberg), Alexandra Silva (Cornell University), Joël Ouaknine (MPI-SWS Saarbrücken), Andrzej Murawski (University of Oxford), Sebastian Junges (Radboud University Nijmegen), Guy Katz (The Hebrew University of Jerusalem), Christian Schilling (Aalborg University), Naijun Zhan (Peking University), Joost-Pieter Katoen (RWTH Aachen and University of Twente), Dirk Beyer (LMU Munich), Fabrice Kordon (Sorbonne University Paris), Laure Petrucci (Université Paris 13), Peter Y.A. Ryan (University of Luxembourg), Claudio Menghi (University of Bergamo and McMaster University Hamilton), Mark Lawford (McMaster University Hamilton), Maurice ter Beek (CNR-ISTI Pisa), Ferruccio Damiani (University of Turin), Kim Guldstrand Larsen (Aalborg University), Bernhard Beckert (KIT Karlsruhe), Mattias Ulbrich (KIT Karlsruhe), Reiko Heckel (University of Leicester), Vladimiro Sassone

(University of Southampton), Anton Wijs (Eindhoven University of Technology), and Nikolai Kosmatov (Thales Research and Technology, Palaiseau).

The ETAPS 2026 local organization team consisted of Maurice ter Beek (CNR-ISTI Pisa, general co-chair), Ferruccio Damiani (University of Turin, general co-chair), Barbara Boni (Synesthesia Turin, local organization chair), Vincenzo Ciancia (CNR-ISTI Pisa, satellite events co-chair), Luca Paolini (University of Turin, satellite events co-chair), Maria Tacconi (Synesthesia Turin, satellite events co-chair and publicity co-chair), Francesco Brocero (Synesthesia Turin, web co-chair and volunteers co-chair), José Proença (University of Porto, web co-chair), Gianluca Torta (University of Turin, publicity co-chair and local proceedings co-chair), Lucy James (Synesthesia Turin, sponsor chair), Giovanna Broccia (CNR-ISTI Pisa, local proceedings co-chair), Giorgio Audrito (University of Turin, volunteers co-chair), Riccardo Sieve (UiO Oslo, volunteers co-chair), and Reiner Hähnle (TU Darmstadt, wine chair).

I would like to take this opportunity to thank all authors, keynote speakers, invited tutorial speakers, and attendees. Special thanks goes to the organizers of the ETAPS 2026 satellite workshops and colocated events. ETAPS 2026 is grateful for the generous support of Amazon Web Services, AccessiWay, Camera di Commercio Industria Artigianato e Agricoltura di Torino, the Department of Computer Science of the University of Turin, Springer Nature, and Turismo Torino e provincia Convention Bureau. I thank our general co-chairs Maurice ter Beek (CNR-ISTI Pisa) and Ferruccio Damiani (University of Turin), who made it all happen in Turin, and their local organization team for their enormous efforts to make ETAPS 2026 a fantastic event. I am especially grateful to Barbara Boni, Maria Tacconi, and Lucy James (Synesthesia Turin) for handling the organizational process in a smooth and reliable way. Last but not least, a big thanks to Jan Kofroň for all his help as an ETAPS Fellow and providing online presence support for the ETAPS conferences and the ETAPS Association.

I hope you all enjoyed ETAPS 2026!

April 2026

Laura Kovács
ETAPS SC Chair, President of the ETAPS Association

Preface

FASE 2026, the 29th International Conference on Fundamental Approaches to Software Engineering, was held from April 11–16, 2026, in Turin, Italy, as part of the 29th ETAPS International Joint Conferences on Theory and Practice of Software (ETAPS 2026). FASE serves as a premier venue for researchers, developers, and users to discuss innovations in software engineering.

The topics of interest for FASE include: requirements, design, architecture, and modeling of software systems, applications of AI to software engineering and applications of software engineering to AI-based systems, software quality, model-driven engineering, software processes, as well as software evolution.

There were four submission categories for FASE 2026:

1. Research papers, which clearly identify and justify a principled advance to the fundamentals of software engineering.
2. Empirical-evaluation papers, which evaluate existing software challenges or critically validate current proposed solutions with scientific means, that is, by empirical studies, controlled experiments, rigorous case studies, and simulations.
3. New Ideas and Emerging Results (NIER) papers, which seek to disrupt the status quo with forward-looking, thought-provoking, innovative research on the foundations of software engineering, as well as lessons learned from the past.
4. Tool demonstration and data showcase papers, which present a new tool, a new tool component, novel extensions to an existing tool, or a new dataset.

This year, 68 papers were submitted to FASE. Three papers were desk rejected. The remaining 65 papers were distributed in categories 1–4 as follows: 45 research papers, 9 empirical-evaluation papers, 8 NIER papers, and 3 tool-demonstration and data showcase papers. Each paper underwent a double-blind peer review process, where three program committee members reviewed each submission. The review process spanned 9 weeks, ensuring thorough evaluation and discussion of submissions. It was possible to submit an artifact for evaluation alongside a paper, if made long-term available and declared in the Data-Availability Statement. The program committee extensively discussed the papers and ultimately decided to accept 21 papers included in these proceedings: 15 research papers, 3 empirical studies, 2 NIER papers, and 1 tool paper, resulting in an acceptance rate of 32%.

Artifacts comprise tools, models, proofs, or other data for validating the results of a paper. The artifact evaluation committee (AEC) reviewed the artifacts based on their documentation, ease of use, and, most importantly, whether the results presented in the corresponding paper could be accurately reproduced. As in 2025 FASE offered a joint voluntary artifact evaluation together with ESOP and FoSSaCS to authors of accepted papers. All of the 8 artifact submissions that were linked with accepted FASE 2026 submissions met the requirements for the “Artifacts Available” badge. In addition, 2 submissions were awarded the Artifacts “Evaluated – Functional” badge and 5 submissions the Artifacts “Evaluated – Reusable” badge.

FASE 2026 was proud to host an invited tutorial by Mieke Massink from the C.N.R. –Area della Ricerca di Pisa– Ist. ISTI. These proceedings contain the invited paper supporting the tutorial.

FASE 2026 also hosted Test-Comp 2026, the 8th International Competition on Software Testing. This event evaluated 21 tools for automatic test generation for C programs, where 11 test-generation tools were registered and actively supported by development teams, including one tool that participated for the first time. One coverage validator was run in four different configurations to evaluate the coverage of the test-suites produced by the test-generation tools. In addition, the new test-suite validation tool TestCoCa participated for the first time. The FASE 2026 proceedings contain a competition report by the Test-Comp chair and 5 short papers selected by the competition jury. The short papers describe 5 out of the tools participating with active team support. The 5 short papers were reviewed by a separate program committee (jury); each was assessed by at least three jury members. Two sessions in the FASE 2026 program were reserved for Test-Comp: (1) a presentation session with a report by the competition chair and summaries by the development teams of participating tools, and (2) an open community meeting in the second session, jointly with SV-COMP.

We would like to thank all the people who helped make FASE 2026 successful. First, we thank the authors for submitting their papers. The PC members and additional reviewers did a great job: they contributed informed and detailed reports and engaged in the PC discussions. We thank Reiner Hähnle and Marie-Christine Jakobs, initial and current chairs of the FASE steering committee, and Marieke Huisman and Laura Kóvacs, also initial and current chairs of the ETAPS steering committee, for their valuable advice. Lastly, we would like to thank the overall organization team of ETAPS 2026. We extend our gratitude to Gianluca Torta and Giovanna Broccia, the local proceedings chairs, for their diligent oversight of the proceedings preparation. We also thank the Artifact Evaluation Committee (AEC) for their assessment of submitted artifacts and the Test-Comp 2026 jury members for their evaluation of competition submissions. Additionally, we appreciate the editorial support of Springer Nature, as well as the local organizers of ETAPS 2026 in Turin, Italy, for their efforts in facilitating this event. Finally, we note the Gold Open Access publication of these proceedings in the Lecture Notes in Computer Science (LNCS) series, ensuring unrestricted access to the contributions presented at FASE 2026.

February 2026

Elvira Albert
Corina Pasareanu
Dirk Beyer
Yannic Noller

Organization

Program Committee

Erika Abraham	RWTH Aachen University, Germany
Elvira Albert	Universidad Complutense de Madrid, Spain
Dirk Beyer	LMU Munich, Germany
Artur Boronat	University of Leicester, UK
Tevfik Bultan	University of California at Santa Barbara, USA
Ana Cavalcanti	University of York, UK
Marsha Chechik	University of Toronto, Canada
Priyanka Darke	Tata Consultancy Services, India
Stijn De Gouw	The Open University, UK
Bernd Fischer	Stellenbosch University, South Africa
Gordon Fraser	University of Passau, Germany
Divya Gopinath	NASA Ames (KBR Inc.), USA
Marie-Christine Jakobs	LMU München, Germany
Susmit Jha	SRI International, USA
Martin Jonáš	Masaryk University, Czechia
Sun Jun	Singapore Management University, Singapore
Thierry Lecomte	CLEARSY, France
Ravi Mangal	Colorado State University, USA
Raffaela Mirandola	Politecnico di Milano, Italy
Corina Pasareanu	CMU, NASA, KBR, USA
Luigia Petre	Åbo Akademi University, Finland
Cesar Sanchez	IMDEA Software Institute, Spain
Natasha Sharygina	University of Lugano, Switzerland
Dániel Varró	Linköping University, Sweden and McGill University, Canada
Andrzej Wasowski	IT University of Copenhagen, Denmark
Haoze Wu	Amherst College, USA
Jianjun Zhao	Kyushu University, Japan

Test-Comp Program Committee and Jury

Dirk Beyer (Chair)	LMU Munich, Germany
Kaled Alshmrany	Institute of Public Administration, Saudi Arabia
Max Barth	LMU Munich, Germany

