Thesis for The Degree of Doctor of Philosophy

From Chance to Choice: Strategies to Attaining Resilience in Cyber-Physical Systems

Ricardo Diniz Caldas

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2024

From Chance to Choice: Strategies to Attaining Resilience in Cyber-Physical Systems

Ricardo Diniz Caldas

Copyright ©2024 Ricardo Diniz Caldas
except where otherwise stated.
All rights reserved.

ISBN 978-91-8103-133-1
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5591.
ISSN 0346-718X

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2024.

"Tenho o privilégio de não saber quase tudo,
e isso explica
o resto."

Manoel de Barros

Abstract

Background: Autonomy is a key attribute of cyber-physical systems engineered to achieve human-machine coexistence and collaboration toward human-centered goals. To be trusted, autonomous systems must operate resiliently, yet designing and verifying resilient behavior remains an open challenge. Resilient cyber-physical systems should avoid, withstand, recover from, and adapt to adversities arising from computational, network, or environmental disruptions. Wearable biosensors are a prime example of cyber-physical systems that must operate resiliently. Such a healthcare monitoring system could fail during a network outage or erroneous sensor data, endangering lives. A resilient healthcare monitoring system, with redundant paths and adaptive capacity, ensures continuous monitoring and timely alerts despite disruptions.

Objective: This thesis aims to equip developers and quality assurance teams with strategies for attaining resilience in cyber-physical systems, ensuring that resilience is engineered rather than attained by coincidence. Attaining resilience in cyber-physical systems entails justified adaptation to overcome unknown stimuli, ever-changing objectives, and deprecated components. Software as a tool for self-management is crucial for dealing with uncertainty. Achieving resilience is challenging since unexpected effects may emerge during execution, requiring runtime decision-making rather than design time.

Method: The strategies are rooted in publications in software engineering, self-managed and adaptive systems, robotics, and transportation. They encompass quantitative and qualitative research that follows a design science research methodology.

Results: The thesis introduces seven strategies for attaining resilience, including: (i) best practices for runtime assessment, (ii) tools to manage interactions among diverse and smart agents, (iii) methods for uncertainty mitigation at the code level, runtime adaptation, and explanation of property violations, and (iv) exemplars that serve as models to advance resilience research. Our results demonstrate that resilience is achieved through systematic design and runtime decision-making, ensuring that systems consistently meet operational goals.

Conclusion: This study advocates for resilience as a strategic goal, highlighting its importance as a foundational discipline within software engineering for cyber-physical systems. The findings benefit both researchers and practitioners, emphasizing resilience engineering as essential for the future of autonomous systems.

Keywords

Resilience Attainment, Strategies, Cyber-Physical Systems, Software Engineering, Self-Adaptation, Uncertainty

Acknowledgment

The work presented in this thesis results from years of dedication, sustained effort, and the unwavering support of many individuals and institutions. Without the guidance of my mentors, the encouragement of friends and family, and financial support, this journey would not have been possible.

I would like to extend my deepest appreciation to my supervisors, Patrizio Pelliccione, Thorsten Berger, and Daniel Strüber. Your insights and mentorship have been invaluable, shaping the research and vision of this thesis. Even more importantly, your tutoring over these five years has been instrumental in developing my research abilities, and for this, I am eternally grateful. My sincere thanks also go to my examiner, Gerardo Schneider, for his constructive feedback, which significantly enhanced the quality of my work. I am also profoundly grateful to Genaína Rodrigues for her inspiring contributions to my research growth. I also extend my gratitude to Camila who attentively helped me to untangle the knots and navigate with more serenity.

To everyone I've met over the past five years who helped make Gothenburg feel like home, thank you. To the friends I met through Chalmers–Razan, Krishna, Bea, Sofia, Malsha, Wardah, Cristy, Afonso, Teodor, Hamdy, Mazen, Ranim, Amna, Habib, Tayssir, and Sabina–and to the entire IDSE division, I am deeply grateful for the supportive and memorable environment you created. I also want to thank the friends I met during my research visit to Italy–Rickson, Tony, Eva, Matteo, Tiziano, and Chris–for their warm hospitality. Additionally, I am grateful to friends whose paths crossed mine beyond academia–Zabou, Anton, Kristina, Natasa, Juan, Jelena, Giovanni, and with special regard to Saulo and Gabriel Moisés. Finally, my heartfelt gratitude goes to those across the ocean, whose love and encouragement sustained me in times of *saudade*, my parents, Ricardo and Léa, my siblings, Gabriela and Leonardo, my friends, and, last but not least, my partner, best friend, and beloved, Amanda.

Finally, I am grateful to WASP, my financial supporter, for enabling this work and providing opportunities to connect with researchers committed to advancing knowledge and making a positive impact. This research is financially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

List of Publications

This thesis is based on the following publications:

[A] R. Caldas, J. A. P. García , M. Schiopu , P. Pelliccione, G. Rodrigues, T. Berger, "Runtime Verification and Field-based Testing for ROS-based Robotic Systems" *Trans. Softw. Eng. (TSE) 50(10): 2544 - 2567, IEEE, 2024.* DOI: 10.1109/TSE.2024.3444697.

[B] G. S. Rodrigues, R. Caldas, G. Araujo, V. de Moraes, G. Rodrigues, P. Pelliccione "An Architecture for Mission Coordination of Heterogeneous Robots" *J. Syst.Softw.(JSS) 191:111363 (2022), ACM, 2022.* DOI:10.1016/j.jss.2022.111363

[C] R. Queiroz, D. Sharma, R. Caldas, K. Czarnecki, S. García, T. Berger, P. Pelliccione "A Driver-Vehicle Model for ADS Scenario-based Testing" *Trans. on Intell. Transp. Syst. (ITS) 25(8): 8641-8654, IEEE, 2024.* DOI: 10.1109/TITS.2024.3373531

[D] R. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. N. Vogel, P. Pelliccione "A Hybrid Approach Combining Control Theory and AI for Engineering Self-Adaptive Systems" *Proceedings of the 15th Intl. Symp. on Softw. Eng. for Adapt. and Self-Managing Syst. (SEAMS), IEEE/ACM, 2020.* DOI: 10.1145/3387939.3391595

[E] M. Rizwan, C. Reichenbach, R. Caldas, M. Mayr, V. Krueger, "EzSkiROS: Enhancing Robot Skill Composition with Embedded DSL for Early Error Detection" *Frontiers in Robotics and AI (RAI), Frontiers, 2024.* DOI: 10.3389/frobt.2024.1363443

[F] J. P. C. de Araujo, G. N. Rodrigues, M. Carwehl, T. Vogel, L. Grunske, R. Caldas, P. Pelliccione, "Explainability for Property Violations in Cyber-Physical Systems: An Immune-Inspired Approach" *IEEE Softw. 41(5): 43-51, IEEE, 2024.* DOI: 10.1109/MS.2024.3387289

[G] G. Araujo, R. Caldas, F. Formica, G. Rodrigues, P. Pelliccione, C. Menghi "Search-based Trace Diagnostic" *(submitted).*

[H] E. B. Gil, R. Caldas, A. Rodrigues, G. L. G. da Silva, G. N. Rodrigues, P. Pelliccione "Body Sensor Network: A Self-Adaptive System Exemplar in the Healthcare Domain" *Proceedings of the 16th Intl. Symp. on Softw. Eng. for Adapt. and Self-Managing Syst. (SEAMS). IEEE/ACM, 2021.* DOI: 10.1109/SEAMS51251.2021.00037

[I] M. Askarpour, C. Tsigkanos, C. Menghi, R. Calinescu, P. Pelliccione, S. García, R. Caldas, T. J. von Oertzen, M. Wimmer, L. Berardinelli, M. Rossi, M. M. Bersani, G. S. Rodrigues "RoboMAX: Robotic Mission Adaptation eXemplars" *Proceedings of the 16th Intl. Symp. on Softw. Eng. for Adapt. and Self-Managing Syst. (SEAMS), IEEE/ACM, 2021.* DOI: 10.1109/SEAMS51251.2021.00040

Other publications

The following publications are related to my PhD studies or are currently in submission/under revision. They are not appended to this thesis due to their contents overlapping those of appended publications or their content not related to the thesis.

[a] Silva, S., Caldas, R., Pelliccione, P., Bertolino, A. “Different Approaches for Testing Body Sensor Network Applications.” *Journal of Systems and Software (JSS), ACM, (submitted and currently under revision)*

[b] M. Rizwan, R. Caldas, C. Reichenbach, M. Mayr, “EzSkiROS: A Case Study on Embedded Robotics DSLs to Catch Bugs Early” *Proceedings of the 5th Intl. Workshop on Robotics Software Engineering (RoSE), IEEE/ACM, 2023. DOI: 10.1109/RoSE59155.2023.00014.*

[c] R. Caldas, R. Ghzouli, A. V. Papadopoulos, P. Pelliccione, D. Weyns, T. Berger “Towards Mapping Control Theory and Software Engineering Properties using Specification Patterns” *2nd International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS). IEEE, 2021. DOI: 10.1109/ACSOS-C52956.2021.00067.*

[d] G. F. Solano, R. Caldas, G. Rodrigues, T. Vogel, P. Pelliccione. “Taming uncertainty in the assurance process of self-adaptive systems: a goal-oriented approach.” *In Proceedings of the 14th Intl. Symp. on Softw. Eng. for Adapt. and Self-Managing Syst. (SEAMS), IEEE, 2019. DOI: 10.1109/SEAMS.2019.00020*

[e] A. Rodrigues R. Caldas, G. Rodrigues, T. Vogel, and P. Pelliccione. “A learning approach to enhance assurances for real-time self-adaptive systems.” *In Proceedings of the 13th Intl. Conference on Softw. Eng. for Adapt. and Self-Managing Syst. (SEAMS), IEEE/ACM, 2018. DOI: 10.1145/3194133.3194147*

Research Contribution

This thesis is a composite of 9 appended publications. All publications report on work performed in coordination between more than three researchers in various capacities. This section, thus, precisely describes the thesis author's contributions to the appended publications. The contributions are classified using the Contributor Roles Taxonomy (CRediT)[1].

Overall, the thesis author was leading in Papers A, D, and H. In the other papers, B, C, E, F, G, and I, he contributed to various research activities with a consistent focus on the empirical assessment of the work. Next, we describe the author's contributions to papers **A**–**I**.

- **Paper A**: Conceptualization, Methodology, Literature Review, Data Curation, Validation, Visualization, Writing – Original Draft.
- **Paper B**: Conceptualization (with G. S. Rodrigues), Experimental Design, Data Preparation, Visualization, Results Reporting.
- **Paper C**: Design Artifact Implementation, Experiment Design, Data Collection, Visualization, Writing – Review & Editing.
- **Paper D**: Conceptualization , Artifact Implementation, Experimental Design, Data Collection, Visualization, Writing – Original Draft.
- **Paper E**: Conceptualization (with M. Rizwan), Pair Programming, Experimental Planning, Data Analysis, Visualization, Writing – Related Work, Review & Editing.
- **Paper F**: Algorithm Discussion, Writing – Manuscript Review.
- **Paper G**: Conceptualization (with G. Araujo), Solution Design, Technology Prototyping, Experiment Design, Data Analysis, Writing – Related Work, Review & Editing.
- **Paper H**: Conceptualization, Artifact Implementation, Visualization, Writing – Original Draft, Packaging.
- **Paper I**: Specifications Design, Artifact Documentation, Writing – Manuscript Review.

[1]https://casrai.org/credit/

Contents

Chapter 1

Introduction

Resilience is a core principle to engineering human-centric computing systems. The research presented in this thesis sheds light on resilience within cyber-physical systems. It offers strategies to equip researchers and practitioners with the knowledge, methods, and tools necessary for advancing computing systems for humanity. In this chapter, we describe the problem of attaining resilience in cyber-physical systems, summarize our studies to tackle the problem and frame our results in the context of the research questions, casting our contributions in the format of strategies.

1.1 Problem Formulation

Cyber-physical systems stand at the intersection of the digital and physical worlds, increasingly supporting individuals and groups in their social and professional endeavors. In contrast to traditional embedded systems, cyber-physical systems are often designed as networks of interactive and dynamic elements [1], which include healthcare systems, mobility systems, process control systems, and collective robotics. Such applications directly reflect strategic economic and social development areas, i.e., transport, energy, well-being industry, and infrastructure, and thus, highlight the fundamental importance of rigorous engineering of cyber-physical systems.

Definition – Cyber-Physical Systems

Cyber-physical systems are *"engineered systems built from, and that depends upon, the seamless integration of computational algorithms and physical components"* [2]

Wearable biosensors are a prime example of cyber-physical systems since they are software-intensive systems deeply intertwined with the physiological factors as definitive features of living beings. Such technology has revolutionized personal health management by enabling continuous monitoring of vital signs, early detection of diseases, and personalized treatment plans, improving health outcomes, increasing life expectancy, and reducing healthcare costs [3, 4].

The problem of integrating computational algorithms and physical components is not new. It was pioneered by Turing A. M.'s discourse about abstract and physical machines [5] and Shannon C. E.'s mathematical models of physical systems as the semantics of messages in a generic communication system [6]. The underpinning of such development led to the advent of digital computers in the 20^{th} century and further underwent a drastic transformation with the latest advances in sensors, actuators, and processing units [7]. In the early 2000s, investigations to tackle the integration between computing and physical components ultimately resulted in the birth and rise of cyber-physical systems [8]. However, due to a lack of trust, cyber-physical systems struggle to be integrated into everyday environments such as homes, hospitals, and hotels.

As a catalyst for trustworthy cyber-physical systems development, the European roadmap and strategy for cyber-physical systems (CyPhERS) lists five technological challenges to engineering cyber-physical systems [9].

- Interoperability [10]: Integrating components from different suppliers is especially challenging in the presence of legacy parts and when aiming at the continuous update of the cyber-physical system [11, 12].

- Autonomy: The large scale and complexity of cyber-physical systems amplify the challenge of increasing the level of automatic behavior in the individual components participating in the system and their collaboration [13].

- Privacy: When there is the need to collect and process data along the cross-organization chains of services, data become available to participating parties of these services, and this exacerbates issues on the security and privacy of sensitive information [14, 15].

- Dependability: Faults, and, in general, changes, threaten the integrity of the services provided by cyber-physical systems and might compromise their persistence [16].

- Uncertainty: Dealing with imprecise or incomplete information and seamlessly operating in the face of the unknown [17] threaten confidence in the provided service [18].

Autonomy, dependability, and uncertainty mitigation underscore the critical role of persistent service delivery in the face of change, particularly during execution, which are hallmarks of resilient systems [19]. In this context, resilience emerges as a fundamental trait of cyber-physical systems, consisting of the core technological imperative for developing trustworthy next-generation systems [20]. Building upon the requirements outlined by CyPhERS, we postulate that resilience is indispensable for the successful deployment and practical utility of cyber-physical systems. Nevertheless, systematic approaches to attaining resilience remain under-explored, especially within computer science and engineering.

Whether resilience is a desired property is out of the question; as a matter of fact, resilience is a desirable property for many kinds of systems of all natures and fields, for example, material sciences [21], ecological systems [22] or social systems [23, 24].

In the most general sense, system efficiency is the ability to produce something with minimal effort and resources, and resiliency-driven approaches tend to balance short-term efficiency with longer-term effectiveness [25]. Computer science is prone to the efficiency-driven paradigm. The field of algorithms, for example, focuses almost solely on ensuring efficiency and minimizing computational cost [26, 27]. While research in software engineering focuses on producing tools to support rapid coding and verification [28–31] and cost-effective teaming to efficient software development processes [32–34].

We, though, question the aim of efficiency-driven approaches in opposition to leveraging a balance between resilience and short-term efficiency in algorithms, development processes, testing pipelines, and requirements elicitation techniques [35, 36]. As P.G. Neumann insightfully notes, *"there is much to be gained from farsighted thinking that also enables short-term achievements."* [37], underscoring the value of engineering with a perspective that extends beyond immediate gains. Ackley D., for instance, builds an argument for engineering algorithms with focus robustness by showcasing that the praised quicksort, while efficient, proves less reliable than bubblesort under system faults [38, 39]. Resiliency-driven software engineering has been the particular focus of several research groups: Andersson J. and Mirandola R. et al., for instance, developed a conceptual framework that formally defines resilience as a property of computing systems [40, 41]. In complement, Calinescu R. developed probabilistic tools to collect assurances on correctness towards resilient operation [42, 43]. Moreover, Weyns D. leverages resilience in the context of architecture for self-adaptation and self-evolution [44]. Finally, Cámara et al. focuses on probabilistic model checking for resilience measures evaluation [45–47].

Resiliency-driven engineering in computer science and engineering is not yet a reality for practitioners [48]. To leap from efficiency-driven to resiliency-driven engineering, developers and quality assurance teams first need to better understand strategies for using runtime data or runtime conditions in the systems' development and quality assurance. Second, they need to be able to apply known techniques to attain resilience in systems with heterogeneous components that interact with other autonomous components of the system or the environment. Third, they must be equipped with techniques to acknowledge and tame uncertainty at the code level, during runtime, and after execution. In summary, this thesis aims at equipping developers and quality assurance teams with strategies for attaining resilience in cyber-physical systems.

Thesis Statement

Resilience should not be attained by coincidence; it should be by strategy.

More specifically, within cyber-physical systems, attaining resilience involves rethinking the system's design with a focus on dynamic environments. Cyber-physical systems are subject to interactions with (human) users and operators [49, 50], changing needs [51], and diverse and smart components joining and leaving the system [12]. The software controller must automatically manage its dynamicity and uncertainty, using, for instance, self-adaptation to continuously adequate the system behavior to the specification goals [52–54]. Accordingly, the cyber-physical system control software must also ensure that the system requirements are continuously satisfied [55, 56].

1.2 Research Objective and Research Questions

This research addresses key challenges in attaining resilience in cyber-physical systems. Our approach follows a problem-solution model, addressing the overarching objective through specific research questions (RQs). We define four questions (**RQ1**–**RQ4**) to guide this work.

RQ1 explores runtime assessment for resilience in cyber-physical systems. Next, **RQ2** and **RQ3** focus on knowledge from the literature to tackle system construction challenges. Specifically, **RQ2** examines the engineering of cyber-physical systems with diverse and smart components, while **RQ3** investigates resilience under uncertain conditions. Lastly, **RQ4** considers the role of research artifacts in supporting future work from a community perspective.

1.2.1 Current Practices, Methods, and Tools

Cyber-physical systems operate in real-world environments that introduce unpredictable errors, making it unrealistic to ensure their correct behavior solely at design time [57, 58]. These systems interact with and change their environments, encountering uncertainty and variability that challenge accurate modeling [59]. Assuring such real-world scenarios, especially in fields like service robotics, remains one of the most complex tasks for software engineers [60]. Assurance often requires runtime assessment to capture actual operational data and conditions.

Runtime assessment methods, such as runtime monitoring [61–63] and field-based testing [64, 65], improve observability and control, helping identify changes in the system or environment. However, a gap exists between research outputs and practitioners' needs, particularly in terms of using these techniques to support resilience in cyber-physical systems.

Research Question 1 (RQ1)

How are the current practices, methods, and tools enabling runtime assessment to attain resilience in cyber-physical systems?

RQ1 aims to identify resources that guide software developers in using runtime assessment to enhance resilience in cyber-physical systems. To address this, we analyzed the scientific literature and mined software repositories, developing a list of best practices and recommendations to support developers and QA teams in conducting field-based testing and runtime verification."

1.2.2 Diverse and Smart Cyber-Physical Systems

In cyber-physical systems composed of diverse and smart components, achieving resilience is a complex challenge due to the systems' inherent heterogeneity and distributed nature [66]. These systems are made up of nodes that often differ in hardware capabilities, software configurations, or communication protocols. Ensuring resilience across such varied elements requires approaches that can handle these differences while maintaining system performance.

Research Question 2 (RQ2)

How to attain resilience in cyber-physical systems with diverse and smart agents?

RQ2 seeks to identify strategies for attaining resilience in cyber-physical, particularly those with diverse and smart components. To address this, we examine resilience from two perspectives: (**RQ2.1**) resilience strategies targeting the design stage, ensuring that systems operate robustly despite varied component characteristics, and (**RQ2.2**) quality assurance methods for validating resilience in environments with smart agents.

During the design phase, developers can embed resilience strategies to proactively detect emergent faults, prevent system degradation, and enable failure recovery. Strategies such as redundancy, isolation, and encapsulation [67] have proven effective for robust distributed systems, where robustness serves as a specific form of resilience[1]. Among these, redundancy has shown particularly strong results [68], though it requires complex coordination and consensus among system agents to ensure uninterrupted and correct service delivery [69]. However, coordinating diverse agents in cyber-physical systems is especially challenging due to the varied capabilities and operational constraints of each component [70, 71]. For example, in a team of heterogeneous robots conducting a search-and-rescue mission, each robot's specific capabilities must be considered when tasks are assigned or reallocated. This reallocation must be adaptive yet scalable, as task coordination across diverse agents can significantly impact the system's long-term performance and resilience. Ultimately, principles for automating the coordination of heterogeneous agents with redundant capabilities at the design level are not yet fully established.

Research Question 2.1 (RQ2.1)

How to attain resilience in cyber-physical systems containing diverse agents with redundant capabilities by design?

RQ2.1 explores how engineers can incorporate design strategies that enable resilience in cyber-physical systems composed of heterogeneous agents requiring coordination. Specifically, to address **RQ2.1**, we investigate approaches for automated task allocation within cyber-physical systems, where a high-level declarative specification drives the assignment of tasks to agents with distinct but overlapping capabilities, ensuring coordinated behavior.

While coordination in multi-agent cyber-physical systems often assumes a shared goal, agents may operate with distinct, sometimes conflicting objectives. To attain resilience, these systems must maintain correct behavior even when interacting with other smart, autonomous agents displaying varied behaviors. Building assurance cases for such interactions requires simulating multi-agent environments in controlled testing setups [72, 73]. In complex scenarios, quality assurance teams must integrate dynamic agents, sometimes with adversarial behavior, alongside environmental elements like static (e.g., traffic signs) and dynamic (e.g., vehicles, pedestrians) objects [74, 75].

[1]Robustness is a form of resilience (c.f. Section 2.3.3)

Realistically modeling the behavior of these agents to the level needed for effective testing is challenging and often does not scale well. To address this, quality assurance teams typically rely on existing models of dynamic elements, either data-driven [76, 77] or algebraic models [78, 79]. However, it remains unclear how to specify these test scenarios effectively at scale for applications involving diverse agents and interactions.

Research Question 2.2 (RQ2.2)

How to attain resilience in cyber-physical systems by specifying test scenarios that reflect complex interactions between smart agents?

RQ2.2 investigates methods for specifying dynamic test scenarios to validate the cyber-physical systems. To address **RQ2.2**, we developed a model using reactive behavior trees that enable scalable and flexible composition of agent behaviors for high reusability.

1.2.3 Cyber-Physical Systems Prone to Uncertainty

Attaining resilience in cyber-physical systems necessitates implementing strategies that effectively counteract operational changes' disruptive impacts. These systems must be designed to function reliably despite unpredictable interactions with their environment and other agents [80, 81]. One effective approach to accommodating change in the design and testing phases is to model these changes as uncertainty. Indeed, uncertainty has significantly contributed to failures in cyber-physical systems [82, 83]. To build resilient cyber-physical systems, engineering teams require tools that enable them to manage uncertainty and facilitate adaptability. Various studies have explored methods to understand [84, 85] and mitigate [17, 86, 87] uncertainty. However, it remains an open question how these methods can effectively equip engineers to attain resilience in cyber-physical systems.

Research Question 3 (RQ3)

How to attain resilience in cyber-physical systems prone to uncertainty?

RQ3 examines the implications of uncertainty within cyber-physical systems and explores pathways to attain resilience in the face of uncertainty and the complexity it introduces. Our investigation of **RQ3** encompasses three dimensions. First, we assess how to manage uncertainty through design strategies, either by embedding environmental assumptions into in-code decision-making (**RQ3.1**) or by employing runtime reconfiguration to address uncertainty dynamically (**RQ3.2**). Furthermore, we consider how uncertainty can be mitigated during quality assurance by using automated tools that learn from faults to enhance system recovery capabilities (**RQ3.3**).

Taming uncertainty during design involves equipping cyber-physical systems with mechanisms to manage unpredictable scenarios as they unfold at runtime, even when designers lack precise knowledge of what may happen. However, waiting to address issues at runtime can lead to the awakening of silent bugs that later escalate into system failures, often due to misaligned assumptions made during the design phase. For instance, in robotic pick-and-place tasks, a developer might assume that an object remains in the exact location where the robot places it, overlooking possible environmental factors that could alter its position. Such bugs could be mitigated by embedding relevant environmental assumptions directly into the system's mission code, such as specifying the expected locations of static objects involved in tasks. Bridging the gap between the development environment and real-world conditions can enhance system resilience [88, 89]. Nonetheless, it remains unclear how to embed these environmental assumptions into the programming language developers use without adding unnecessary complexity or extra modeling layers.

Research Question 3.1 (RQ3.1)

How to attain resilience in cyber-physical systems prone to uncertainty through specification patterns that enable developers to embed environmental assumptions directly in code?

RQ3.1 explores language constructs that facilitate the integration of environmental assumptions directly into code during implementation. To answer **RQ3.1**, we propose four specification patterns: domain language mapping, early dynamic checking, symbolic tracing, and source provenance tracking. These patterns support incorporating environmental assumptions at the software level, helping developers catch errors early and enhance system resilience.

Self-adaptation is a fundamental strategy to manage runtime uncertainty, enabling systems to adjust to unforeseen changes in their environment and maintain correct service despite change [17, 90]. Two primary approaches guide self-adaptation in software design: architecture-based adaptation, which employs Monitor-Analysis-Planning-Execution (MAPE) components to handle adaptation autonomously [91, 92], and control-based adaptation, rooted in control theory to ensure stability and robustness through mathematically grounded mechanisms [93]. Control-based adaptation is particularly promising, as it offers robust, correct-by-design solutions that help sustain operations under disturbances. However, traditional control methods may struggle to adapt to unknown interactions that arise at runtime, underscoring the need for continued refinement using field data.

Research Question 3.2 (RQ3.2)

How to effectively use field data in control-based self-adaptation to attain resilience in cyber-physical systems prone to uncertainty?

RQ3.2 investigates the role of field data in adapting control-theoretic mechanisms to respond dynamically to runtime uncertainties. To address RQ3.2, we propose a layered architecture integrating control theory with search-based optimization algorithms, offering a flexible framework to inspire resilient cyber-physical systems' design.