Zhenbang Chen	National University of Defense Technology, China
Marie-Christine Jakobs	LMU Munich, Germany
Martin Jonáš	Masaryk University, Brno, Czechia
Matthias Kettl	LMU Munich, Germany
Thomas Lemberger	LMU Munich, Germany
Christophe Meudec	South East Technological University, Ireland
Marek Trtík	Masaryk University, Brno, Czechia
Henrik Wachowitz	LMU Munich, Germany
Chenfeng Wei	University of Manchester, UK

Additional Reviewers

Mohammad Afzal
David A. Anisi
Aren A. Babikian
Daniel Baier
Anna Becchi
Konstantin Britikov
Matías Brizzio
Margarita Capretto
Boqi Chen
Bharti Chimdyalwar
Achintya Desai
Mara Downing
Grigory Fedyukovich
Attila Ficsor
Dominik Frey
Carolina Gerlach
Marek Jankola
Matthias Kettl
Rick Koenders
Shrawan Kumar
Faezeh Labbaf
Fabrizio Leopardi
Marian Lingsch-Rosenfeld
Kumar Madhukar
Juraj Major
Logan Murphy
Sahar Nasimi Nezhad
Nico Naus
Matthew Peng
Venkatesh Ramanathan
Moeketsi Raselimo
Pedro Ribeiro
Andoni Rodriguez
Chia Sabah
Laboni Sarker
Stefano Schivo
Vincenzo Scotti
Md Shafiuzzaman
Jeroen Spaans
Jan Strejček
Alexandra van der Spuy
Erik Voogd
Henrik Wachowitz
Yiran Wang
Philipp Wendler

Contents

Testing and Verification

Competition on Software Testing (Test-Comp 2026)

Failure Modes and Effects Analysis: An Experience from the E-Bike Domain

Andrea Bombarda[§1], Federico Conti[§1], Marcello Minervini[1], Aurora Zanenga[1], and Claudio Menghi[1,2]

[1] University of Bergamo, Bergamo, Italy
f.conti12@studenti.unibg.it
{andrea.bombarda,marcello.minervini,aurora.zanenga,
claudio.menghi}@.unibg.it
[2] McMaster University, Hamilton, Canada

Abstract. Software failures can have catastrophic and costly consequences. Failure Mode and Effects Analysis (FMEA) is a standard technique used within Cyber-Physical Systems (CPS) to identify software failures and assess their consequences. Simulation-driven approaches have recently been shown to be effective in supporting FMEA. This paper presents our experience with using FMEA to analyze the safety of a CPS from the e-Bike domain. We used Simulink Fault Analyzer, an industrial tool that supports engineers with FMEA. We identified 13 realistic faults, modeled them, and analyzed their effects. We sought expert feedback to analyze the appropriateness of our models and the effectiveness of the faults in detecting safety breaches. Our results reveal that for the faults we identified, our models were accurate or contained minor imprecision that we subsequently corrected. They also confirm that FMEA helps engineers improve their models. Specifically, the output provided by the simulation-driven support for 38.4% (5 out of 13) of the faults did not match the engineers' expectations, helping them discover unexpected effects of the faults. We discuss our results and ten lessons learned.

Keywords: E-Bikes, Simulink® Fault Analyzer™, Safety Analysis

1 Introduction

Failure Mode and Effects Analysis (FMEA) is widely used in Cyber-Physical Systems (CPS) development to analyze software safety [62,67,47]. For example, it has been used in NASA programs [49,46] such as Apollo and Skylab, and it is used by Ford [21]. FMEA requires identifying potential failure modes (Failure Mode - FM) and analyzing their causes and effects (Effects Analysis - EA).

Safety assurance and safety analysis are increasingly becoming essential in software engineering [11,56,69]. To support software safety analysis from the early stages of the software development process, researchers have developed approaches that embed safety analysis within system modeling [28,33,24,51,31,6]. For example, Simulation-driven FMEA [54] is an automated software engineering solution that checks the effects of the faults defined in an FMEA table by

E. Albert and C. Pasareanu (Eds.): FASE 2026, LNCS 16504, pp. 259–282, 2026.
https://doi.org/10.1007/978-3-032-22774-4_14

performing simulations of models enriched with faults. The results of these simulations enable engineers to analyze the effects of the faults and, depending on their criticality, develop strategies that mitigate them. Simulation-driven FMEA provided encouraging results on a pivotal study that represented a flight control system of an unmanned ultralight helicopter [54]; it is currently integrated within the Simulink® Fault Analyzer™ [39]. The Simulink® Fault Analyzer™ is a tool for the safety management of software systems developed in Simulink®; it enables the connection of faults, hazards, and mitigation logic.

Safety analysis experts need empirical evidence regarding the benefits and limitations of the different techniques to decide whether to incorporate them into their working pipeline. Simulation-driven FMEA [54] and the Simulink® Fault Analyzer™ have been recently introduced within Simulink® (R2023b). To assess the industrial applicability of Simulation-driven FMEA, it is paramount to empirically evaluate its effectiveness and provide practitioners with guidelines and lessons learned [3], as widely recognized by the research and industrial software engineering communities [3,15,30,50,57,42,17,43,66]. This need is of primary importance for Simulink® [7,60,8,9], as Simulink projects and models are typically industrial and usually not shared or available due to corporate policies [5,14].

To mitigate this need, this paper reports on industrial applicability and the usefulness of Simulation-driven FMEA on a study subject from the e-Bikes domain. Specifically, our study subject is a Simulink® model related to the software controller of an e-Bike. It is developed in the context of a project which aims at improving "green" mobility solutions, including e-Bikes, and involving two large companies operating in the automotive and mobility sectors. E-Bikes are vehicles that support riders with power from an electric motor. Our case study is relevant because (a) the e-Bike software is often designed in Simulink® [62]; (b) e-Bikes have a considerable market size (27.15 USD billion in 2022, expected to grow to USD 82.84 billion by 2030 [25]); (c) they are software-intensive systems [64], as the software controls the electrical motor of the e-Bike and regulates its speed and possibly the regenerative power; (d) they are safety-critical systems which must comply with safety regulations and requirements [58,59]; and (e) our results and lessons learned are based on the interaction with an expert (the third author of this paper) who is designing the software for the controller of an e-Bike within the context of an EU-funded project that involves several industrial partners who specialize in the e-Bike domain as well as on discussions with Simulink® engineers. Our findings are useful for software engineers who work as Simulink® engineers, use the Simulink® Fault Analyzer™, or work as safety analysts.

To evaluate the industrial applicability of Simulation-driven FMEA, we considered 13 realistic faults; we (a) modeled them within the Simulink® model using the Simulink® Fault Analyzer™ and (b) analyzed their effects. This is a significant number of software faults considering the application domain. Designing these faults required a holistic view of the system, including its motor and battery. Acquiring this knowledge is time-consuming. It required us to interact with domain experts and thoroughly study the application domain to acquire

the knowledge to operate in this field. We retrospectively estimate this effort as six months of work for the entire team.

To assess the industrial applicability and the benefits of Simulation-driven FMEA in our case study, we interviewed our expert to ensure that the models of our faults reflected their intended behavior. Our results reveal that all faults were modeled correctly (53.8%) or with minor imprecisions (46.2%) that were fixed. Our discussion with Simulink® experts enabled us to fix the models for these faults. We interviewed the expert to understand the expected consequences of the faults and compare them with those provided by Simulation-driven FMEA. For over 38% of the faults, the output provided by the simulation-driven support for FMEA provided by the Simulink® Fault Analyzer™ did not match the engineer's expectation, thereby leading to the field engineers discovering unexpected effects of the faults. To provide a thorough interpretation of our results, we discussed them with Simulink® engineers, where we presented the objective of the work, our results, and collected their informal feedback and interpretation of the results. These discussions led to the formalization of ten lessons learned — i.e., five Simulink® specific (useful for Simulink® engineers and users of the Simulink® Fault Analyzer™) and five Simulink® agnostic (useful for safety analysts).

This paper is organized as follows. Section 2 introduces our study subject. Section 3 summarizes Simulation-driven FMEA. Section 4 presents our empirical evaluation. Section 5 discusses our results and presents lessons learned. Section 6 presents related work. Section 7 concludes our work.

2 The E-Bike Study Subject

Figure 1 presents our study subject: The Simulink® model of an e-Bike. This model is developed within the context of a project that involves two large companies operating in the automotive and mobility sectors. Its development time is approximately 100 hours of work by a single electronic engineer. The controlled system consists of several components.

The *Environment* models external forces (e.g., the friction and aerodynamic drag). It models external loads, mimicking the actual resistance an e-Bike would encounter during operation and that would influence motor performance. Its input is the speed of the e-Bike (*Measured speed*). Its output is a signal that simulates the effects of friction and aerodynamic torque (R).

The *Brushless Direct Current Motor (BLDC)* converts electrical energy into rotational motion and torque. Its inputs are the currents (a, b, c) applied to the three phases of the BLDC and the neutral point (n). Its outputs are the mechanical rotation of the rotor of the motor (R) and motor case (C).

The *Sensor* monitors the torque of the e-Bike, and the speed and the sector of its rotor. Its input is the torque of the rotor (R). Its outputs are the active sector (*Sector*) of the BLDC motor and the e-Bike speed (*Measured speed*).

The *Battery* stores and retrieves electrical energy. It receives the current that recharges the battery (- *Batt*) as input. It outputs a current from the energy stored within the battery (+ *Batt*) to feed the motor.

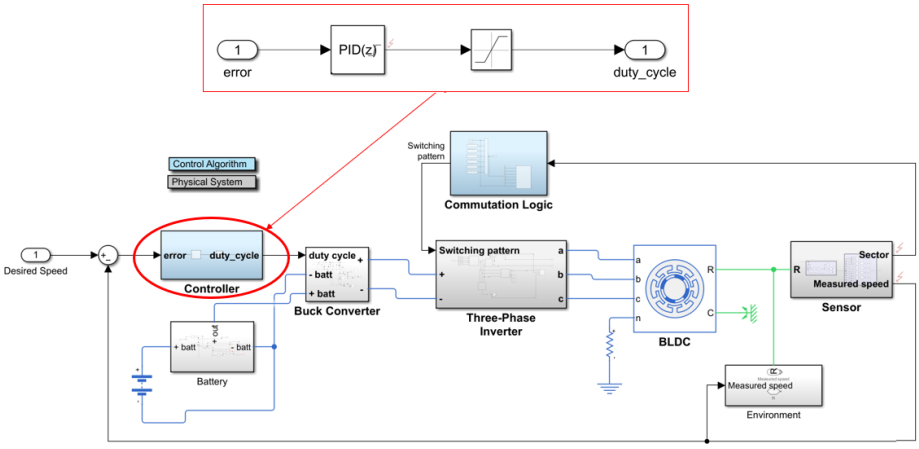


Fig. 1: Simulink® model for the e-Bike.