Self-adaptive mechanisms can respond to runtime disruptions, but diagnosing the root causes of failures, particularly in systems tightly coupled to the physical environment, remains challenging. Automated change explanation has become vital, aiming to identify and understand failure-inducing changes to reduce recovery time. However, most existing methods depend on predefined failure scenarios [94] or operate solely from execution traces, lacking a direct link to system requirements [95, 96]. This gap suggests the need for adaptable, requirement-driven explanation methods that systematically trace runtime faults to root causes.

Research Question 3.3 (RQ3.3)

How to systematically derive explanations for detected faults to attain resilience in cyber-physical systems prone to uncertainty?

RQ3.3 investigates how information from system executions can be used to explain faults systematically. By treating explanation as a search problem, we aim to link property violations with execution traces, allowing developers to pinpoint root causes, thus enabling resilience through dynamic, adaptive fault understanding.

1.2.4 Resilience Exemplars

Research on resilience in computing systems has not kept pace with advancements in other fields, also due to a lack of representative exemplars, i.e., challenge problems and solutions that enable testing and refinement of resilience attainment strategies. With the focus on self-adaptation, the SEAMS community established a repository of exemplars[2], providing accessible artifacts for experimentation and evaluation. However, this repository lacks adequate support for control-theoretic self-adaptation and multi-robot mission scenarios, crucial for advancing resilience in cyber-physical systems characterized by complex interactions between computational and physical elements. Exemplifying change is essential for achieving resilience, offering developers practical examples of strategies applicable to real-world situations. Notable existing exemplars, such as DeltaIoT for the Internet of Things [97] and DARTSim for unmanned air vehicle simulations [98], provide a foundation for assessing resilience tactics. Addressing gaps with new exemplars helps to enhance the repository and accelerate the development of resilient systems.

Research Question 4 (RQ4)

How to develop exemplars to advance research on resilience attainment in cyber-physical systems?

RQ4 aims to identify and create exemplars that empower the research community with concrete resources, thereby accelerating research in resilience for cyber-physical systems. To contribute to this objective, we have developed two exemplars for the SEAMS repository, designed to fill current gaps and enable meaningful experimentation in resilience research.

[2] https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/

1.3 Methodology

This thesis aims to devise *strategies* to equip practitioners with methods, tools, and techniques to attain resilience in cyber-physical systems. With this objective in mind, we have followed the design science methodology applied to such research intentions, in which strategies are the design artifacts. Fundamentally, design science is a problem-solving paradigm [99], seeking to deliver innovative and useful ideas, practices, technical capabilities, and products. The paradigm looks at research as a means to delivering relevant artifacts justified by knowledge applied with rigor [100–102]. Design science leverages solution-seeking research [103] in a suitable research framework to tackle our research goal. Ultimately, the strategies are the research artifacts resulting from a systematic design effort aiming at relevance, rigor, and novelty.

1.3.1 Devise Strategies with Design Science

The distinct nature of our research questions (RQ1–RQ4) requires different types of studies within the design science approach. We use the design science problem-solving framework [104] to identify the different studies followed in this thesis. According to the framework, there are four individual knowledge-constructing activities, namely *descriptive study*, *solution validation study*, *solution design study*, and *problem solution pair study*. Figure 1.1 describes that we used descriptive study to answer to **RQ1**, problem-solving studies to answer to **RQ2** and **RQ3**, and solution design to answer **RQ4**. Next, we elaborate on the nature of each research question and provide the reasoning behind the choice of the research method to tackle them from the point-of-view of the knowledge-constructing activities from design science. Especially for **RQ2** and **RQ3**, we provide only one example of how the Problem-Solution pair constructs map to the study **RQ3.2**, although a similar reasoning applies to all.

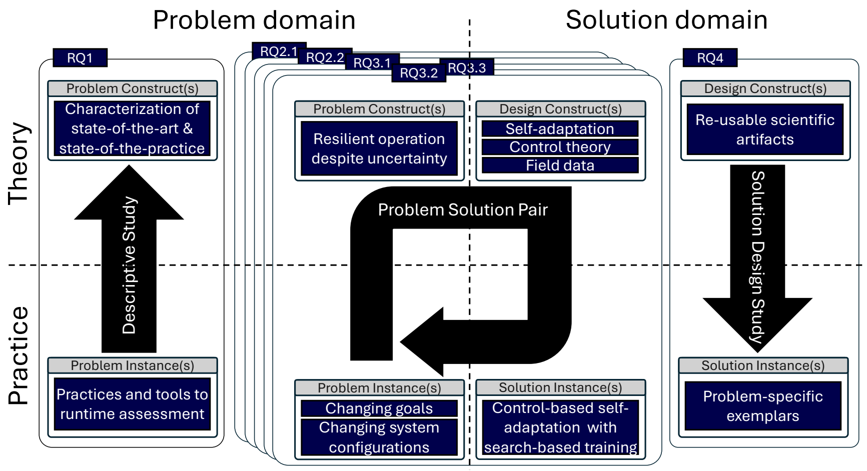


Figure 1.1: Methodology description according to the problem-solution framework for design science [104]

First, **RQ1** investigates the scientific literature and software repositories to attain resilience in cyber-physical systems, focusing on runtime assessment, assuming that practices and tools may be extracted from public knowledge. Therefore, to answer the question, the researchers must find and extract instances of recurring artifacts from known published knowledge and reason within a theoretical framework that best fits the study's goal, i.e., runtime assessment to attain resilience in cyber-physical systems, in the case of **RQ1**. Consequently, the design science study that best answers **RQ1** is a descriptive study, as depicted in Figure 1.1. In short, descriptive studies develop an understanding of a currently poorly understood phenomenon, e.g., attaining resilience in cyber-physical systems. Such studies may reveal problems that need to be addressed or practices or tools that could benefit practitioners and researchers. Through synthesis, problems, practices, and tools elicited in descriptive studies may turn into strategies (i.e., actionable items) to steer research and practice toward the research goal.

Then, **RQ2** and **RQ3** investigate solutions to open problems. **RQ2** asks for means to attain resilience in a particular type of cyber-physical systems that contain heterogeneous and smart agents. **RQ3** asks for ways to attain resilience in uncertain cyber-physical systems. Both problems are open and not well defined, given the particularities of cyber-physical systems. For instance, depicted in Figure 1.1, **RQ3.1** investigates self-adaptation to attaining resilience in cyber-physical systems in the face of uncertainty. The question departs from uncertainties emerging from changing goals and system configurations during runtime. It requires runtime mechanisms that carry the capacity of self-adaptation, with mathematical rigor, and learning with data from the field. Ultimately, the solution instance reflects the design constructs implemented within an architecture on the language that can be exercised to gather empirical evidence of its effectiveness. In conclusion, a problem-solution pair study is the appropriate study to address **RQ2** and **RQ3**. Problem solution pair studies rely on identifying a problem instance, e.g., changing goals and system configurations, and investigating it to devise a generalized problem formulation (e.g., resilient operation despite uncertainty) that should match a proposed solution composed of the design constructs (e.g., self-adaptation, control theory), and materialized as a solution instance, (e.g., code integrated to the experiment apparatus). The researchers rigorously validate the solution artifact by following empirical methods, typically experimentation.

Finally, **RQ4** asks for new exemplars for equipping researchers with concrete support to developing new scientific studies, methods, and tools, resulting in further and faster advancement of scientific knowledge for attaining resilience in cyber-physical systems. This question assumes that providing novel exemplars to the scientific community constitutes a solution to an overarching and well-understood problem and that the lack of (re-)usable exemplars hinders the evolution of and advancement of research in the field. Therefore, as depicted in Figure 1.1, **RQ4** requires a solution design study. Solution design studies present new instances of a general solution to a well-established problem that is the best fit to address **RQ4**. In this case, the design constructs abide by re-usable scientific artifacts to replicate experiments, validate research proposals, or compare to scientific artifacts, which can be concretely implemented as specific exemplar artifacts to recurring problems.

1.3.2 Data Collection Techniques

Various data collection techniques fulfill the studies in this thesis: focused group, user study, systematic literature review, repository mining, case study, and experimentation. Distinct study types ask for different data collection methods. This section discusses the methods used in this thesis to navigate the problem-solution framework from Fig. 1.1. A summary of data collection techniques mapped to the studies is presented in Table 1.1.

Data Collection Technique	Descriptive Study (RQ1)		Problem Solution Pair (RQ2, RQ3)		Solution Design (RQ4)	
	Design	Validation	Design	Validation	Design	Validation
Focused Group			x			
User Study		x			x	
Systematic Literature Review	x					
Repository Mining	x					
Case Study				x		
Experimentation				x		x

Table 1.1: Summary and mapping of data collection techniques to studies to answer **RQ1–RQ4**

Aiming at realizing a descriptive study that sheds light on the current practices, methods, and tools to enable runtime assessment as a means to attain resilience in cyber-physical systems, answering to **RQ1**, the researchers should combine a systematic literature review [105] and artifact analysis through repository mining. The systematic assessment of the scientific literature defines the theoretical grounding that supports the characterization of such current practices and tools. At the same time, the collection of repositories and code analysis allows for the provision of concrete examples of how such theoretical constructs are used in reality. Moreover, to collect confidence that the uncovered practices are useful and applicable for developers, practitioners should validate the resulting characterization of the practice. Qualitative studies, such as user studies, are recommended to achieve such results.

Problem solution pair studies to answer **RQ2** and **RQ3** follow a protocol in which the problem instance and problem constructs result from focused group discussions mixed with lightweight literature reviews among the participant researchers (i.e., paper authors) with varied experience in the fields of cyber-physical systems and resilience attainment. Design constructs are synthesized to answer the raised questions and implemented in a solution instance. To collect evidence that the solution instance offers an answer to the research questions, the main technique is experimentation [106] for attesting the effectiveness and efficiency of the solution instance. At the same time, case study [107] may also be used to attest qualitative properties of the solution instance that may not be accurately measured, such as readability and maintainability.

The solution design study supporting the scientific community with reusable scientific artifacts, answering **RQ4**, departs from defined design constructs that can be gathered from the community and made available through surveys or synthesized and validated through experimentation.

1.4 Brief Outline of Contributions

In the light of design science, our study devises seven strategies (**STG1-STG7**) that respond to the research questions (**RQ1-RQ4**). Fig. 1.2 displays the strategies within a model that assembles the contributions of this thesis in a holistic perspective of the work, mapping research questions and publications. The strategies are organized into design and quality assurance activities. The design and development group engulfs STG2, STG4, and STG5; the quality assurance group engulfs STG3 and STG6; STG1 and STG7 pertain to both.

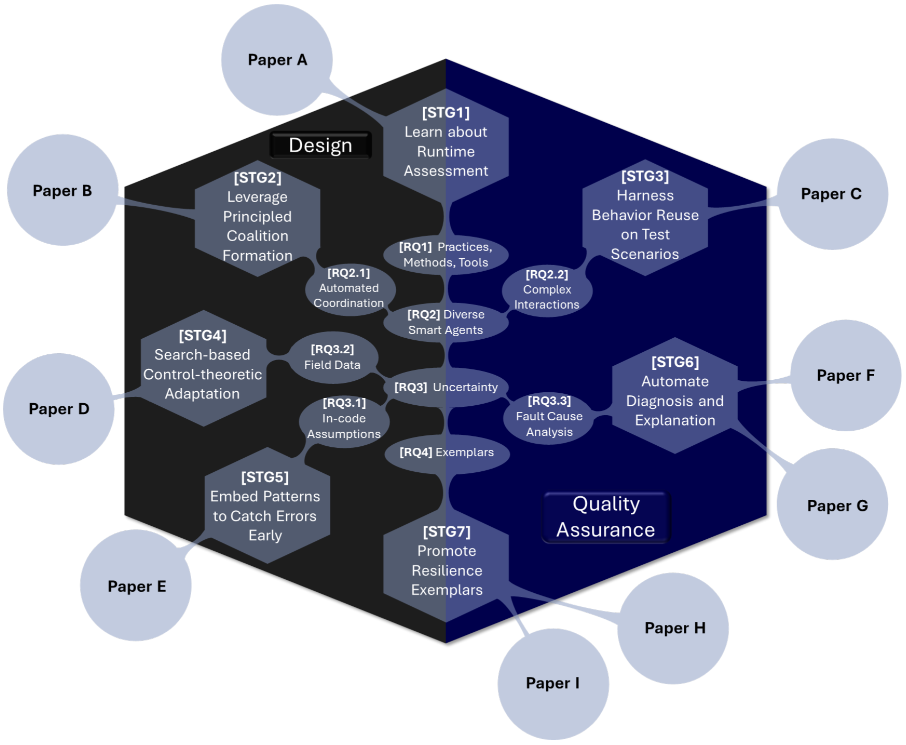

Paper A
Design
[STG1] Learn about Runtime Assessment
Paper B
[STG2] Leverage Principled Coalition Formation
[STG3] Harness Behavior Reuse on Test Scenarios
Paper C
[RQ1] Practices, Methods, Tools
[RQ2.1] Automated Coordination
[RQ2.2] Complex Interactions
[RQ2] Diverse Smart Agents
[STG4] Search-based Control-theoretic Adaptation
[RQ3.2] Field Data
[STG6] Automate Diagnosis and Explanation
Paper F
Paper D
[RQ3] Uncertainty
[RQ3.1] In-code Assumptions
[RQ3.3] Fault Cause Analysis
[RQ4] Exemplars
Paper G
[STG5] Embed Patterns to Catch Errors Early
Quality Assurance
[STG7] Promote Resilience Exemplars
Paper E
Paper H
Paper I

Figure 1.2: Overview of the strategies (**STG**) and how they map publications (**Paper**) and research questions (**RQ**)

Strategy 1 (STG1) – Learn Best Practices to Runtime Assessment

Resilient cyber-physical systems require decision-making during runtime. To effectively adhere to changes, the system must benefit from awareness of the system itself and its environment. Runtime assessment techniques enable introspection and self-management of software-intensive systems. STG1 aims to equip developers and quality assurance teams with practical knowledge of runtime assessment of cyber-physical systems. Consequently, STG1 promotes a set of 20 guidelines that equip practitioners with methods, tools, and the current (and best) practices for employing runtime verification and field-based testing to attain resilience in cyber-physical systems. STG1 is an outcome of the descriptive study presented in Paper A as an answer to RQ1 and supports attaining resilience from both design and quality assurance angles.