The *Three-Phase Inverter* converts the direct current into alternating current. It receives two inputs (*-Batt* and *+Batt*) from the buck converter. It acts on the motor, depending on the direct current (DC) signal (*Switching pattern*) received from the software controller (*Controller*). It outputs the currents applied to the three phases of the BLDC (*a*, *b*, *c*).

The *Commutation Logic* determines the voltage to be supplied by the three-phase inverter that is used to regulate the stator winding currents and to create a rotating magnetic field of the rotor of the BLDC motor. Its input is the active sector of the BLDC (*Sector*). Its outputs are the control signals sent to the electronic switches of the three-phase inverter (*Switching pattern*).

The *Buck Converter* converts a higher input voltage to a lower output voltage based on the duty cycle value. Its inputs are the duty cycle (*duty_cycle*) and the voltage from the battery (*- Batt*, *+ Batt*). Its output is the voltage to be fed into the Three-Phase Inverter (*-*, *+*).

The *(Software) Controller* regulates the BLDC motor to ensure that the measured speed (the model's output) matches the desired speed selected by the user (the model's input). Its input is the error difference (*Error*) between the desired and the measured speed. Its output is the duty cycle (*duty_cycle*).

Engineers *did not* analyze the *safety* of their system design. This work aims to analyze how Simulation-driven FMEA and the tool support provided by Simulink® Fault Analyzer™ can enable engineers to design a safer system.

3 Simulation-driven FMEA

Simulink® Fault Analyzer™ enables engineers to perform FMEA safety analysis by providing support for (a) defining potential failure modes (FM) by modeling faults (Section 3.1) and (b) analyzing their effect (EA) by simulating their consequences (Section 3.2).

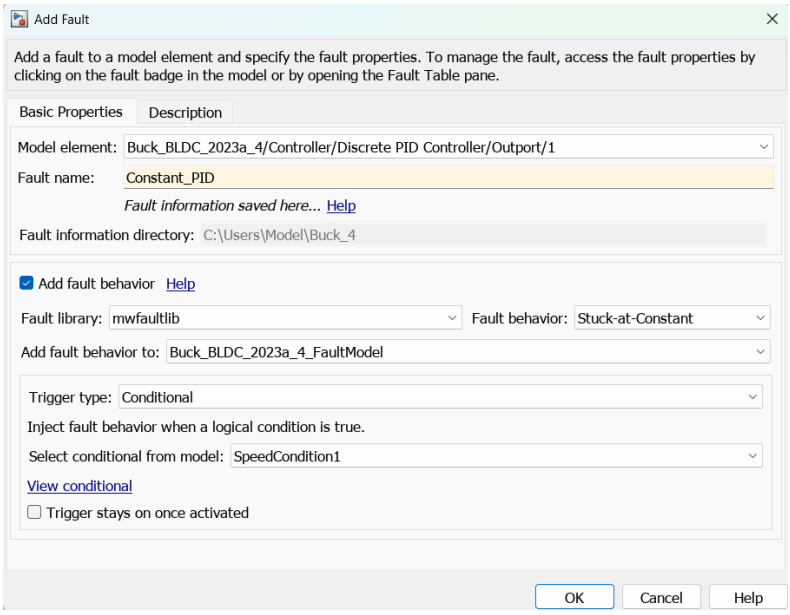


Fig. 2: Adding new faults with the Simulink® Fault Analyzer™.

3.1 Support for Failure Mode - FM

Simulink® Fault Analyzer™ enables engineers to extend the design model with faults. For example, Figure 1 presents three faults that are inserted on the *Sector*, *Measured speed*, and *PID(z)* signals. Simulink® Fault Analyzer™ graphically identifies faults with a lightning symbol on the signal on which the fault is applied (or on a component, for the Simscape™ [37] toolbox).

To insert a fault, engineers should select a signal (or component) and use the graphical interface presented in Figure 2. The interface displays the model element related to the fault ("*Model element*") and enables engineers to set the fault name ("*Fault name*") as well as to define its behavior ("*Add fault behavior*"). Simulink® Fault Analyzer™ maintains the fault logic encapsulated in specific blocks to keep it separated from the model logic and prevent the fault logic from inadvertently becoming part of the model. To define the fault behaviors, engineers can reuse existing faults from a predefined library ("*Fault library*"). Table 1 presents the list of faults provided by the Simulink® Fault Analyzer™ and a short description. For example, for the "*Constant_PID*" from Figure 2, the engineer selected the "*Stuck-at-Constant*" fault behavior from the library *mwfaultlib*, thereby forcing the signal to assume a constant value. Graphically, the fault is modeled as in Figure 3: The fault input is discarded, and the constant value 1 is always given as output.

Furthermore, faults can be activated via triggers. Table 2 presents the triggers provided by the Simulink® Fault Analyzer™ and a short description. For

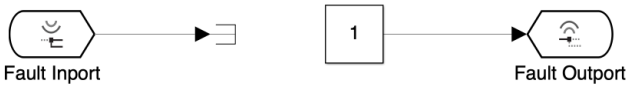


Fig. 3: Graphical representation of the "*Constant_PID*" fault.

Table 1: Fault behavior

Fault behavior	**Description**
Add Noise	Adds random noise to the signal.
Negate Value	Negates the value of a signal.
Absolute Value	Transforms the signal into its absolute value.
Stuck-at-Ground	Sets the signal to the ground.
Gain	Adds a gain to the signal.
Offset-by-1	Adds an offset to the input signal.
Unit Delay	Delays the signal by a single time unit.
Stuck-at-Constant	Sets the signal to a constant value.

example, for the "*Sector_fault*", the engineer selected the "*Conditional*" trigger, and specifies a condition ("*SpeedCondition1*") that forces the fault to be activated when the speed becomes greater than 50 *rpm*.

3.2 Support for Effect Analysis - EA

Simulation-driven FMEA supports EA by providing engineers the possibility of simulating the model with the faults injected to analyze their effects.

The Simulink® Fault Analyzer™ provides Fault Tables that enable engineers to activate different faults. For example, Figure 4 presents (a portion of) the Faults Table for our e-Bike model. The engineers activated the fault "*Constant_PID*" and disabled all the other faults.

Then, simulating the model within Simulink® enables the analysis of the fault effect. For example, Figure 5 depicts the desired speed (yellow line), the speed of a simulation where the fault is not enabled (blue line), and the speed of the current simulation (pink line). The simulation reveals that when the condition on the speed ("*SpeedCondition1*") is triggered — i.e., when the motor speed exceeds 50 *rpm* — the speed of the e-Bike suddenly stops following the desired speed. It then increases until 300 *rpm*, decreases until 20 *rpm*, and begins oscillating. This result enables engineers to understand the consequences of the fault. Specifically, forcing the gain of the PID to a constant value results in a significant increase in the speed of the e-Bike that exceeds the maximum speed of $25km/h$ (i.e., 170 *rpm* wheel speed, considering a 28-inch wheel), which is mandated by most European countries [58,59].

Information regarding the effects of the faults helps engineers improve their models. For example, engineers can decide to mitigate the fault "*Constant_PID*"

Table 2: Trigger types

Trigger type	Description
Conditional	Activates a fault when a specific condition, on any of the signals or measures available in the model, is satisfied.
Timed	Activates a fault after the specified simulation time.
Manual	Activates a fault manually during the simulation by clicking on a button.

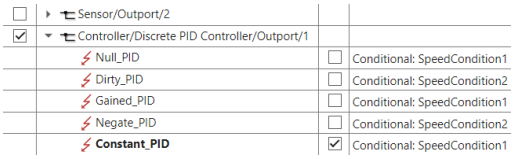


Fig. 4: A portion of the Fault Table for our e-Bike.

by applying appropriate strategies (e.g., adding another PID that takes control when problems occur or adding saturation mechanisms).

4 Evaluation

To assess the usefulness of Simulation-driven FMEA, we consider the following research questions (RQ):

- **RQ1**: How helpful is the support from the Simulink® Fault Analyzer™ in *modeling* the faults?
- **RQ2**: How helpful is the Simulation-driven FMEA for *safety* analysis?

We present our benchmark (Section 4.1) and answer RQ1 (Section 4.2) and RQ2 (Section 4.3).

4.1 Benchmark Design

To answer our research questions, two of the authors analyzed the e-Bike study subject and identified 15 faults. Table 3 provides the identifier ("ID") and a textual description ("Description") for each of these faults. Our faults involve both the cyber (*Controller*) and the physical (*Sensor*) parts of the CPS specified in the lower and upper parts of Table 3, and are applied both to Simulink® and Simscape™ components. For example, fault F1 refers to the breakage of the speed sensor, while fault F12 refers to the software controller and involves some noise on the output of the PID.

To ensure that our evaluation is based on realistic faults, we interacted with our expert to assess the criticality, plausibility, and frequency of each fault. We considered realistic faults that are either plausible, frequent, or critical.

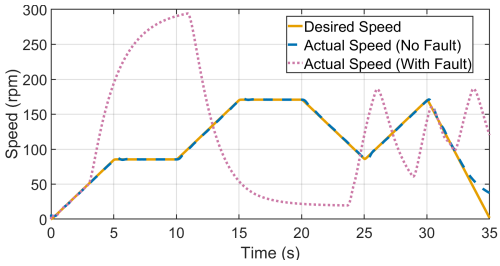


Fig. 5: The desired speed and the speed of the e-Bike when the fault is activated and deactivated.

We conducted a structured interview with the designer of the e-Bike controller to identify realistic faults. The designer had to assign a score between one and five for plausibility, frequency, and criticality; Moreover, the designer was required to justify their decision. The lowest bound (0) represents lower plausibility, frequency, or criticality, and the highest bound (5) corresponds to highly plausible, frequent, or critical faults. Table 3 reports the score provided by the expert for the plausibility (Plaus.), frequency (Freq.), and criticality (Critc.) of each fault. We considered a fault plausible, frequent, and critical if the score assigned by the expert is greater than or equal to 3.

The majority of our faults are plausible (66%), critical (74%), and not frequent (74%). Note that we expected the majority of faults to be infrequent, as undesirable behavior is not expected to be common.

We excluded faults F3 and F8 (identified with a blue background in Table 3). These faults involve an amplification error in the speed or sector sensors. We considered them to be non-realistic as they are not plausible, critical, and frequent, i.e., they had both plausibility, frequency, and criticality values lower than 3. Therefore, our benchmark consists of the remaining 13 realistic faults that are used to answer our RQs.

4.2 RQ1: Support of Simulink Fault Analyzer

To assess how the Simulink® Fault Analyzer™ helps in modeling faults, we proceeded as follows.

- *Step 1.* We considered each of the 13 faults and modeled them. The goal of this step is to assess whether we could model all the faults of our benchmark using the Simulink® Fault Analyzer™ (see Section 3.1). However, our models can potentially be inaccurate and not adequately reflect the expected fault behavior. Indeed, we considered a fault to be *accurate* when the model we provided matched the real behavior of the fault. To mitigate this problem, we performed an additional step.
- *Step 2.* We interviewed our expert to determine whether the models of our faults were appropriate. For inaccurate models, we tried to fix them.

Table 3: Plausibility, frequency, and criticality of the faults from our benchmark

ID	Description	Plaus.	Freq.	Critic.
F1	Breakage of the speed sensor [65]	4	4	5
F2	Noisy speed sensor [48]	4	2	1
F3	Amplification error in the speed sensor	1	1	1
F4	Delayed speed sensor [13]	1	1	3
F5	Speed sensor with constant value	4	4	5
F6	Breakage of the sector sensor	4	4	5
F7	Noisy sector sensor	4	2	1
F8	Amplification error in the sector sensor	1	1	1
F9	Delayed sector sensor [13]	1	1	3
F10	Sector sensor with constant value	4	4	5
F11	PID controller design error	3	2	5
F12	Noise on the output of the PID	4	2	4
F13	Amplification of the output of the PID	4	2	3
F14	Inverted PID output	1	1	5
F15	Constant output of the PID	3	2	5

Results. Table 4 presents the models of the faults from *Step 1* (Table 3). Note that, for many faults, multiple models (determined by several trigger types) were produced. Table 4 provides the identifier of the modeled fault ("ID"), the modeled behavior ("Fault behavior"), and the selected trigger type ("Trigger type"). For certain faults, we could not decide whether a "Timed" or a "Conditional" trigger was more appropriate because we did not discuss the triggers when we identified our faults with our expert. Therefore, we proposed alternative trigger options for discussion with our expert. For example, for fault F1, we considered the behavior "Stuck-at-Ground" and two triggers: Timed and Conditional.

The feedback from the domain expert (*Step 2*) confirmed that 53.8% (7 out of 13) of the models selected for the faults were accurate. These models are those with white background in Table 4. For the faults that had multiple trigger options (F1, F2, F4, F5, F6, F7, F9, and F10), the expert confirmed that "Conditional" was the appropriate trigger. Indeed, expert confirmed that faults do not typically emerge after a fixed time interval; rather, they occur when certain conditions arise, and those conditions cannot be defined a priori in purely time-based terms.

Further, the expert confirmed that 46.2% (6 out of 13) of our faults were correctly modeled, except for minor inaccuracies, e.g., a non fully correct condition or timing in the trigger. These models are those with a blue background color in Table 4. Examples of inaccuracies include incomplete conditions in conditional triggers and incorrect constant values (e.g., in F10 and F15). The minor inaccuracies have been fixed.

To summarize, Simulink® Fault Analyzer™ enables us to model all the 13 faults of our benchmark. In this process, we did not identify any major limitations in the support provided by this tool. Nevertheless, we learned a few lessons (see Section 5) that may help further improve this tool.

Table 4: Identifier (ID), fault behavior, and trigger type for our faults.

ID	Fault behavior	Trigger type
F1	Stuck-at-Ground	Timed & Conditional
F2	Add Noise	Timed & Conditional
F4	Unit Delay	Timed & Conditional
F5	Stuck-at-Constant	Timed & Conditional
F6	Stuck-at-Ground	Timed & Conditional
F7	Add Noise	Timed & Conditional
F9	Unit Delay	Timed & Conditional
F10	Stuck-at-Constant	Timed & Conditional
F11	Stuck-at-Ground	Conditional
F12	Add Noise	Conditional
F13	Gain	Conditional
F14	Negate Value	Conditional
F15	Stuck-at-Constant	Conditional

RQ1: Tool support

Simulink® Fault Analyzer™ successfully helped us design the models for the 13 faults of our benchmark.

4.3 RQ2: Usefulness of FMEA for Safety Analysis

To assess the usefulness of FMEA for safety analysis, we proceeded as follows.
• *Step 1*. We collected feedback from the expert regarding the expected effects of the faults by conducting a structured interview. During the interview, we asked what the expected system behavior was when each of the faults from Section 4.2 was activated and recorded their answer.
• *Step 2*. We used the support for EA provided by Simulation-driven FMEA (Section 3.2) to analyze the effect of the fault.
• *Step 3*. We compared the expected effect from the expert and the experimental effect from Simulation-driven FMEA.

Results. For each fault model presented in Table 4, Table 5 describes the expected effect from the expert (*Step 1*). Table 6 reports faults for which the experimental results did not match the expectations of the engineer (*Step 2* and *Step 3*). The portion of the effect that did not match the expectation is highlighted in italic. The expectation of the engineer is confirmed in 61.6% (8 out of 13) of the faults. For the remaining 38.4% (5 out of 13) of the faults, the simulation revealed effects that deviated from the engineer's expectation.

We observed that the differences identified during our experiments deviated from experts' expectations for two reasons. In certain cases (i.e., F7), the expected outcomes did not align with the actual results obtained during simulation. Indeed, for fault F7, while the expert expected "*extremely high currents until the motor stops spinning*," the simulation stopped with an error message caused by the sector signal having a value of "0", which falls outside the accepted range of 1 to 6. This error is due to a bug in the modeling of the three-phase

Table 5: Expected effect from the expert

ID	Expected Effect
F1	The PID will have a high duty cycle. Additionally, at least one of the three current phases (and the battery currents) will show high values.
F2	The duty cycle will reach its maximum value.
F4	The PID will take more time to regulate the speed.
F5	The PID will increase the duty cycle, leading to an actual speed higher than the desired one.
F6	The system will stop because it is unable to encode the value, which lies between 1 and 6.
F7	The system will show extremely high currents, and the motor will stop spinning.
F9	The system will stop.
F10	The system will show very high phase currents until the motor slows down and eventually stops, then remains stationary.
F11	The duty cycle will fall to zero, and the motor will slow down until it stops.
F12	Phase currents will significantly oscillate.
F13	The PID will saturate, and the system will try to accelerate even if unnecessary.
F14	The system will slow down until it stops.
F15	The system will accelerate the motor regardless of speed, until electrical equilibrium is reached or until the fault disappears.

inverter. The engineer did not consider the case in which the component could receive invalid values, which could happen because of noise or external factors in real environments. Thus, the three-phase inverter does not correctly handle invalid sector values, causing the simulation to stop prematurely instead of exhibiting the expected system behavior in the presence of the fault. Ideally, the system should perform an emergency stop and slowly decelerate till the motor completely stops. In other instances (i.e., F10, F12, F13, and F15), while the outcomes partially matched the expectations, additional effects were observed during the experiments, or minor inaccuracies emerged. For example, for fault F13, the expert expected the PID to force the system to accelerate, even if this was unnecessary. The simulation revealed that the PID output had significant oscillations when the fault occurred, which was not expected. Therefore, this analysis demonstrates the effectiveness of the Simulation-driven FMEA supported by Simulink® Fault Analyzer™.

RQ2: Usefulness

FMEA was useful for analyzing the safety of our e-Bike. It enabled the engineers to uncover fault side-effects that had previously been overlooked. In addition, it helped identify unexpected effects.

5 Discussion and Lessons Learned

We present the lessons learned (Section 5.1), discuss them (Section 5.2), and summarize threats to the validity of our findings and experiments (Section 5.3).

Table 6: Experimental effect from Simulation-driven FMEA

ID	Experimental Effect
F7	*The simulation stops with an error message since the value "0" for the sector signal is not included within the accepted range [1-6].*
F10	The system shows very high phase currents. When the fault is triggered, the measured speed shows marked oscillations around zero, if the initial speed is low, or it decreases slowly, showing evident oscillations. *When the fault is deactivated, the PID maximizes the duty cycle to restore the speed as desired from the desired speed input. This process is very aggressive, especially if compared to the scenario in which the fault is not triggered, where smoother control actions are performed.*
F12	*Constant, but slight,* oscillations in the phase currents.
F13	*The PID shows evident oscillations when the fault occurs*, then restores standard behavior. *There is no relationship between the initial value of the PID and the experimental outcome.*
F15	The system accelerates the motor as much as possible, with duty cycle 1, then drops to 0 when the fault deactivates. *This causes an alternation of very high and very low measured speeds, and also of very high and very low phase currents, both when the fault alternates between active and inactive.*

5.1 Lessons learned

We held joint meetings with MathWorks experts to describe and analyze our results and outline the lessons learned from this study. The feedback from MathWorks experts was pivotal to confirm a proper usage of the Simulink® Fault Analyzer™, its modeling capabilities, and the Simulation-driven FMEA support. In the following account, we outline our lessons learned by classifying them into two categories: Simulink® Specific and Simulink® Agnostic.