Strategy 2 (STG2) – Leverage Integrability and Modifiability to Multi-Agent Coalition Formation

Attaining resilience in cyber-physical systems with heterogeneous agents requires coordinating agents with diverse and, at times, overlapping capabilities. STG2 offers principles to guide developers in designing resilient cyber-physical systems with multiple heterogeneous and intelligent agents, aiming to prevent failures due to sub-optimal task allocation. STG2 centers on two design principles: integrability and modifiability. Integrability enables agents to combine capabilities effectively, supporting coalition formation and reducing task allocation errors. Modifiability ensures agents can adapt coalition roles as conditions change, enhancing system responsiveness. STG2 includes an example of multi-robot architecture that applies these principles, demonstrating their role in improving resilience. Developed from a problem-solution study in Paper B, STG2 answers RQ2.1 by offering developers guidance for building robust cyber-physical systems.

Strategy 3 (STG3) – Harness Behavior Reuse for Scalable Specification of Smart Agents with Distinct Goals

Attaining resilience in cyber-physical systems requires collecting evidence that multiple agents with distinct, even conflicting, objectives can coexist without compromising the systems' integrity. Such evidence is important, especially for complex systems operating in dynamic environments. However, collecting assurance cases for dynamic interactions is impractical. Quality assurance teams must repeatedly re-implement intricate agent behaviors in test scenarios with shared environments, which is inefficient and lacks scalability. STG3 addresses this by equipping quality assurance teams with a reusable behavior library, reducing time and effort in defining new test scenarios involving multiple smart agents. STG3, developed in Paper C, answers RQ2.2 by helping quality assurance teams efficiently collect assurances about the system's correct behavior in multi-agent interactions.

Strategy 4 (STG4) – Tame Field Data for Control-based Adaptation

A core strategy to attain resilience in cyber-physical systems under uncertainty is integrating field data into control-based self-adaptation mechanisms. Self-adaptation enables systems to degrade gracefully and recover from faults amid unforeseen runtime changes. However, designing adaptive systems capable of handling unpredictable conditions is challenging and requires a structured approach to tame field data. Control theory offers a solid framework for building feedback loops that respond dynamically to variability and disturbance. Search-based optimization offers algorithms to navigate large configuration spaces efficiently. STG4 guides practitioners using control-theoretic principles aligned with search-based optimization to achieve resilient operation by incorporating runtime data to enhance adaptability and fault tolerance. This strategy, derived from a problem-solution study in Paper D, directly addresses RQ3.2 by proposing a design-focused approach for embedding resilience into cyber-physical systems through adaptive, data-driven control mechanisms.

Strategy 5 (STG5) – Embed Specification Patterns to Catch Errors Early

Building resilience in uncertain environments requires cyber-physical systems to be prepared for interactions influenced by external agents and the system itself. Such interactions bring uncertainty, making incorporating environmental assumptions in the codebase essential. Failures often arise from assumptions overlooked or not formalized during design. STG5 proposes specification patterns to encode these assumptions, like relationships, attributes, and constraints, directly in code. These patterns enable dynamic checking, symbolic tracing, and source provenance tracking, allowing developers to catch errors caused by assumption violations early in development. Based on a problem-solution study in Paper E, STG5 provides a structured method for embedding assumptions as patterns, enhancing resilience in cyber-physical systems under uncertainty, and addressing RQ3.1 from a design perspective.

Strategy 6 (STG6) – Automate Diagnostics and Explanations for Swift Recovery

Cyber-physical systems must quickly recover from failures to enhance resilience, which requires a clear understanding of their root causes. In such systems, tightly coupled with their physical environment, replicating failure scenarios for root cause analysis can be impractical, costly, or impossible. STG6 provides tools for quality assurance teams to automatically derive diagnostics and explanations for runtime failures, enabling faster recovery. Using traces and property specifications, STG6 supports automated root cause analysis, paving the way for resilience through efficient failure explanation and swift system recovery. This strategy is based on insights from problem-solution studies in Papers F and G, directly addressing RQ3.3.

Strategy 7 (STG7) – Promote Exemplars to Advance Resilience Research

The scientific community must prioritize the development of methods, tools, and evidence that support developers and quality assurance teams in attaining resilience in cyber-physical systems. STG7 encourages researchers to create and share exemplars that drive motivation and innovation in resilience research for cyber-physical systems. Two solution design studies detailed in Papers H and I inform this strategy, effectively addressing RQ4. STG7 enhances resilience from both design and quality assurance perspectives by focusing on exemplification.

1.5 Thesis Organization

Chapter 1 discussed the problem and research questions tackled by the strategies to attain resilience in cyber-physical systems. Next, Chapter 2 defines the terminology used throughout the thesis and outlines related works. Then, in Chapter 3, we summarize all the work that amounts to the thesis regarding published papers and discuss their results. Finally, on Chapters 4–12, we append the publications containing the works' details.

Chapter 2

Terms and Related Work

Resilience, as a theme, is as antique as recurrent in the universal literature. Over time, resilience has been adopted and refined across various fields, yet it has only recently gained attention in computer science and engineering. This chapter defines fundamental terms used throughout the thesis. It highlights current and related works by comparing the thesis to the most recent contributions in the scientific literature on resilience in cyber-physical systems.

2.1 Terminology

What is resilience? Resilience, as a term, is broadly used, both in the universal and scientific literature. This thesis, sheds light on resilience as a property of systems, thus needs to be precisely defined. While providing a map to resilience in cyber-physical systems, this section elaborates on the term using a conceptual framework fuelled by the Boeing 737-MAX 8, Tesla Autopilot and Waymo case studies.

2.1.1 What is Resilience?

Resilience, as a term, comes from the Latin word *resilire*, meaning "to leap back", "rebound", "recoil", or "retreat." The term was first documented in the Material Sciences during the beginning of the 19th century; elucidated from a historical perspective in the *History of strength of materials* book from Timoshenko S. [21]. The book reviews the evolution of strength of materials and refers to Thomas Young's studies that posit resilience as a property of a beam (or bar) and that resilience is proportional to the beam's length. Over time, its meaning has been adapted in various contexts, from material sciences (early 1800s) to ecology (1970s) and sociology and psychology (1990s), before becoming prominent in computing (early 21st century). As of today, resilience, a concept broadly studied across fields like environmental sciences [108], ecology [109], psychiatry [110], economics [111], lacks a universally accepted definition due to its diverse applications [112].

This thesis approaches resilience in cyber-physical systems from a broader software engineering perspective and, thus, draws upon J.C. Laprie's foundational definition.

Definition – Resilience

Resilience is *"the persistence of service delivery that can justifiably be trusted, when facing changes."*, Laprie J. C. [19]

Following Laprie's definition, resilience has garnered significant attention in computing, encompassing various aspects such as system-level fault tolerance, robustness, and software adaptability. Resilience, in computing, often refers to the system's ability to maintain required functionality in the face of adversity; Whereas change, materialized as faults, errors, or failures, plays a key role in software system's success. As a systems' property, resilience may manifest in architectural design patterns, for instance, promoting loose coupling and modularity (e.g., in micro-services [113]), as enablers to runtime modification and flexibility. Resilience may also manifest in verification and validation techniques that, for example, proactively induce changes in the system through fault injection (e.g., chaos engineering [114]).

As a foundational study to enable the discourse and systematic attainment of resilience in computing systems, we present a recent work that conceptualizes and describes the basic terms on which this thesis relies. Formally, Andersson J. et al. [41] defined resilience within the conceptual framework depicted in Fig. 2.1. Other frameworks also formally define resilience [115,116]. We follow Andersson J.'s definition since it allows us to focus on robustness, change, and uncertainty, further discussed in the thesis.

The conceptual framework, depicted in Fig. 2.1, separates the space of possible system states into three: acceptable states, survival space, and dead space. *System states* (●, in Fig. 2.1) correspond to a collection of attributes required for describing the system and its behavior. Each state may be classified according to distinct acceptance criteria. The acceptance criterion (θ) defines a set of constraints and relationships on the system state that allows for the

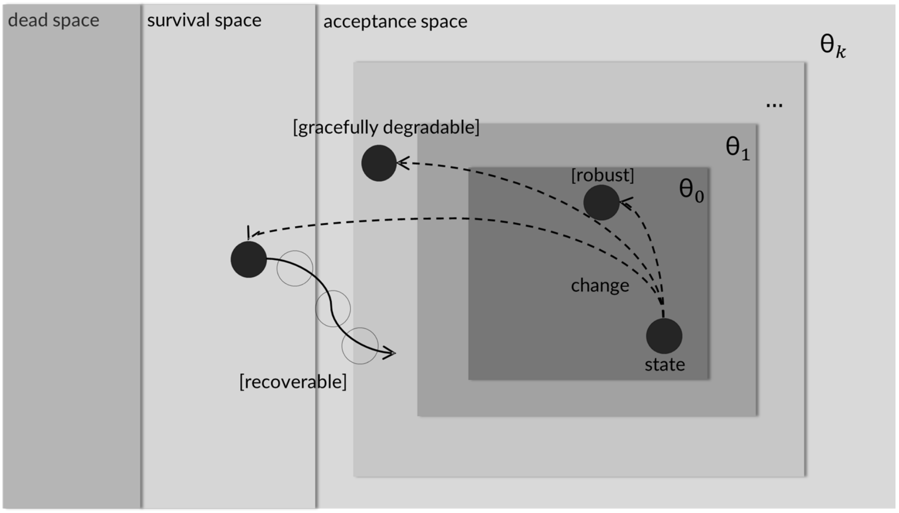


Figure 2.1: Visual representation of resilience for computing systems. Adapted from the version defined by Andersson J. et al. [41].

identification of the subset of the system state space consisting of all those states where the service delivered by the system can be considered correct and acceptable according to θ. There can be several acceptance levels within the determined criteria (θ_k), in which k corresponds to the identifier of a series of progressively less stringent acceptance criteria. The survival space encompasses all those states where the service delivered by the system is not acceptable but for which a sequence of internally or externally initiated corrective actions exists, which rings the system back to a state within the acceptable space. The dead space, in opposition, encompasses all states where the delivered service is not acceptable and precludes the possibility of returning to an acceptable state.

System changes, depicted as dashed lines in Fig. 2.1, are categorized along two dimensions: readiness and persistence. Readiness concerns the system's readiness to deal with the change, and it differentiates between expected and unexpected changes, with expected changes further divided into handled, in which the system has resources to manage, and unhandled, which are foreseen but not addressed. On the other side, unexpected changes, or "surprises", may lead to any subspace in the state space, acceptable, survival, or dead spaces. Persistence distinguishes between transient, either temporary or reversible effects, and permanent changes, with long-lasting impacts on system function. Together, these dimensions allow for a precise assessment of a system's capacity to manage and recover from disruptions.

Next, we study three cases of cyber-physical systems in the contexts of the aviation and automotive industries from the perspective of resilience within the conceptual framework.

2.1.2 Boeing 737-MAX: A Case of Brittle Software

Non-resilient systems are brittle; they break when bent. This section analyses the Boeing 737-MAX 8 case, which caused two accidents in late 2018[1] and early 2019[2]. Remark: There are several hypotheses on what led to the crashed airplanes, including design flaws, lack of transparency with the released software, and lack of monitoring from the regulatory corpus [117] – failures of this dimension are often due to a chain of events. In this section, we focus on the software system, namely MCAS, from the point of view of software and resilience as a property of the cyber-physical system [118].

In early 2018, Boeing released the 737-MAX 8, which, compared with previous 737 models, had a larger engine to enhance fuel efficiency. Such enhancement required repositioning the engines higher and further forward on the wings compared to earlier designs. The new engine placement caused in-flight instability in particular flight conditions. While the engine housings typically do not generate lift during normal flight, they create more lift in situations like takeoffs or hard turns. Under such conditions, the extra lift combined with the plane's momentum could cause it to exceed the desired pitch, potentially leading to a stall[3].

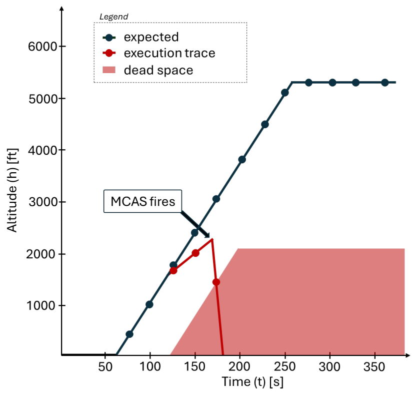
[1] https://edition.cnn.com/2018/10/28/asia/lion-air-plane-crash-intl

[2] https://edition.cnn.com/2019/03/10/africa/ethiopia-airline-crash-nairobi-intl

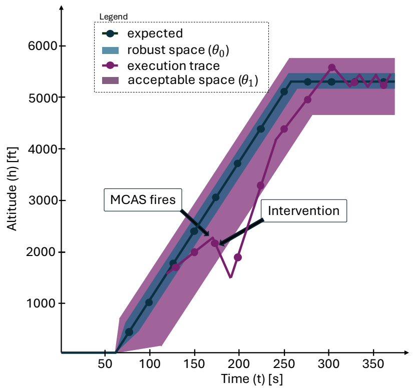
[3] In aviation, a stall occurs when an aircraft's wings can no longer generate enough lift to sustain flight

Boeing relied on software to release fuel-efficient aircraft competitively in the market. The cyber-physical systems producer released a software solution that automatically prevents the aircraft from stalling, the Maneuvering Characteristics Augmentation System (MCAS). The software is part of the flight management system and is wired to the flight controllers and aircraft sensors. In short, the MCAS monitors the airspeed, altitude, and angle of attack, and, in case of imminent stall, the MCAS acts by adjusting the rear stabilizer, lowering the nose, and pushing the yoke downwards; once the aircraft has no risk of stalling, the control is handed back to the pilot. The pilot can override the MCAS system if desired. However, the system is reactivated if the combination of airspeed, altitude, and angle of attack still requires so. Nonetheless, the software system was unprepared for erroneous sensor measurement data.

Curran N. T., et al. [119] analyze the MCAS system and simulate how the original MCAS design can fail under realistic conditions. We rely on their study to understand the software design brittleness in the lenses of the conceptual framework presented in Fig. 2.1. The case study analyses maneuvers from commercial flights: takeoff, climb, cruise, descent, and approach. We focus on the takeoff since it was the phase in which the Lion Air Flight JT610 and the Ethiopian Airlines Flight 302 failed, illustrated in Fig. 2.2. Assuming the states $\theta_k(t)$ represent the airplane's altitude $h(t)$; $\theta_0(t)$ (robust space) determines that the aircraft follows the required altitude during takeoff, and $\theta_1(t)$ represents a degraded state in which the error does not escalate and permits the airplane to reach cruise, and the survival space is characterized by a recoverable state, in which, the pilot reaction time in which the pilot can still engage in a recovery maneuver. Otherwise, the system is said to be in a dead space.



(a) Crash scenario (non-resilient)



(b) Scenario with intervention (resilient)

Figure 2.2: Graphical representation of the takeoff maneuver from Boeing 737-MAX 8. This figure is inspired in data and graphs from [119].

To simulate the scenarios under which the 737 MAX 8 faced edge conditions, Curran N. T. et al. implemented an injection mechanism that emulates changes (i.e., erroneous inputs) from the sensors or the pilot. The changes were permanent, injected around time $t = 100s$, and caused MCAS to fire, pushing the aircraft downwards to stop stalling. In the case of Fig. 2.2a, in which no resilience mechanisms exist, the aircraft crashes on the ground. On the other side, in Fig. 2.2b, we illustrate a case in which there is an intervention that maintains the system delivering its service in a degraded state, yet acceptable.

Manually, the *Intervention*, in Fig. 2.2b, asks the pilot to disengage from the MCAS functionality, which was unsuccessfully attempted in both flights, resulting in a crash. In fact, a report from the Lion JT610 crash explained that the pilot tried to correct the aircraft by pointing the nose higher, but the system kept pushing it down 21 times before the crash occurred – such a manual solution is error-prone. *The MCAS deployed in the Boeing 737-MAX was not resilient.*

How could the MCAS detect, identify, and thus understand whether the sensors collected erroneous information? How can the new update of the MCAS be developed to acknowledge change and react? Given that the MCAS is inevitably subject to uncertain operational conditions, how to prepare the system for operating in a degraded but acceptable state? How to collect explanations for faults that enable system recovery before they turn into failures? These are open questions that, in a broader context, motivate the writing of this thesis.

2.1.3 Tesla Autopilot and Waymo ADS: Resilience in the Automotive Industry

Road traffic kills 1.19 million people per year according to WHO[4]. It is as if a Boeing 737-MAX 8 would fall every hour of a year in the death toll. The automotive industry faces a grand safety challenge and focusing on resilience when building automotive systems has a great potential to save lives. This section analyses both Tesla Autopilot and the Waymo Autonomous Driving System (ADS) from the point of view of resilience attainment and its challenges. Moreover, we analyze the cases separately since Tesla's ADS is autonomy level 2 and Waymo's ADS is level 4, meaning that they are developed to different capacities even though both share similar goals. [120]

2.1.3.1 Is Tesla Autopilot Ready for the Real World?

Resilient systems are firm; they might deform when bent but do not break. Tesla's Autopilot, the company's advanced driver-assistance system (ADS), plays a critical role in this context. In this section, we examine the resilience of the Tesla's ADS, particularly in light of Joshua Brown's fatal crash. This case highlights key challenges in the interaction between autonomous systems and their environments, revealing potential vulnerabilities in the design of cyber-physical systems.

In 2016, Joshua Brown's Tesla Model S, operating on Autopilot, was involved in a fatal crash – the first of 47 verified fatalities associated with Tesla's Autopilot.[5][6] Autopilot, unlike the MCAS (Sect. 2.1.2), employed a combination of forward-facing cameras, radar, and ultrasonic sensors to assist with driving tasks. However, it failed to detect a white tractor-trailer crossing the road under a bright sky. The forward-facing camera misinterpreted the

[4]https://www.who.int/publications/i/item/9789240086517

[5]https://money.cnn.com/2016/07/01/technology/tesla-driver-death-autopilot/

[6]https://www.tesladeaths.com (accessed on 08-10-2024)

trailer's reflective surface as part of the sky, and the radar, tuned to ignore overhead objects, failed to engage emergency braking, misclassifying the trailer as a sign. This radar algorithm, designed to reduce false positives, ultimately contributed to the system's inability to recognize the critical obstacle in its path.

This crash underscores significant limitations in the resilience of Tesla's Autopilot system. Although Autopilot was designed for highway driving, with lane-keeping and adaptive cruise control, it was not equipped to handle complex scenarios such as cross-traffic, as seen in Brown's accident. The National Transportation Safety Board (NTSB) report revealed that the system relied heavily on driver attention for safety-critical interventions [121]. In the moments leading up to the crash, the driver did not respond to system alerts or take control of the vehicle, exposing a critical flaw in the human-machine interaction model. The NTSB attributed part of the crash to the driver's overreliance on automation – an indication of brittleness.

Resilient cyber-physical systems must account for complex interactions with multiple agents in their environment. These interactions may include unexpected events such as a tractor-trailer with a reflective surface crossing the road, risky maneuvers by other vehicles, or unpredictable pedestrian behavior. Each agent, whether human-driven or automated, operates with distinct goals – some global, like minimizing traffic time, and others individual, like reaching a destination faster. These agents may collaborate or act in opposition, and they are often designed by different manufacturers, each employing distinct capabilities and decision-making frameworks. Despite adherence to manufacturing standards, variability in these systems' abilities creates further challenges for ensuring resilience in real-world scenarios.

2.1.3.2 Dynamic Safety for Resilience in Waymo ADS

Resilient systems acknowledge change and uncertainty as key concerns to successful execution. According to the National Highway Traffic Safety Administration (NHTSA), there is an average of one crash every 670,000 miles driven in the United States. In 2024, Waymo reported 22.2 million driverless miles across Phoenix, San Francisco, and Los Angeles, with one crash occurring every 115,000 miles driven. This marks significant progress, including 84% fewer airbag deployment crashes, 73% fewer injury-causing crashes, and 48% fewer police-reported crashes compared to human-driven vehicles. Importantly, no fatalities have been reported in association with Waymo's ADS operations.[7] In this section, we examine the Waymo's ADS from the perspective of resilience attainment.

The Waymo ADS adopts team a dynamic safety approach, treating safety as an emergent property of software design, an observable metric for deployment readiness, and a continuously evolving measure of system confidence [122]. In-use monitoring supports field testing throughout the dynamic safety phases, ensuring performance alignment with targets set during the readiness review phase. Continuous monitoring further helps with detecting changes in existing or emerging threats. As the deployment scales and data availability increase, the statistical confidence in the system's readiness improves. Waymo's team leveraged field data to reconstruct fatal crashes within its operational design

[7]https://waymo.com/blog/2024/09/safety-data-hub (accessed on 09-10-2024)

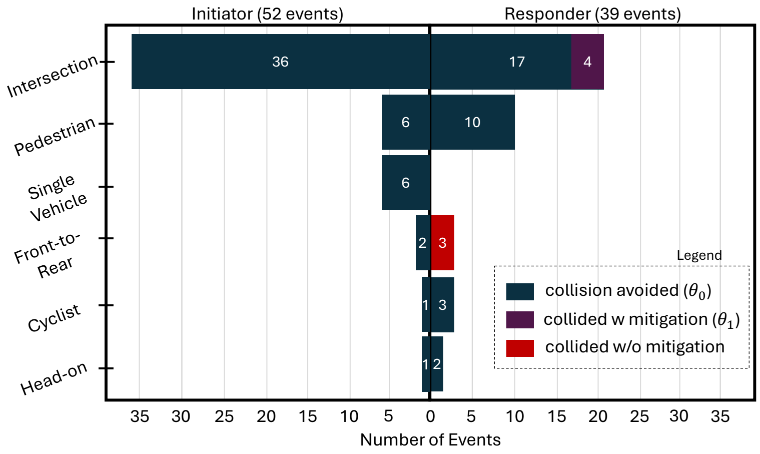


Figure 2.3: Crash outcomes of tests on the Waymo's autonomous driving system. This figure is inspired by data and graphs from [123].

domain to further quantify the system's effectiveness, as described by Scanlon et al. [123]. The team collected data from fatal collisions in Chandler, Arizona, over a 10-year span (2008–2017), which were then reconstructed by a specialized company. These reconstructions generated pre-crash collision sequences, allowing Waymo to conduct counterfactual simulations where the Waymo team tested the ADS against real-world collision scenarios.

In contrast to the analysis of the MCAS system (Sect. 2.1.2), this study assesses resilience from a scenario-based perspective. Here, $\theta_k(t)$ represents the vehicle's ability to avoid crashes or mitigate crash severity. Scenarios where the Waymo ADS successfully avoided a collision demonstrate system robustness ($\theta_0(t)$). In cases where the system could not prevent a crash but reduced its severity, the vehicle exhibited degraded yet acceptable behavior ($\theta_1(t)$). If the vehicle did not mitigate or prevent a crash, it is considered to have reached an irrecoverable state. Fig. 2.3 illustrates 72 experiments conducted by Scanlon et al., which recorded whether the vehicle avoided a crash, collided with mitigation, or collided without mitigation.

Scanlon et al. [123] simulated 72 crashes and 91 vehicle actors, 52 initiators, and 39 responders. Fig. 2.3 presents the crash outcomes based on the vehicle's role. The system successfully navigated the scenario without a collision in all cases where the Waymo ADS acted as the initiator (52 events). As a responder (39 events), the Waymo ADS avoided 82% of collisions (32 events) and mitigated an additional 10% (4 events). The remaining 8% (3 events) resulted in unchanged collisions, all in front-to-rear impacts. Notably, none of the simulations showed severe collisions.

Waymo strives to achieve resilience in its automotive system. However, questions remain about the system's ability to coordinate groups of vehicles with different capabilities, such as varying perception and speed profiles. How can Waymo's developers and quality assurance teams systematically assess and ensure that the ADS will function reliably under uncertain conditions? Additionally, how can they effectively analyze and explain the failures that led to collisions without mitigation?

2.2 Means to Attain Resilience

Attaining resilience in cyber-physical systems requires adopting strategies that counteract the disruptive effects of operational changes. Across various fields, e.g., emergent distributed systems, like the Internet of Things [124] and cloud computing [125], researchers widely agree on three core goals to support resilience: reducing the probability of failure, minimizing the impact of failures, and shortening recovery time [41]. While resilience research has also expanded to human factors, addressing developers' resilience for creating resilient software [126], studies specific to cyber-physical systems remain limited and mostly focused on security [127]. Resilience, however, transcends security, requiring a broader, cross-cutting approach to effectively handle change.

Definition – Resilience Attainment

Strategies to attain resilience handle runtime changes to mitigate failure.

Inspired by existing frameworks [41], we posit that resilience attainment depends fundamentally on addressing change. Techniques supporting resilience must manage change through strategies that encompass understanding the change, representing the change, and preparing the system for change. Understanding the change concerns detection, identification, and explanation of failure-inducing changes; Representing the change concerns modeling, specification, and exemplification of changes; and Preparing the system for change concerns, designing and testing systems to accommodate change or enable rapid recovery from failures. Fig. 2.4 maps the goals to support resilience attainment and strategies to enact each goal. In the following, we discuss each strategy, showcasing noteworthy studies on resilience attainment and gaps in the literature.