Simulink®-specific. These lessons are related to the usage of the Simulink® environment. They are useful for SW engineers working as Simulink® engineers and for users of the Simulink® Fault Analyzer™.

• *Lesson 1 (Triggers).* Simulink® Fault Analyzer™ offers several types of triggers, including manually activated, activated at a specific time instant (timed), or activated when a specific condition is true (conditional). For our e-Bike example, we had to model an event-based trigger. For example, the trigger for fault F1 (Stuck-at-ground) applies when the sensor breaks. This problem is likely to occur when the desired speed undergoes a sudden change — for example, in the case of a sudden acceleration or braking. Therefore, the fault should apply to the measured speed when the sensor suddenly breaks and remains broken till the end of the simulation. By default, conditional triggers activate the faults each time a specific system condition is true and stop the fault injection as soon as it becomes false. To force the fault to remain active until the end of the simulation, the flag "Trigger stays on once activated" must be activated. We initially overlooked this flag and defined the logic to detect the sudden change and to maintain the condition of the trigger to true until the end of the simulation within the

Simulink® model (partially defeating the benefits of using the Simulink® Fault Analyzer™). The meeting with the Simulink® experts evidenced the presence of the flag. We modified the model of the fault to appropriately utilize the features provided by the Simulink® Fault Analyzer™.

• *Lesson 2 (Target of the fault application).* For Simulink® models, faults are typically applied to physical signals, except for the Simscape™ portion of the models [36]. For Simscape™, faults are applied to components rather than to signals. We believe that extending Simscape™ by providing the opportunity to add faults to signals can benefit practical applications. For example, fault F2 requires adding noise to the speed sensor. This fault may occur when electromagnetic noise is present on the controller board's cables. However, to directly apply the fault to an electrical (Simscape™) signal, it is necessary to drive a Simscape™ component (e.g., a variable resistor [41]) that changes the resistance depending on the value of a Simulink® signal that is affected by the fault.

• *Lesson 3 (Single fault activation on a specific signal).* The Simulink® Fault Analyzer™ enables the activation of multiple faults (related to different signals) within a single simulation using the multiple simulations panel [38]. However, activating both F1 and F2 requires simultaneously activating two faults on the same signal. This situation either requires (a) inserting the two faults at the source and the destination of the connection, respectively, or (b) creating an additional fault behavior that mimics the scenario in which both faults are activated. Note that while the second solution also applies to situations in which more than two faults need to be activated, the first solution only applies to the case of two faults. Extending the Simulink® Fault Analyzer™ by providing the activation of multiple faults on the same signal will avoid the creation of additional fault behavior.

• *Lesson 4 (Activation of a fault on multiple signals).* In certain cases, we had to simultaneously apply the same fault to multiple signals. For example, fault F7 requires considering noise on the sector sensor caused by electromagnetic interference. In this case, the same noise should be applied to more than one signal, i.e., all cables in the same area of the controller's board. To represent this situation, we modified the model by adding a Simulink® block that generated a constant signal with value "0", adding the noise fault to this signal, and summing the value of this signal to all signals subjected to electromagnetic interference. In this manner, when the fault is activated, the noise is applied to all the signals subjected to electromagnetic interference. Extending the Simulink® Fault Analyzer™ by providing the opportunity to apply the same fault to multiple signals would avoid modifying the model under analysis.

• *Lesson 5 (Accelerator Execution Mode).* Unlike the normal execution mode, the accelerator mode uses Just-in-Time (JIT) acceleration to generate an execution engine in memory instead of generating C code or MEX files [40]. This mode increases performance and reduces the simulation time. This mode is useful to support quick prototyping. The Simulink® Fault Analyzer™ does not currently support the accelerator mode. In our scenario, the accelerator mode would have enabled us to save approximately 30 seconds for each simulation.

Simulink®- agnostic. These lessons are not Simulink-specific. They are valuable for safety analysts, regardless of the safety framework they use.

• *Lesson 6 (Fault modeling — Section 4.2).* Explicitly modeling faults enabled us to discuss and reason with our e-Bike expert regarding the possible causes for the system's malfunctions. Our e-Bike expert confirmed that this practice helped with reasoning regarding the causes of possible safety breaches. These faults were not explicitly identified, discussed, and documented earlier. It also enabled the analysis of conditions (modeled by triggers) that activate these faults. This enabled the strengthening of the reasoning regarding situations and conditions that are more critical for the e-Bike software and its safety, and thus, more likely leading to faults. It encouraged deeper reflection, which led to the identification of faults that were not considered earlier. For example, the engineer identified additional faults, such as those related to the battery degradation or those involving the inverter, that they had never considered in their design. These faults are added by the engineer pipeline to continuously assess the safety of the e-Bike. To summarize, explicitly modeling faults can benefit CPS applications as they support a more rigorous safety analysis.

• *Lesson 7 (Fault simulation — Section 4.3).* Engineers typically assess the safety of their system by manually defining inputs and visually inspecting the models' outputs. This is a common practice in industrial environments, particularly when preliminary and high-level models of the system (such as the one we considered) are evaluated. However, this practice may lead engineers to overlook certain possible issues and the effects of these faults. Having a framework that enables the automatic analysis of CPS failure, such as the Simulink® Fault Analyzer™, mitigates this problem. The engineer who developed the model utilized in this study acknowledged that the proposed Simulation-driven FMEA approach was significantly beneficial for reasoning about plausible faults and their effects. Using this framework enables the identification of discrepancies between the expected and observed effects for certain faults (Section 4.3). For example, Figure 6 depicts the measured speed of the e-Bike when fault F15 occurs, thereby highlighting the maximum speed reached (in orange) and the regions where significant deceleration is observed (in purple). First, it can be noted that when F15 is active, the e-Bike reaches a maximum speed of approximately 40 km/h, which is above the maximum legal speed limit (25 km/h) [20] for an e-Bike. This limit is exceeded when the fault is active. Moreover, the sections highlighted in purple in Figure 6 exhibit a deceleration of approximately 2 m/s^2 — i.e., about 0.20 g — which is within the same order of magnitude as the deceleration of a commercial aircraft during a normal landing typically between 1.54 m/s^2 and 3.09m/s^2 and approximately one-fifth of the deceleration experienced during an emergency stop (rejected take-off), which can reach 5.14 m/s^2 [61]. However, the critical issue is that such braking is neither expected nor commanded by the user, which poses safety and stability risks for both the vehicle and the rider. To summarize, fault simulation support benefits CPS designers in performing a more systematic and rigorous analysis of the effects of faults.

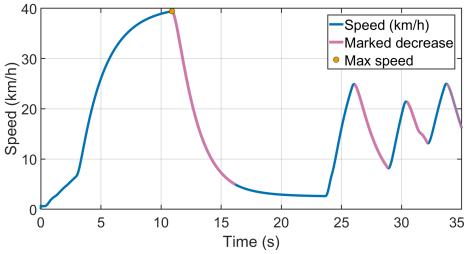
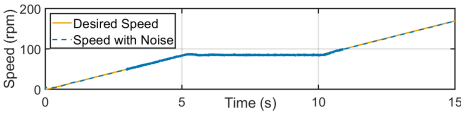


Fig. 6: The measured speed of the e-Bike when fault F15 is activated.

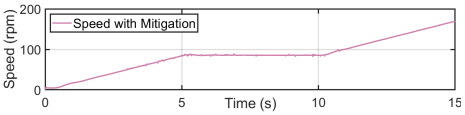
• *Lesson 8 (Development of mitigations).* Our expert not only modeled and analyzed the effects of faults but also initiated reasoning regarding possible mitigations. During our informal discussions preceding the structured interviews, the expert provided possible mitigations that could have been activated when specific faults occurred. For example, for fault F2 (noisy speed sensor), adding a backup sensor is a possible mitigation. The control logic switches the active sensor when it detects that the noise on the monitored speed exceeds a certain threshold. Figure 7 depicts the system behavior when this mitigation is activated. Figure 7a presents the desired speed and the monitored speed by the original (and faulty) sensor, while Figure 7b illustrates the measured speed when the mitigation is applied. Figure 7c depicts the activation status of the mitigation strategy: When mitigation is *On*, the backup sensor is used; when it is *Off*, the speed is read from the original sensor. The mitigation enables the system to switch between the two sensors when the fault is activated. This strategy enables the e-Bike to maintain the desired speed, mitigating the fault effect.

• *Lesson 9 (Mindset instauration).* Nowadays, developing safer and more secure systems is becoming pivotal, particularly considering that new CPS are ubiquitous, more connected, and perform more critical activities. Therefore, it is necessary to instantiate a proper engineering mindset based on rigorous and systematic safety reasoning. While this is a well established mindset in the research community, we found that it may be overlooked by industries. The results of this study confirm that a proper interaction among software engineers, system experts, academics, and industries developing tools for safety reasoning can help create this mindset by sharing knowledge, experiences, and solutions.

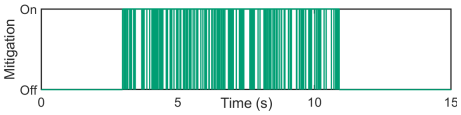
• *Lesson 10 (Software engineering for e-Bikes).* Bikes are traditionally not controlled and regulated by software. Therefore, e-Bike design typically considers both electrical aspects (e.g., the performance of the e-Bike) and physical aspects related to traditional bikes (e.g., material selection and aerodynamics). This traditional view is also reflected in recent analyses of e-Bike safety studies (e.g., [12]) that mostly consider the behavior of the rider, maximum speed of the electrical motors, etc. However, as software becomes a central part of these vehicles, safety analysis cannot be limited to hardware. It must also consider software-related aspects that may lead the vehicle to violate the safety standards and behave improperly. The results of this study reveal that as e-Bikes



(a) The desired speed and measured speed of the e-Bike without mitigation.



(b) The measured speed of the e-Bike with mitigation.



(c) Mitigation activation status.

Fig. 7: Mitigation of fault F2. Two sensors are used to measure the speed, and the backup sensor is activated when the noise exceeds the threshold.

are becoming software systems, software engineering techniques are needed to support rigorous development as well as the safety analysis of these systems.

5.2 Discussion

In this section, we discuss the originality of our work, the relevance of our findings for industrial applications, the significance of our contributions, and the generalizability of our results.

In this paper, we applied the Simulation-driven FMEA approach to a case study in the e-Bike domain — the e-Bike motor controller component. This contribution is *original*: Simulation-driven FMEA is a recent technology [54,59] that has never been applied to the e-Bike domain. Therefore, it improves the current state of the practice in the e-Bike industrial context.

In addition, our results are *relevant* to the e-Bike industry, as e-Bikes are software-intensive systems [62]. These systems are safety-critical and must comply with safety regulations and requirements [58,59]. Our results can also be potentially generalizable to other industrial domains (see Section 5.1).

Further, this work provided *significant* contributions. Compared with previous work assessing Simulation-driven FMEA on a robotic [45] and avionic [54] case studies, this work provides significant Simulink®-specific and Simulink®-agnostic lessons learned that can support the usage of Simulation-driven FMEA in practical scenarios. These lessons learned have been considered valuable by Simulink® experts we interacted with.

Finally, our work provides significant *generalizable* lessons learned. We discussed how our results on fault modeling and simulation apply to contexts beyond the specific industrial context (e-Bikes) of our paper.

5.3 Threats to Validity

The fault models and triggers we designed threaten the *internal validity* of our results; other models and triggers could lead to different results. However, the fact that all models and triggers provided by the Simulink® Fault Analyzer™ have been used mitigates this threat since it ensures that each of them has been considered by our evaluation.

We do not claim that *all* our results can necessarily be generalized to study subjects from other domains. We acknowledge that the selection of our study subject (a model from the e-Bike domain) and the faults could threaten the *external validity* of our results, as it affects their generalizability. We only considered transient and permanent faults, without considering byzantine or intermittent faults. However, the fact that we validated them through an interview with the engineer who developed the model mitigates this threat since it ensures that the faults are representative (at least) for this domain. While we interacted with just one domain expert, the fact that it was the developer of the model under test mitigates this threat since it ensures that he possessed the required knowledge on the system. Moreover, architecture of our case study is representative of common patterns in CPSs including motor control components. Furthermore, our lessons learned result from meetings with Simulink® experts who possess a comprehensive understanding of many CPS domains and needs.

The process we applied in our study is general and can be potentially applied to other study subjects. Future studies will assess if and how our results generalize to other domains and confirm or refute our hypotheses.

6 Related Work

This section (a) summarizes other techniques that support engineers in analyzing the safety of their systems and justifies the choice of Simulation-driven FMEA, and (b) presents other works that analyze the software of e-Bikes.

Support for safety analysis. FMEA [44] is a widely used reliability analysis tool for identifying and mitigating failures in systems, designs, and processes [68]. Simulation-driven FMEA [54] integrates the simulation capabilities of existing tools within FMEA. This technique was initially demonstrated by considering a flight control system of an unmanned ultralight helicopter [54]. Unlike the original work [54], which focuses on presenting the proposed solution, this paper empirically evaluates the effectiveness of Simulation-driven FMEA in an e-Bike case study to provide practitioners with lessons learned.

There are many techniques and approaches similar to FMEA that support engineers in risk evaluation and reliability checks. For example, data-driven

FMEA [18] relies on historical data and system metrics to evaluate and prioritize risks, life cost-based FMEA [53] measures risks for the system life cycle costs, model-based safety analysis [29] extends a shared system model with faults and physical elements to enable automation, and neuro-fuzzy techniques [27] manage the uncertainty and the subjectivity of risk evaluations. Other studies integrate Monte Carlo simulations to manage uncertainty in complex scenarios, integrate FMEA with a widespread reliability and safety model technique (GO methodology) [34], and propose a multiperspective FMEA method based on rough number projection [4] to obtain more accurate and coherent safety evaluations [4]. There are also solutions (e.g., [32,22,23]) that automatically identify failure causes using testing techniques. In our study, we decided to assess Simulation-driven FMEA since it automates the fault analysis and identifies its effects on the system and also because it is integrated with industrial solutions (Simulink® Fault Analyzer™). In addition, recent studies [12,63] advocate a more extensive use of FMEA to analyze software failures, as safety analysis typically overlooks software failure because system designers usually (erroneously) assume that the software does not fail. Our study considers both the cyber (software) and physical failures of the CPS (lower and upper parts of Table 3).

E-Bike software analysis. Software plays a critical role in modern e-Bikes. A recent article [2] clarifies that while most riders "*think the e-Bike difference has to do with the frame, motor, or battery, its usually all in the code*", developing software for eBikes is costly [2]. To design their software, engineers need to have a holistic view of the system. Acquiring this knowledge is complex and time-consuming. We experienced these factors in our project and had to invest significant time acquiring the know-how to work within this domain.

The e-Bike market has a large number of players (over 10,000 companies and a 27.15 USD billion market in 2022 [25]) that typically develop proprietary software. Nevertheless, several open-source e-bike projects exist. For example, the OpenSource project [16] provides software to help users build their e-Bikes.

Recent studies confirm that recent changes in electric vehicles have introduced new security threats and additional software safety concerns [10]. Standards have been developed to regulate the development of e-Bike software systems: The ISO 26262 standard (Part 6) [26] specifies software safety requirements for road vehicles (including e-Bikes), EN 15194 [19] is dedicated to e-Bikes. Furthermore, safety breaches were found in e-Bikes. In 2020, VanMoof updated their app since customers could increase their pedal-assisted power beyond the EU limit of 25 km/h by switching to the US country setting in their app: in the US, e-Bikes are generally capped at 32 km/h (20 mph) [55]. Despite these safety concerns, limited research has considered software engineering techniques for e-Bikes. A recent study considers the problem of automatically generating test cases for e-Bikes [35]. It assessed the effectiveness of existing test case generation solutions (HECATE [22]) in identifying bugs on Simulink models from the e-Bike domain. Our work is significantly different in its scope and objective (supporting the safety analysis of the system), technological support (Simulink® Fault Analyzer™), and evaluation methodology.

7 Conclusions and Future Work

In this paper, we applied Simulation-driven FMEA on an industrial case study in the e-Bike domain. We used Simulink® Fault Analyzer™ as a tool to evaluate FMEA. Our objective was to assess the effectiveness and practicality of Simulation-driven FMEA for assessing and improving software safety in CPSs. We identified 13 realistic faults involving both cyber and physical parts, modeled them using Simulink® fault injection features, and evaluated their effects through simulation. Our results, which we evaluated with a domain expert and Simulink® experts, show that Simulink® Fault Analyzer™ effectively supports fault modeling and effect analysis. For 38% of the cases, simulation results deviated from the engineers initial expectations, revealing unexpected fault impacts, enabling meaningful safety insights and design refinements, and allowing deeper engineering understanding and model improvement. We identified a set of lessons learned, which involved both Simulink®-specific and Simulink®-agnostic considerations, and we presented them to Simulink® experts. Our findings are useful for software engineers who work as Simulink® engineers, use the Simulink® Fault Analyzer™, or work as safety analysts. In future work, we plan to extend our study to other CPSs to investigate the generalization of our conclusion.

Data Availability

The original Simulink® model, the model extended with the faults, the results of the simulation, and the transcripts of our interviews are available online [1].

References

1. Replication Package. https://github.com/foselab/SimulationDrivenFMEA_Bikes (2025), accessed: July 21, 2025
2. Software Takes eBikes to New Heights (2025), https://www.eetimes.eu/software-takes-ebikes-to-new-heights/
3. Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K.: A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. IEEE Transactions on Software Engineering **36**(6), 742–762 (2010). https://doi.org/10.1109/TSE.2009.52
4. An, H., Xu, G., Sun, Z., Chen, W., Pan, J.: Multi-perspective failure mode and effects analysis based on rough number projection. Engineering Failure Analysis **169**, 109192 (2025). https://doi.org/https://doi.org/10.1016/j.engfailanal.2024.109192
5. Badreddin, O., Lethbridge, T.C., Elassar, M.: Modeling Practices in Open Source Software. In: Open Source Software: Quality Verification. pp. 127–139. Springer (2013)
6. Bartocci, E., Mariani, L., Ničković, D., Yadav, D.: FIM: fault injection and mutation for Simulink. In: Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 17161720. ESEC/FSE, ACM (2022). https://doi.org/10.1145/3540250.3558932