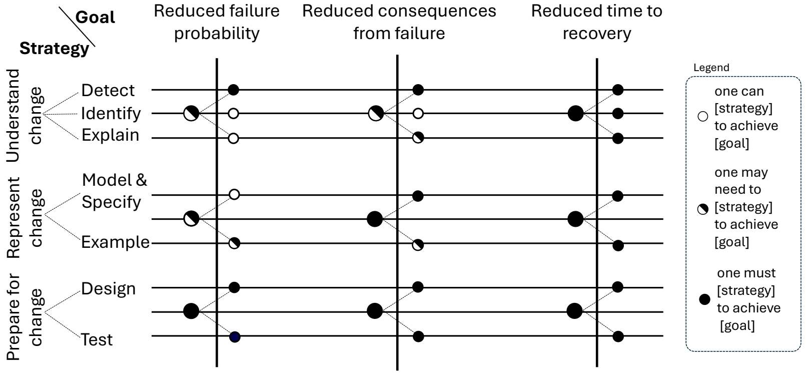

Goal
Strategy
Reduced failure probability
Reduced consequences from failure
Reduced time to recovery
Understand change
Detect
Identify
Explain
Represent change
Model & Specify
Example
Prepare for change
Design
Test
Legend
one can [strategy] to achieve [goal]
one may need to [strategy] to achieve [goal]
one must [strategy] to achieve [goal]

Figure 2.4: Goals and Strategies to attain resilience.

2.2.1 Understanding Change

Understanding change is often useful to attain resilience in cyber-physical systems, as it enables a system to observe the change and adapt in response to internal and external shifts that may threaten its functionality. Understanding change entails detecting, identifying, and explaining changes to manage failure risks, promote recovery, and sustain operational goals despite uncertainties.

Detecting change is essential to resilience attainment in cyber-physical systems. Change detection may be useful for techniques that aim at reducing failure probability and the consequences of failures, yet are fundamental for techniques that promote system recovery. Change detection is concerned with finding patterns in how the system behavior evolves over time. Two key techniques for change detection are anomaly detection and runtime monitoring. Anomaly detection, similarly to outlier detection [128], focuses on extracting such patterns from data (i.e., system logs) that do not conform to expected behavior [129–131]. Runtime monitoring continuously checks the system for violations of predefined properties on system traces and may happen along the system's execution [132–134]. Literature on change detection to attain resilience in cyber-physical systems discusses attack detection [135, 136], deep learning [137, 138] and control theory [139, 140], and timidly discuss runtime monitoring [61]. Thus, it sheds light on the lack of studies on aspects other than security anomalies, like safety checks, and the need for tooling and a better understanding of systematically detecting changes in cyber-physical systems during runtime.

Identifying change may help understand the nature and impact of the detected change. Changes are either evolutionary or operational. They may be expected or unexpected, once expected they can be handled or unhandled. Changes might be transient or permanent. Changes may also be intended or unintended. The changes may be done or triggered by software developers, the system itself, or externalities. Each change has a different potential effect in the system [141]. In the context of resilience attainment, we discuss failure-inducing runtime changes. Reactive self-adaptation techniques, for instance, may require the identification of the changes in order to restore it and maintain correct service delivery, e.g., FORMS [142] models change with environment processes in the context of autonomous vehicles. Although the literature on software changes is extensive, especially in software evolution [143, 144], there lacks approaches targeting the specific nature of cyber-physical systems, which are closely intertwined with the physical world. As a solution, change has been modeled as a probability that the system will not behave as expected, a.k.a. *uncertainty*. A recent thrust has been towards understanding and identifying uncertainty as a key activity to establish system-level resilience. Perez-Palacin D. and Mirandola R. propose MUSE, an approach to identify the presence of uncertainty in software systems [145]. Maribel A. et al. then discuss uncertainty awareness and classification in the context of automated driving [146], and Calinescu R. et al. promote an understanding of (classes of) uncertainty from a community perspective [147].

Explaining change concerns collecting facts about a failure-inducing runtime change, which involves determining its root cause. A change explanation is essential to techniques that reduce the time needed to recover since the cause of failure must be understood. Fundamentally, root cause analysis is derived from the works of Pearl J., from statistics, in which *"its aim is to infer not only beliefs or probabilities under static conditions, but also the dynamics of beliefs under changing conditions, for example, changes induced by treatments or external interventions"* [148]. However, automatically deriving explanations for changes has recently gathered thrust in computer science due to the trend in artificial intelligence algorithms [149, 150]. The literature in property violations explanation within cyber-physical systems is substantial [151, 152]. However, these approaches either ask for a known set of possible failure-inducing changes up front, e.g., obtained through change identification [94] or extrapolate explanations from traces of execution rather than from the system requirements [95, 96]. Methods that do not depend on identified change types or are requirements-focused have a noteworthy potential to support resilience attainment in cyber-physical systems.

2.2.2 Representing Change

Cyber-physical systems can better understand, anticipate, and respond to runtime changes or environmental shifts by representing change since they can reason over models. Change representation enables structured approaches to handling uncertainties and guiding system responses, ultimately fostering adaptability and recovery. We discuss representing changes with modeling and specifying, and deriving concrete examples of change.

Modeling & Specifying change is key to effective failure-inducing change mitigation. The outcome of strategies for understanding change (i.e., detection, identification, and explanation) manifests in models. When modeling and specifying for resilience attainment in cyber-physical systems, it is crucial to account for dynamic behavior. Transition systems [153], such as finite state machines (FSMs), Markovian Models (e.g, Markov Chains, Markov Decision Processes and its variations), and hierarchical tree structures (e.g., Hierarchical Task Networks [154], Goal Models [155], and Behavior Trees [156,157]) are used to capture the system behaviors. Specification languages like Hybrid Logic of Signals (HLS) [158] or Signal Temporal Logic (STL) [159] offers a tractable way to specify continuous behavior formally. Recently, interest has grown in using natural language for specification in software, opening avenues for naturalness as a property of behavior specification [160]. Explicitly modeling uncertainty or embedding uncertainty annotations in the system or environmental models is important in specifying change. For instance, RELAX introduced flexible operators (e.g., MAY ... OR, AS CLOSE AS POSSIBLE TO ...) and uncertainty annotations (e.g., environmental uncertainty, behavioral uncertainty) for modeling systems. Such operators are powered by an underlying fuzzy logic [161]. On another stance, RUNE uses Parametric Markov Decision Processes (pMDPs) and partial knowledge to encode the notion of uncertain model regions to reasoning over uncertainty [162]. Solano G. et al. [86] enables uncertainty annotations in goal models using markov decision processes and parametric model checking as the underlying notation. Finally, Filippone G. et al. discussed implementing new operators in HTNs for uncertainty handling in multi-robot applications [163]. Whether these methods fully capture predictable or unpredictable changes to enable runtime reasoning remains an open question.

Exemplifying change enhances resilience in cyber-physical systems by supporting developers and testers with model problems or solutions leveraging strategies that can be (re-)used for real-world applications. In the context of self-adaptive systems, exemplifying change involves using case studies, prototypes, and simulations that reflect the kinds of changes a cyber-physical system is likely to face. The self-adaptive systems community, primarily through SEAMS, offers a rich library of exemplars for self-adaptation in cyber-physical systems. Notable examples include DeltaIoT for self-adaptation in Internet of Things [97], Platooning LEGOs as a physical exemplar of an industrial cyber-physical system [164], and DARTSim containing a simulation of a team of uncrewed air vehicles performing a reconnaissance mission in a hostile and unknown environment [165]. They are cyber-physical systems encoded with dynamic adaptation mechanisms to cope with runtime system changes. By grounding these adaptations in well-defined, scenario-driven exemplars, SEAMS research enables designers to assess the effectiveness of specific resilience tactics, improving their resilience before deployment.

2.2.3 Preparing the System for Change

Developers must prepare the system for change to attain resilience in cyber-physical systems. In other words, they must enhance the system's *readiness* to handle expected or unexpected change. Thus, we discuss designing for change and testing for change as strategies to prepare the system to handle change at runtime effectively.

Designing for change focuses on establishing strategic design decisions to accommodate anticipated and unforeseen changes. Practical design for change incorporates static and dynamic techniques, providing a comprehensive approach to preparing the system for resilience.

Static design techniques acknowledge that unpredictable changes may emerge and allocate resources to handle them, for instance, embedding redundant components or subsystems to ensure correct behavior in case a component fails or becomes compromised. Redundant components, when complementary but not identical, need to be correctly allocated to satisfy the purpose of the overall system. The literature allocation with heterogeneous components is broad; in robotics, there is a focus on task allocation [166, 167]. Ramachandran R. K. [168] proposes a method based on simulated annealing that aims to maintain the high availability of a team of quadrotor unmanned aerial vehicles with complementary sensors, subject to resource failures. Similarly, Chen C. [169] study and implement an architecture for rejecting actuator faults and their effect in the synchronization of heterogeneous multi-agent systems. Still, on static techniques, code-level approaches may also prepare the system to handle runtime changes [170]. There is, however, a lack of research corpus on implementation strategies that bring possibly emerging changes from runtime to design, while they have a great potential to mitigate runtime faults.

Dynamic design techniques feature self-adaptation [171, 172]. Designing for self-adaptation plays a key role in sustaining runtime changes; such techniques aim at re-configuration, changing parameters, and modifying the system structures [91, 92, 173]. To this end, self-adaptation techniques involve continuously monitoring, analyzing, and acting on the system under adaptation [174].

Control-based self-adaptation is a promising line of research offering mathematically grounded adaptation with guarantees of stability and timing constraints from control theory [93, 175, 176]. As an example of using control-based self-adaptation to attain resilience by design, Papadopoulos A. et al. explored the applicability of control theory for the design of resilient run-time scheduling algorithms for mixed-criticality systems, allowing for guaranteed service to concurrently to multiple hard real-time criticality-cognizant servers [177].

Testing for change is equally essential, as it provides assurance that the system will respond correctly to various change events. Testing techniques to attain resilience follow three paradigms: field-based testing, scenario-based testing, and runtime verification, which allow developers to assess the system's runtime behavior.

Scenario-based testing is the dominant paradigm of black-box testing, where scenarios are used to check how the system copes with both nominal and non-nominal situations, such as environmental shifts or operational faults [178]. This paradigm aims to collect assurance cases by simulating the interaction of the system under test with static or dynamic objects from the operational environment. A notable case of scenario-based testing for attaining resilience in autonomous driving vehicles stems from the works on SCENIC [179, 180]. The approaches rely on probabilistic modeling of the dynamic objects (vehicles and pedestrians) to challenge the vehicle under testing within a realistic setting in simulation. Other works follow a similar direction [181, 182]. Although powerful, scenario-based testing depends on effectively describing the behavior of the scenario, including the dynamic objects; how to encode complex enough behavior on such objects is an open challenge.

Field-based testing is a technique that may use information from the real world [65]. This testing paradigm aims to bridge the gap between lab environments and the system's operational environment under test. Test cases can use information from the field or can be fully generated in the field. Field testing can also happen on a separate instance running in production, i.e., offline testing. For example, a study builds on ChaosMonkey (i.e., field testing technique from Netflix [183]) for developing a fault injection mechanism for testing the robustness of radio base stations similarly in the running system [184]. Notably, recent works study self-adaptive testing in the field, which carries a potential for resilience attainment through testing [185, 186]. However, the approach has not yet been studied in the context of cyber-physical systems.

Runtime verification employs monitoring techniques to detect and address deviations from expected behavior while the system operates, ensuring that it meets the requirements [187, 188]. Runtime verification is more lightweight than other forms of verification, e.g., model checking [153], and aims to check system properties over runtime data, either online or offline. Runtime verification techniques also enable the enforcement of rules over the system to help maintain correct service delivery. A noteworthy approach using runtime verification to attain resilience in cyber-physical systems is RUNE^2 [162], which focuses on equilibrium verification through Bayesian inference to reason about the ability of the system to remain viable. Their approach has shown promising results in the realm of search-and-rescue robotics though still needs to be tested in the field.

2.3 Related Terms and Definitions

Resilience as a property of software systems emerges from previous research on what researchers refer to as fault-tolerance and dependability [19]. The evolution of research on resilience cross-cuts many areas of computer science research, leading to different terms with similar meanings. This section mainly discusses terms similar to resilience, correlated to resilience, or from which resilience derives. The topics described here consist of self-sustained areas of research that advance in parallel to resilience research within computing systems. We discuss the terms in Fig. 2.5 comparing them to resilience.

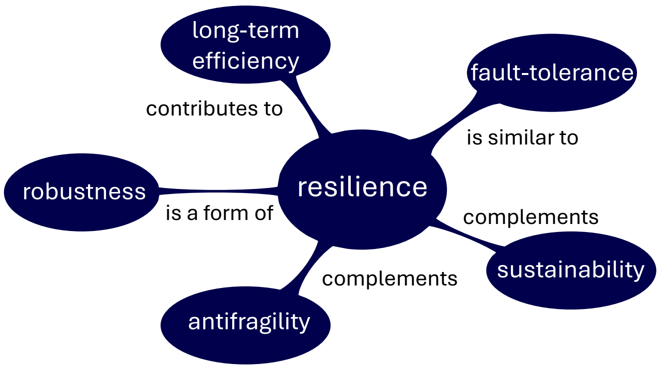


Figure 2.5: Resilience and related terms.

In short, long-term *efficiency* contributes to resilience. While an efficient system can optimize resource use and operate within constraints, resilience focuses on the system's ability to recover and adapt to disruptions. *Fault-tolerance* has been used as a synonym for resilience. However, use ignores that resilient systems face the unexpected, while fault-tolerant systems do not. *Robustness* is a form of resilience. While robust systems can absorb disturbances and continue operating within acceptable performance bounds, resilience is more flexible. *Antifragility* complements resilience by focusing on growth in response to stress. While resilience is about withstanding faults and returning to a stable state, antifragility thrives on these faults, using them as opportunities for improvement. *Sustainability* emphasizes the long-term viability, complementing resilience. Resilience emphasizes a system's ability to withstand and recover from specific faults or disturbances, while sustainability focuses on optimizing resource usage and maintaining efficiency over time.

Next, we comprehensively describe each of the five related terms. We provide definitions and examples of how to attain such properties in computing systems, focused on cyber-physical systems, when possible, and how they compare to resilience.

2.3.1 Efficiency

Long-term efficiency contributes to resilience. A system is considered efficient if it minimizes the waste of resources, such as time, energy, and computational power, while achieving its intended functionality. In energy-efficient software systems, the optimal use of computational resources ensures that the system can continue to operate effectively even when power availability is reduced, making it more resilient to environmental changes. For example, consider a mobile robotic system operating in an environment with limited battery power. An efficient mobile robot would prioritize essential functions like navigation and obstacle avoidance while minimizing energy consumption for non-critical tasks. This allows the robot to extend its operational time without frequent recharges or interruptions, ultimately increasing its ability to sustain long-term operations.

Definition – Efficiency

Efficiency is *"the probability that an intended chain of actions produces a particular outcome while minimizing the use of resources"* [189].

In computing, the literature on efficiency discusses energy efficiency as a software system property [190–192]. It also discusses time efficiency in the context of testing and verification programs [193, 194] and algorithmic efficiency [26]. Energy efficiency is significant in cyber-physical systems such as mobile and embedded systems, where power constraints limit the systems' performance [195]. Efficiency, as a property of software-intensive systems are typically attained through various strategies, including optimization algorithms [193], resource-aware allocation [196, 197], and energy-efficient coding practices [198–201]. These approaches ensure that systems can perform their intended functions while minimizing resource usage, contributing to short-term performance and long-term resilience.

This thesis critiques the traditional focus on (short-term) efficiency in the design of cyber-physical systems as opposed to emphasizing long-term efficiency, which supports resilience. While an efficient system can optimize resource use and operate within constraints, resilience focuses on the system's ability to recover and adapt to disruptions. Long-term efficiency, therefore, supports resilience by reducing the strain on resources and prolonging system functionality in dynamic or adverse environments.

2.3.2 Fault-Tolerance and Dependability

Fault-tolerance has been used, in the past, as a synonym of resilience. A system is fault-tolerant if its programs can be correctly executed despite logic faults [16]. For example, consider a fault-tolerant automated driving system, e.g., the Tesla vehicle from Section 2.1.3. In the event of a sensor failure, the ADS could rely on redundancy from other sensors to maintain its operational capabilities. Despite the fault, it could still navigate safely by leveraging sensor fusion to replace the missing data and ensure no collisions occur. This is a typical scenario where a fault-tolerant system can handle component-level failures without impacting the overall safety and mission objectives.

Definition – Fault-Tolerance

Fault-Tolerance is *"the persistence of service delivery despite the occurrence of logic faults"* [202]

Fault-tolerance[8] has been broadly studied within computer science, featuring robotics [203], distributed computing [204], operating systems [205], real-time systems [206, 207], and others. It lays important foundations for computer science and engineering as is. Fault tolerance as a property of computing systems, or dependability, is typically attained by design techniques. For example, error detection and recovery [208, 209], redundancy [210–212], and checkpointing and rollback recovery [213, 214]. Error detection ensures that faults are caught early, while recovery mechanisms, such as rollbacks or retries, are triggered to restore the system to a safe state [215]. Rollback-recovery, for instance, involves design redundancy, and checkpointing periodically saves the state of a system so that it can recover from that point in the case of an emerging fault [216].

Fault-tolerant systems are robust to faults through error handling because they are prepared for known failure modes. Such use ignores that resilient systems face the unexpected while fault-tolerant systems do not.

2.3.3 Robustness

Robustness is a form of resilience. A system is robust if, for any change, the system state displacement is not enough to overcome the acceptance criteria for which the system was designed [67, 217]. For example, consider a high-traffic e-commerce platform during a sale event. A sudden increase of concurrent users entangles the server infrastructure, with each user interaction generating many requests. In the face of such a stressful situation, a robust system would remain operational and accessible, even under extreme load. It would successfully handle the increased request volume, ensuring uninterrupted customer service. A robust server under these circumstances demonstrates the ability to maintain core functionality despite significant environmental stress.

Definition – Robustness

Robustness is *"the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions"* [218]

The literature on robustness is extensive [218, 219]. In task scheduling, for example, plan robustness generally means that it can be executed and will lead to satisfying results despite changes in the environment [220]. As a property of software systems, robustness is attained by design or testing. Design-based robustness incorporates architectural tactics such as redundancy, isolation, and encapsulation to ensure that systems can withstand unexpected inputs and operational stress from the ground up [67, 221, 222]. For example, Huhns et al. investigate the design of robust multi-agent systems, showing how redundancy

[8]Fault-tolerance is used interchangeably with dependability in this thesis. Despite the similarity, they are different terms. Fault tolerance is means to dependability [16].

and fault-tolerant agent interactions can create resilience even in complex, dynamic environments [68,69,223]. On the testing side, robustness testing forms a substantial portion of test suites in mature software – often around 80% [224]. Techniques such as fuzz testing, stress testing, and fault injection are essential for robustness assurance, subjecting systems to boundary conditions and abnormal inputs to reveal vulnerabilities [225–227]. For distributed embedded systems with limited computational resources, targeted fault injection tests can be employed to evaluate robustness under constrained conditions [184].

Although robustness is fundamental, it does not encompass all aspects of resilience. While robust systems can absorb disturbances and continue operating within acceptable performance bounds, resilience is more flexible. In other words, a resilient system remains operational in the face of faults and can degrade gracefully and recover to its original state after a disruption. Robustness can be seen as a foundational element of resilience but does not fully address its capacity.

2.3.4 Antifragility

Antifragility complements resilience by focusing on growth in response to stress. While resilience focuses on a system's ability to recover from disruptions and maintain acceptable performance, antifragility refers to a system's capacity to gain strength or improve when exposed to volatility, stressors, or faults [228]. A system is antifragile if it benefits from the same factors that might cause other fragile software systems to degrade [229]. For instance, consider an autonomous drone fleet deployed for environmental monitoring. When subjected to strong winds or sensor malfunctions, an antifragile fleet would not merely compensate for these disturbances but actively improve the flight stability control algorithm to better handle similar conditions in the future.

Definition – Antifragility

Antifragility is *"the ability of a system to leverage disturbances to self-improve and enhance its resilience over (run)time"* [230].

Antifragility is growing as a research field in computer science and engineering [231]. In the work of Monperrus M. [232], principles of antifragile software systems emphasize learning from runtime disruptions. The author suggests automated software repair [233–235] or runtime fault-injection [236, 237] to enhance the software system's antifragility. Similarly, Botros J.S. [238] demonstrates how chaos engineering, adaptation, and monitoring can be intertwined in a framework to foster antifragility in cloud computing systems. Another example is the use of machine learning in autonomous systems to evolve in the face of failure [239], which can thrive when supported by the mathematical grounding from control theory [240].

Antifragility differs from resilience in its approach to disruption. While resilience is about withstanding faults and returning to a stable state, antifragility thrives on these faults, using them as opportunities for improvement. An antifragile system evolves through adversity, developing enhanced capabilities beyond recovery or restoration.

2.3.5 Sustainability

Sustainability complements resilience by emphasizing long-term viability through resource maintenance. A sustainable system can maintain its operational quality and performance over extended periods while efficiently utilizing the computational resources and maintenance efforts [241]. For example, sustainable ADS software would efficiently process data from the visual perception sensors (i.e., cameras, radar, and LiDAR), ensuring reliable performance for collision detection while minimizing power and resource usage. As the vehicle's environment changes, for instance, due to changes in seasons or new sensors are added, the system would continue to operate effectively without requiring significant hardware upgrades or excessive energy consumption. This ensures long-term performance with minimal resource strain.

Definition – Sustainability

Sustainability is *"the capacity of a system to endure and evolve while minimizing resource consumption, maintaining efficiency, and ensuring long-term performance"* [242].

Sustainability grows as a pivoting area within computer science research [243–246]. In cyber-physical systems, techniques to attain sustainability may be implemented through processor architecture for optimized energy usage in task scheduling [247–249]. On the software side, sustainability has been studied from the point of view of architecture and requirements. For instance, system sustainability can be attained by proposing design decisions that tackle the "sustainability debt" [250] in the AUTOSAR[9] architecture for automotive vehicles or promoting decision maps to understand the effects of design decisions to sustainability of the software system [251]. Moreover, the sustainability of software systems can be assessed using non-functional requirements [252] or implemented following industry-compliant guidelines [253]. Penzenstadler, B. lays means to leveraging sustainability within requirements engineering for software systems [254, 255], for instance, by introducing the environmental sustainability goals from stakeholder finding and domain analysis toward usage model and system requirements.

While sustainability and resilience share a common goal of ensuring a system's long-term viability, the two concepts diverge in their primary focus. Resilience emphasizes a system's ability to withstand and recover from specific faults or disturbances, while sustainability focuses on optimizing resource usage and maintaining efficiency over time.

[9]http://www.autosar.org

Chapter 3

Summary and Discussion

Attaining resilience is a matter of strategy. A joint effort between researchers and practitioners casts the necessity to prioritize scientific and technological advancement toward human-centric resilient computing. Contributions to resilience attainment in computing systems pave the way towards that. This chapter lists and summarizes the research work that amounts to the thesis. The chapter also discusses our contributions and impact on research and practice when possible. Finally, the chapter synthesizes possible avenues for future work.

3.1 Summary of Original Work

This section presents a summary of the papers (**A**–**I**) highlighting the main contributions, how they support the strategies (**STG1**–**STG7**), and answers to questions (**RQ1**–**RQ4**). Each paper is briefly described, focusing on the novelty and the paper's evaluation results. We also emphasize the main contributions of the author of the thesis to each paper.

RQ	Strategy	Paper	CPS Domain	Contributions	Evaluation
RQ1	**STG1**	**A**	ROS-based Robotics	• Guidelines for Runtime Assessment • State-of-the-art and practice	• User study
RQ2	**STG2**	**B**	Mobile Robotics Multi-Robot Systems	• Architecture for Multi-Robot Systems	• Experimentation • Guideline-based analysis
	STG3	**C**	Automotive Autonomous Driving	• Driver-Vehicle Model • Behavior trees for testing	• Static analysis • Experimentation
RQ3	**STG4**	**D**	Healthcare Body Sensor Network	• Control Theory and AI • Field data for PID tuning	• Experimentation
	STG5	**E**	Robotics Manipulation	• Patterns for robotics coding • A case study of skill-based dev.	• User study
	STG6	**F**	Healthcare Body Sensor Network	• Immune-inspired explainability	• Feasibility experiment
		G	Automotive	• Automated Trace Diagnostics • Evolutionary search for diagnostics	• Experimentation
RQ4	**STG7**	**H**	Healthcare Body Sensor Network	• Exemplar for experimentation	• Feasibility experiment
		I	Robotics	• Repository of mission exemplars • Uncertainty in robotics missions	-

Table 3.1: Summary of contributions from papers A-I.

Paper A– R. Caldas, J. A. P. García , M. Schiopu , P. Pelliccione, G. Rodrigues, T. Berger, "Runtime Verification and Field-based Testing for ROS-based Robotic Systems", In: Trans. Softw. Eng. (TSE), IEEE, 2024.

In paper A, we propose a catalog of 20 guidelines for runtime assurance of ROS-based robotic systems (Fig. 3.1) by looking at the scientific literature and mined ROS code, using the lenses of Runtime Verification [256] and Field-based Testing [65]. Grounded on theory, the guidelines deliver practical recommendations to equip robotics developers with the means to develop and collect assurance arguments on ROS-based systems. In addition, the paper comprehensively assesses the state-of-the-art and state-of-the-practice runtime assurance for ROS systems. We show that our guidelines are sound by conducting a user study that received 55 responses from practitioners and academics with experience in robotics development. The results show that the guidelines were highly appreciated regarding applicability, usefulness, and clarity. We offer an overview of existing methods and tools for testing ROS-based systems, focusing on how they address the challenges of field-based testing and runtime verification. Ultimately, the paper provides insights and avenues for future research. For more information, check the original work in Chapter 4.