7. Boll, A., Brokhausen, F., Amorim, T., Kehrer, T., Vogelsang, A.: Characteristics, Potentials, and Limitations of Open-Source Simulink Projects for Empirical Research. Software and Systems Modeling **20**(6), 2111–2130 (2021)
8. Boll, A., Kehrer, T.: On the Replicability of Experimental Tool Evaluations in Model-Based Development: Lessons Learnt from a Systematic Literature Review Focusing on MATLAB/Simulink. In: Systems Modelling and Management. p. 111130. Springer (2020). https://doi.org/10.1007/978-3-030-58167-1_9
9. Boll, A., Vieregg, N., Kehrer, T.: Replicability of Experimental Tool Evaluations in Model-Based Software and Systems Engineering With MATLAB/Simulink. Innovations in Systems and Software Engineering **20**(3), 209–224 (2024)
10. Chakraborty, S., Jha, S., Samii, S., Mundhenk, P.: Introduction to the Special Issue on Automotive CPS Safety & Security: Part 2. ACM Trans. Cyber-Phys. Syst. **8**(2) (May 2024). https://doi.org/10.1145/3650210
11. Chechik, M.: On safety, assurance, and reliability: a software engineering perspective (keynote). In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 2. ESEC/FSE 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3540250.3569443
12. Cherry, C.R., MacArthur, J.H.: E-bike safety. A review of Empirical European and North American Studies. Light Electric Vehicle Education and Research Initiative (2019)
13. Dialynas, G., Christoforidis, C., Happee, R., Schwab, A.: Rider control identification in cycling taking into account steering torque feedback and sensory delays. Vehicle System Dynamics **61**(1), 200–224 (2023). https://doi.org/10.1080/00423114.2022.2048865
14. Ding, W., Liang, P., Tang, A., Vliet, H.v., Shahin, M.: How Do Open Source Communities Document Software Architecture: An Exploratory Survey. In: International Conference on Engineering of Complex Computer Systems. p. 136145. ICECCS, IEEE Computer Society (2014). https://doi.org/10.1109/ICECCS.2014.26
15. Dyba, T., Kitchenham, B.A., Jorgensen, M.: Evidence-Based Software Engineering for Practitioners. IEEE Software **22**(1), 58–65 (2005)
16. EBike, O.: EBike / EScooter modular DIY OpenSource electronics and software. https://github.com/openSourceEBike (2024), accessed: July 21, 2025
17. Engström, E., Petersen, K.: Mapping Software Testing Practice With Software Testing Research SERP-Test taxonomy. In: IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 1–4 (2015). https://doi.org/10.1109/ICSTW.2015.7107470
18. Ervural, B., Ayaz, H.I.: A fully data-driven FMEA framework for risk assessment on manufacturing processes using a hybrid approach. Engineering Failure Analysis **152**, 107525 (2023). https://doi.org/https://doi.org/10.1016/j.engfailanal.2023.107525
19. European Standard: EN 15194 Cycles — Electrically power assisted cycles — EPAC Bicycles (2023), https://www.en-standard.eu/ilnas-en-15194-cycles-electrically-power-assisted-cycles-epac-bicycles/?srsltid=AfmBUorJ582mAq4zKC1fsLYUqBxr1XM6OvYdcDctO1dRM26dr1eGDz9d, accessed: 21 July 2025
20. European Union: Regulation (EU) No 168/2013 of the European Parliament and of the Council of 15 January 2013 on the approval and market surveillance of two- or three-wheel vehicles and quadricycles Text with EEA relevance. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:02013R0168-20241127 (2024), accessed: July 21, 2025

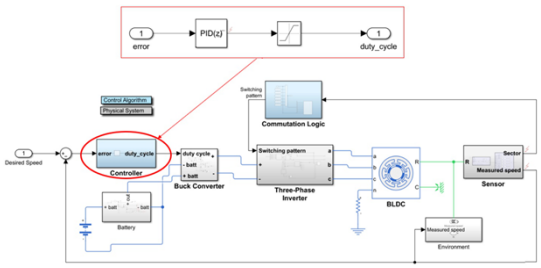
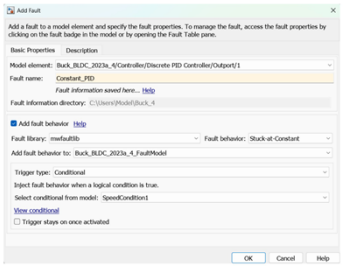
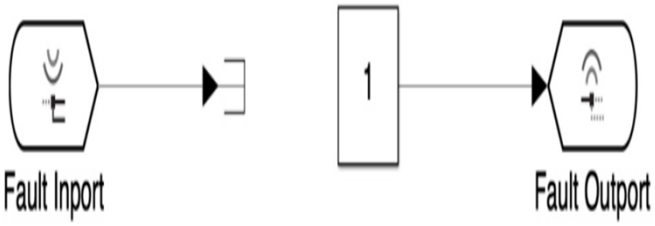
21. Ford Motor Company: Ford Motor Company Customer-specific Requirements, for Use with ISO/TS 16949: 2002 (2003)
22. Formica, F., Fan, T., Rajhans, A., Pantelic, V., Lawford, M., Menghi, C.: Simulation-based testing of simulink models with test sequence and test assessment blocks. IEEE Transactions on Software Engineering **50**(2), 239–257 (2023)
23. Formica, F., Petrunti, N., Bruck, L., Pantelic, V., Lawford, M., Menghi, C.: Test Case Generation for Drivability Requirements of an Automotive Cruise Controller: An Experience with an Industrial Simulator. In: Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 19491960. ESEC/FSE 2023, Association for Computing Machinery (2023). https://doi.org/10.1145/3611643.3613894
24. Goddard, P.: Software FMEA techniques. In: Annual Reliability and Maintainability Symposium. International Symposium on Product Quality and Integrity (Cat. No.00CH37055). pp. 118–123 (2000). https://doi.org/10.1109/RAMS.2000.816294
25. Insights, F.B.: E-Bike Drive Unit Market Size, Share & COVID-19 Impact Analysis, By Product Type (Mid-drive Motors and Hub Motors), By Application (OEM and Aftermarket), and Regional Forecasts, 2023-2030. https://www.fortunebusinessinsights.com/e-bike-drive-unit-market-107520 (2024), accessed: July 21, 2025
26. International Standard Organization: ISO 26262-6:2018 — Road vehicles — Functional safety — Part 6: Product development at the software level (2024), https://www.iso.org/standard/68388.html, accessed: 21 July 2025
27. Ivanan, J., Lisjak, D., Pavleti, D., Kolar, D.: Improvement of Failure Mode and Effects Analysis Using Fuzzy and Adaptive Neuro-Fuzzy Inference System. Machines **11**(7) (2023). https://doi.org/10.3390/machines11070739
28. Joshi, A., Heimdahl, M.P., Miller, S.P., Whalen, M.W.: Model-based safety analysis. Tech. rep. (2006)
29. Joshi, A., Whalen, M.: Model-based safety analysis final report (02 2006)
30. Kitchenham, B., Dyba, T., Jorgensen, M.: Evidence-Based Software Engineering. In: International Conference on Software Engineering. pp. 273–281 (2004). https://doi.org/10.1109/ICSE.2004.1317449
31. Krishnan, R., Bhada, S.V.: An Integrated System Design and Safety Framework for Model-Based Safety Analysis. IEEE Access **8**, 146483–146497 (2020). https://doi.org/10.1109/ACCESS.2020.3015151
32. Le Tolguenec, P.A., Rachelson, E., Besse, Y., Teichteil-Koenigsbuch, F., Schneider, N., Waeselynck, H., Wilson, D.: Exploration-Driven Reinforcement Learning for Avionic System Fault Detection (Experience Paper). In: SIGSOFT International Symposium on Software Testing and Analysis. p. 920931. ISSTA, ACM (2024). https://doi.org/10.1145/3650212.3680331
33. Lisagor, O., Kelly, T., Niu, R.: Model-based safety assessment: Review of the discipline and its challenges. In: International Conference on Reliability, Maintainability and Safety. pp. 625–632. IEEE (2011)
34. Liu, L., Fan, D., Wang, Z., Yang, D., Cui, J., Ma, X., Ren, Y.: Enhanced GO methodology to support failure mode, effects and criticality analysis. Journal of Intelligent Manufacturing **30**, 1451–1468 (2019). https://doi.org/10.1007/s10845-017-1336-0
35. Marzella, M., Bombarda, A., Minervini, M., Bisceglia, N.M., Gargantini, A., Menghi, C.: Test Case Generation for Simulink Models: An Experience from the E-Bike Domain. In: Search-Based Software Engineering. Springer Nature Switzerland, Cham (2025)

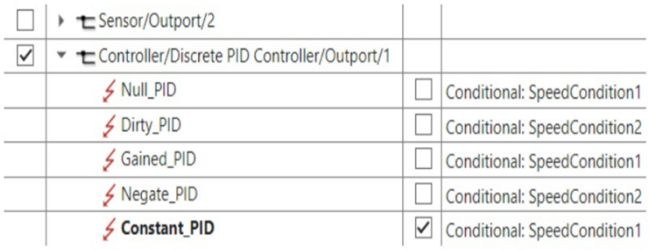
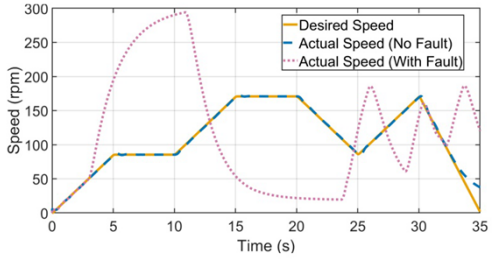
36. MathWorks: Blocks that Support Fault Modeling, https://www.mathworks.com/help/simscape/ug/block-support.html, accessed: June 22, 2025
37. MathWorks: Simscape Model and simulate multidomain physical systems, https://www.mathworks.com/products/simscape.html, accessed: July 21, 2025
38. MathWorks: Simulate Models with Faults by Using the Multiple Simulations Panel, https://www.mathworks.com/help/fault-analyzer/ug/simulate-models-with-faults.html, accessed: July 21, 2025
39. Mathworks: Simulink Fault Analyzer (2003), https://www.mathworks.com/products/simulink-fault-analyzer.html
40. MathWorks: How Acceleration Modes Work. https://www.mathworks.com/help/simulink/ug/how-the-acceleration-modes-work.html (2024), accessed: July 21, 2025
41. MathWorks: Variable Resistor. https://www.mathworks.com/help/simscape/ref/variableresistor.html (2024), accessed: July 21, 2025
42. Melo, S.M., Carver, J.C., Souza, P.S., Souza, S.R.: Empirical Research on Concurrent Software Testing: A Systematic Mapping Study. Information and Software Technology **105**, 226–251 (2019)
43. Mezhuyev, V., Al-Emran, M., Ismail, M.A., Benedicenti, L., Chandran, D.A.P.: The Acceptance of Search-Based Software Engineering Techniques: An Empirical Evaluation Using the Technology Acceptance Model. IEEE Access **7**, 101073–101085 (2019). https://doi.org/10.1109/ACCESS.2019.2917913
44. MILP1629: Procedures for Performing a Failure Mode Effects and Criticality Analysis. U.S. Department of Defense. (1949)
45. Miraglia, G., Bimbi, M., Nanjundappa, M.: Enhancing Mobile Robot Safety Evaluation with Simulation-Driven Model-Based Safety Analysis. In: RSS Safe Autonomy Workshop (2024), https://sites.google.com/view/rss2024-safe-autonomy/home
46. Mode, F.: Effects and Criticality Analysis (FMECA). Reliability Analysis Center (1993)
47. Modelwise: Automatic assessment of an industrial safety-critical system (2025), https://modelwise.ai/automatic-assessment-of-an-industrial-safety-critical-system/
48. Murgano, E., Caponetto, R., Pappalardo, G., Cafiso, S.D., Severino, A.: A Novel Acceleration Signal Processing Procedure for Cycling Safety Assessment. Sensors **21**(12) (2021). https://doi.org/10.3390/s21124183
49. Office of manned space flight, Apollo program, Apollo Reliability and Quality Assurance Office: Procedure for Failure Mode, Effects and Criticality Analysis (FMECA) (1966)
50. Panichella, A., Kifetew, F.M., Tonella, P.: A large scale empirical comparison of state-of-the-art search-based test case generators. Information and Software Technology **104**, 236–256 (2018)
51. Rebello, S., Goyal, N.K.: Software system reliability and safety assessment: an extended FMEA approach. International Journal of Reliability and Safety **4**(4), 366 (2010). https://doi.org/10.1504/ijrs.2010.035575
52. Reifer, D.J.: Software Failure Modes and Effects Analysis. IEEE Transactions on Reliability **R-28**(3), 247–249 (1979). https://doi.org/10.1109/TR.1979.5220578
53. Rhee, S.J., Ishii, K.: Using cost based FMEA to enhance reliability and serviceability. Advanced Engineering Informatics **17**(3), 179–188 (2003). https://doi.org/https://doi.org/10.1016/j.aei.2004.07.002

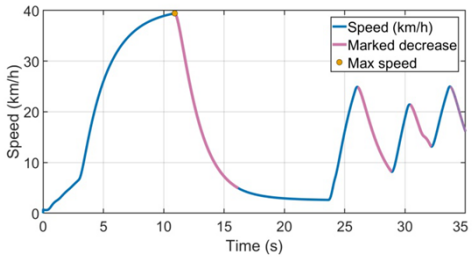
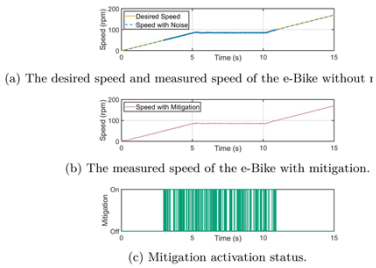
54. Rhein, J., Bimbi, M., Miraglia, G., Holzapfel, F.: Simulation-Driven Failure Modes and Effects Analysis of Flight Control System Architectures. In: Digital Avionics Systems Conference (DASC). pp. 1–10. IEEE (2024)
55. Ricker, T.: Cheating VanMoof e-bikes will be slowed outside the US. https://www.theverge.com/2020/11/10/21558235/vanmoof-slows-s3-x3-europe-japan-speed-limit (2024), accessed: November 7, 2024
56. Rodriguez, A.D., Newman, T., Dearstyne, K.R., Cleland-Huang, J.: SAFA: A Tool for Supporting Safety Analysis in Evolving Software Systems. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3551349.3559535
57. Sayyad, A.S., Goseva-Popstojanova, K., Menzies, T., Ammar, H.: On Parameter Tuning in Search Based Software Engineering: A Replicated Empirical Study. In: International Workshop on Replication in Empirical Software Engineering Research. p. 8490. RESER, IEEE (2013). https://doi.org/10.1109/RESER.2013.6
58. Schepers, P., Wolt, K.K., Fishman, E.: The safety of e-bikes in The Netherlands. International Transport Forum Discussion Paper 2018-02, Paris (2018). https://doi.org/10.1787/21de1ffa-en
59. Schleinitz, K., Petzoldt, T., Franke-Bartholdt, L., Krems, J.F., Gehlert, T.: The German Naturalistic Cycling Study — Comparing cycling speed of riders of different e-bikes and conventional bicycles. Safety Science **92**, 290297 (2017). https://doi.org/10.1016/j.ssci.2015.07.027
60. Shrestha, S.L., Chowdhury, S.A., Csallner, C.: Replicability Study: Corpora For Understanding Simulink Models & Projects. In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–12. IEEE (2023). https://doi.org/10.1109/ESEM56168.2023.10304867
61. SKYbrary: Deceleration on the Runway. https://skybrary.aero/articles/deceleration-runway (2025)
62. Stories, M.C.: Bosch eBike Systems Develops Electric Bike Controller with Model-Based Design. https://www.mathworks.com/company/user_stories/bosch-ebike-systems-develops-electric-bike-controller-with-model-based-design.html (2024), accessed: November 7, 2024
63. Thieme, C.A., Mosleh, A., Utne, I.B., Hegde, J.: Incorporating software failure in risk analysis. Reliability Engineering & System Safety **198**, 106804 (2020)
64. Times, E.: Software Takes eBikes to New Heights. https://www.eetimes.eu/software-takes-ebikes-to-new-heights/ (2025), accessed: July 21, 2025
65. Tran, C.D., Kuchar, M., Sobek, M., Sotola, V., Dinh, B.H.: Sensor Fault Diagnosis Method Based on Rotor Slip Applied to Induction Motor Drive. Sensors **22**(22) (2022). https://doi.org/10.3390/s22228636
66. Valle, P., Riccio, V., Arrieta, A., Tonella, P., Arratibel, M.: An industrial experience report on applying search-based boundary input generation to cyber-physical systems. Empirical Software Engineering **30**(4), 112 (2025)
67. Valyayev, D., Mukasheva, A., Yedilkhan, D., Aigerim, B., Mukhammejanova, D.: FMEA variables of software failure risks based on journals and metrics. In: 2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST). pp. 302–307 (2024). https://doi.org/10.1109/SIST61555.2024.10629469
68. Wang, Z., Gao, J.M., Wang, R.X., Chen, K., Gao, Z.Y., Zheng, W.: Failure Mode and Effects Analysis by Using the House of Reliability-Based Rough VIKOR Approach. IEEE Transactions on Reliability **67**(1), 230–248 (2018). https://doi.org/10.1109/TR.2017.2778316

69. Zhu, J., Wang, L., Han, X.: Safety and Performance, Why not Both? Bi-Objective Optimized Model Compression toward AI Software Deployment. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3551349.3556906

Alternative Texts for Your Images, Please Check and Correct them if Required

Page no	Fig/Photo	Thumbnail	Alt-text Description
4	fig1.jpg		Diagram illustrating a control system for a Brushless DC (BLDC) motor. The flow chart begins with a desired speed input, leading to a controller that processes error and duty cycle through a PID controller. The output is sent to a buck converter, which adjusts the power from a battery. The commutation logic determines the switching pattern for a three-phase inverter, which drives the BLDC motor. A sensor measures the motor speed, feeding back to the controller to adjust the system. Key components are highlighted, showing the interaction between control algorithms and physical systems.
5	fig1.jpg		Dialog box titled "Add Fault" for configuring fault properties in a model. It includes fields for specifying the model element, fault name, and fault information directory. Options to add fault behavior are available, with selections for fault library, fault behavior type, and model to apply the behavior. The trigger type is conditional, with a condition named "SpeedCondition1." Buttons for "OK," "Cancel," and "Help" are at the bottom.
6	fig1.jpg		Flow chart depicting a process from "Fault Inport" to "Fault Outport." The flow begins with an input symbol labeled "Fault Inport," connected by an arrow to a central box labeled "1." Another arrow leads from this box to an output symbol

Page no	Fig/Photo	Thumbnail	Alt-text Description
			labeled "Fault Outport." The symbols include icons representing electrical components.
7	fig1.jpg		Table displaying a hierarchical structure of a control system. It includes checkboxes and icons for different components. The main categories are "Sensor/Output/2" and "Controller/Discrete PID Controller/Output/1," with the latter being selected. Subcategories under the controller include "Null_PID," "Dirty_PID," "Gained_PID," "Negate_PID," and "Constant_PID," with "Constant_PID" highlighted in bold and selected. Each subcategory is associated with conditional speed conditions, with "Constant_PID" linked to "Conditional: SpeedCondition1," which is checked.
8	fig1.jpg		Line graph showing speed in revolutions per minute (rpm) over time in seconds. Three lines represent different conditions: a solid yellow line for desired speed, a dashed blue line for actual speed with no fault, and a dotted pink line for actual speed with a fault. The graph illustrates variations in speed under these conditions, highlighting discrepancies between desired and actual speeds, especially when faults are present.

Page no	Fig/Photo	Thumbnail	Alt-text Description
15	fig1.jpg		Line chart showing speed in kilometers per hour over time in seconds. The blue line represents speed, peaking at 40 km/h around 10 seconds, marked by an orange dot. A pink line indicates a marked decrease in speed after the peak. The chart includes fluctuations in speed from 15 to 35 seconds. A legend identifies the lines and peak point.
16	fig1.jpg	 (a) The desired speed and measured speed of the e-Bike without (b) The measured speed of the e-Bike with mitigation. (c) Mitigation activation status.	Three-panel figure showing e-Bike speed and mitigation data over time: (a) Line chart comparing desired speed and measured speed with noise over 15 seconds. The desired speed is a steady line, while the speed with noise fluctuates slightly. (b) Line chart showing measured speed with mitigation over 15 seconds, indicating a smoother increase compared to the noisy data. (c) Bar chart displaying mitigation activation status over 15 seconds, with frequent activations indicated by vertical bars.
24	fig1.jpg		Creative Commons license symbols indicating "Attribution-NonCommercial-NoDerivs" (CC BY-NC-ND). The symbols include a circle with "CC," a person icon, a crossed-out dollar sign, and an equals sign.