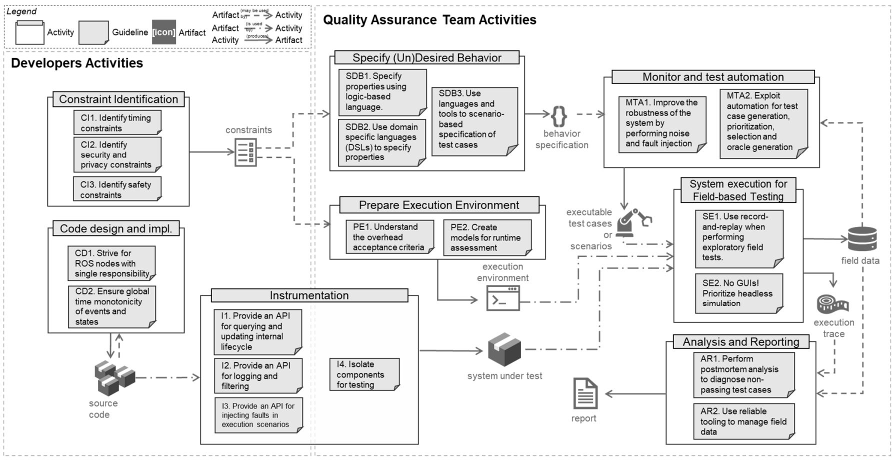

Legend
Activity
Guideline
[icon] Artifact
Artifact (may be used by) Activity
Artifact (is used by) Activity
Activity (produces) Artifact
Developers Activities
Constraint Identification
CI1. Identify timing constraints
CI2. Identify security and privacy constraints
CI3. Identify safety constraints
constraints
Code design and impl.
CD1. Strive for ROS nodes with single responsibility
CD2. Ensure global time monotonicity of events and states
source code
Instrumentation
I1. Provide an API for querying and updating internal lifecycle
I2. Provide an API for logging and filtering
I3. Provide an API for injecting faults in execution scenarios
I4. Isolate components for testing
Quality Assurance Team Activities
Specify (Un)Desired Behavior
SDB1. Specify properties using logic-based language.
SDB2. Use domain specific languages (DSLs) to specify properties
SDB3. Use languages and tools to scenario-based specification of test cases
behavior specification
Monitor and test automation
MTA1. Improve the robustness of the system by performing noise and fault injection
MTA2. Exploit automation for test case generation, prioritization, selection and oracle generation
Prepare Execution Environment
PE1. Understand the overhead acceptance criteria
PE2. Create models for runtime assessment
executable test cases or scenarios
execution environment
system under test
System execution for Field-based Testing
SE1. Use record-and-replay when performing exploratory field tests.
SE2. No GUIs! Prioritize headless simulation
field data
execution trace
Analysis and Reporting
AR1. Perform postmortem analysis to diagnose non-passing test cases
AR2. Use reliable tooling to manage field data
report

Figure 3.1: Catalog of Guidelines for Runtime Verification and Field-based Testing for ROS-based Robotic Systems. Note: originals are in Chapter 4

Paper A solely realizes the strategy **STG1** and answers to the research question **RQ1**.

Individual Contribution: The author of this thesis led the work. He conceived the original ideas of the work, designed the methodology, conducted the analysis of literature in partnership with another researcher, and the analysis of the mined codebases with others. He led the writing of the paper and was responsible for validating and visualizing the data and results.

Paper B– G. S. Rodrigues, R. Caldas, G. Araujo, V. de Moraes, G. Rodrigues, P. Pelliccione, "An Architecture for Mission Coordination of Heterogeneous Robots", In: J. Syst. Softw. (JSS) 191: 111363 (2022), ACM, 2022.

Paper B proposes MissionControl, an architecture to coordinate missions to automatically form coalitions of heterogeneous robots, shown in Fig. 3.2. Our proposed architecture has been defined to satisfy key attributes of modifiability and integrability by decoupling coordination and task execution. Coordination builds on ensemble-based systems [257] while for task execution, we leverage behavior trees [156] as the technology for skill-based execution. Empirically, we show that MissionControl leads to 57,63% more complete missions, on average, compared to a simple task allocation mechanism. We ran our experimental apparatus for 81 scenarios (totaling 648 trials). The mission could fail due to 'no skill' found to complete the mission, robots could stop due to lack of battery, the trial could timeout, or the robot could get stuck in walls. In addition, we show that our architecture complies with best practices on robotic architecture design, following a guideline-based evaluation. Check the original work in Chapter 5 for more information.

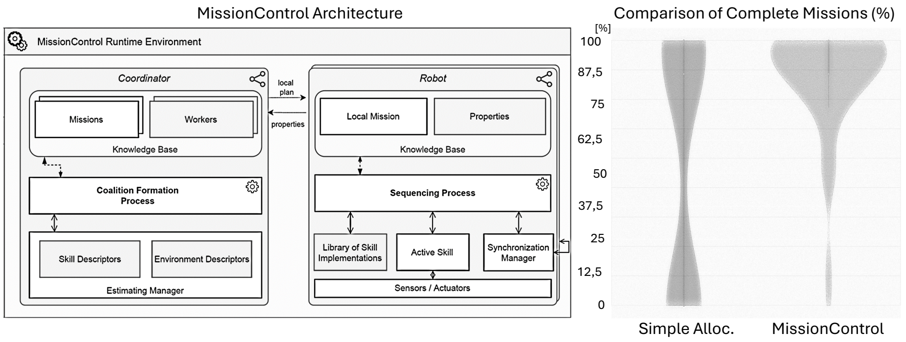


Figure 3.2: Overview of the MissionControl architecture and results from the empirical assessment of the implementation. Note: originals are in Chapter 5

Paper B realizes the strategy **STG2** and, jointly with **Paper C**, answers to the research question **RQ2**.

Individual Contribution: The author of this thesis contributed to the work led by another researcher, namely G. S. Rodrigues from the University of Brasilia, Brazil. In this work, the thesis author contributed to discussions on conceptualizing the main idea. He focused mainly on the experimental setting and planning the evaluation work. His work included outlining the experiments, preparing for data analysis and visualization, and reporting the results. Particularly, he contributed to the feasibility evaluation pipeline and the guideline-based evaluation of the architecture.

Paper C – R. Queiroz, D. Sharma, R. Caldas, K. Czarnecki, S. García, T. Berger, P. Pelliccione, "A Driver-Vehicle Model for ADS Scenario-based Testing", Trans. on Intell. Transp. Syst. (ITS) 25(8): 8641-8654, IEEE, 2024.

In this paper, we propose the GeoScenario Simulated Driver-Vehicle model to specify and simulate the realistic behavior of human-operated vehicles in testing, offering high expressiveness, execution accuracy, scalability, and reuse. Illustrated in Fig. 3.3, the simulated driver-vehicle model extends the GeoScenario [73], an autonomous driving scenario simulation environment by using the driver behavior model proposed in Michon et al. [258], in which the maneuver selection logic is implemented using behavior trees [259].
Empirically, we show that the model can successfully express with over 80% of reuse and accurately execute all eighteen National Highway Traffic Safety Administration's (NHTSA) vehicle-to-vehicle pre-crash scenarios (except one variant). We also show that, after calibration, the model can produce maneuver decisions and trajectories that closely resemble those from recorded real-world traffic concerning Spatial-Temporal Euclidean Distance (STED) to the expected trajectories. The model also scales in scenarios with up to 10–20 simultaneous and highly interactive vehicles in real-time simulation. For more information, check the original work in Chapter 6.

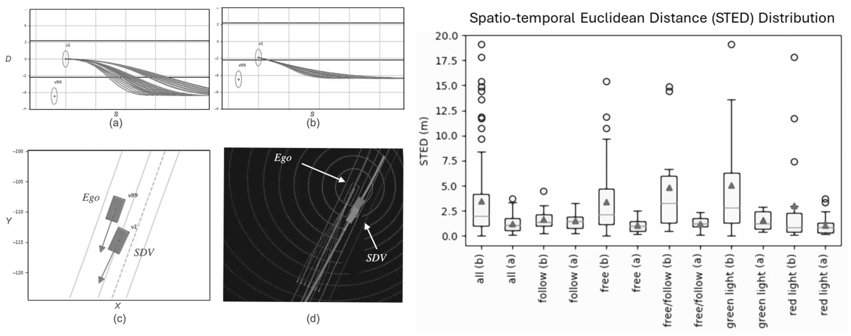


Figure 3.3: Cut-in scenario simulated in GeoScenario and results from the empirical assessment of realism of scenarios. Note: originals are in Chapter 6

Paper C solely realizes the strategy **STG3** and, jointly with **Paper B**, answers to the research question **RQ2**.
Individual Contribution: The author of this thesis contributed to the work led by another researcher, namely R. Queiroz from the University of Waterloo, Canada. The thesis author contributed to implementing a significant portion of the design artifact. Notably, the thesis author implemented the internal domain-specific language based on behavior trees to allow for the specification of human-driver behavioral models with reuse and flexibility. In addition, the thesis author evaluated the work, contributing to the design and execution of experimental trials, data collection, analysis, visualization, and reporting. The thesis author focused on assessing the expressiveness and level of reuse of the designed human-driver specifications. This thesis's author also contributed to manuscript revisions during the peer-review process.

Paper D– R. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. N. Vogel, P. Pelliccione, "A Hybrid Approach Combining Control Theory and AI for Engineering Self-Adaptive Systems", In: Proceedings of the 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE/ACM, 2020

This paper proposes intertwining control theory and AI in a two-phase optimization approach to support the engineering of control-based self-adaptation systems. Depicted in Fig. 3.4, our control-based self-adaptation architecture relies on two layers: strategy management and strategy enactment. The strategy management layer utilizes parametric formulae of the system requirements in goal model notation [86]. The strategy enactor follows a typical proportional-integral-derivative (PID) controller and enacts the selected strategy by fine-tuning the knobs of (a.k.a. re-configuring) the application at runtime. Both layers used the Non-dominated Sorting Genetic Algorithm (NSGA-II) [260] for synthesizing adaptation strategies and finding optimal actions to enact the change in the system. Empirically, we show that the NSGA-II algorithm can effectively find nearly optimal configurations for the PID controller, which, in turn, results in a self-adaptive system capable of resiliently adapting in the face of various kinds of uncertainties in a body sensor network example (from paper I). Check the original work in Chapter 7 for more information.

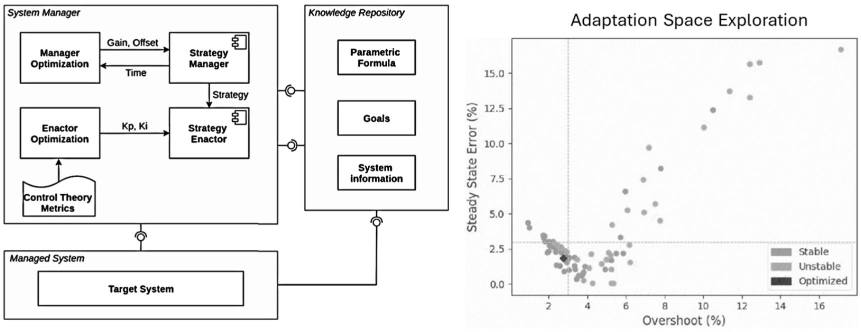


Figure 3.4: The hybrid approach to control-based adaptation and the empirical results of understanding whether the tool finds nearly optimal configurations. Note: originals are in Chapter 7.

Paper D solely realizes the strategy **STG4** and, jointly with **Papers E, F, G**, answers to the research question **RQ3**.

Individual Contribution: The author of this thesis led this work. The thesis author conceptualized the idea with another researcher from the team. The author of this thesis also implemented the hybrid approach and developed the experimental setting to gather empirical evidence that the approach is sound. The author of the thesis collected the data, analyzed, reported, and created the visualizations.

Paper E– M. Rizwan, C. Reichenbach, R. Caldas, M. Mayr, V. Krueger, "EzSkiROS: Enhancing Robot Skill Composition with Embedded DSL for Early Error Detection", In: Frontiers in Robotics and AI (RAI), Frontiers, 2024

Paper E proposes EzSkiROS, an embedded domain-specific language to support the development of robotic applications. Depicted in Fig. 3.5, EzSkiROS is an extension of SkiROS2 [261], a framework for specifying robotics control code driven by skills implementations. EzSkiROS results from a case study in which we promote four design patterns for embedded domain-specific languages, namely: domain language mapping [262], early dynamic checking [263], symbolic tracking [264], and source provenance tracking [265]; addressing common challenges in robotic software development. Through a user study, we show that EzSkiROS helped to catch bugs due to mismatching environmental assumptions and that the resulting specification is more readable, clear, and concise. Check the original work in Chapter 8 for more information.

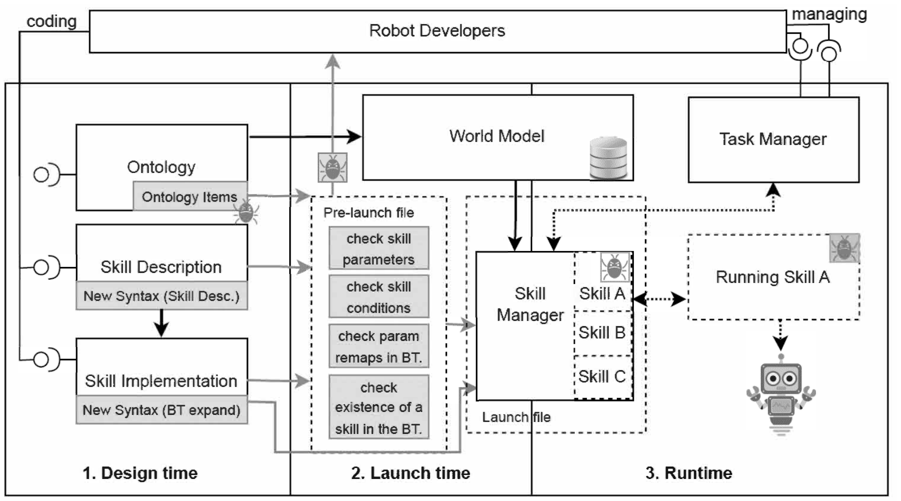


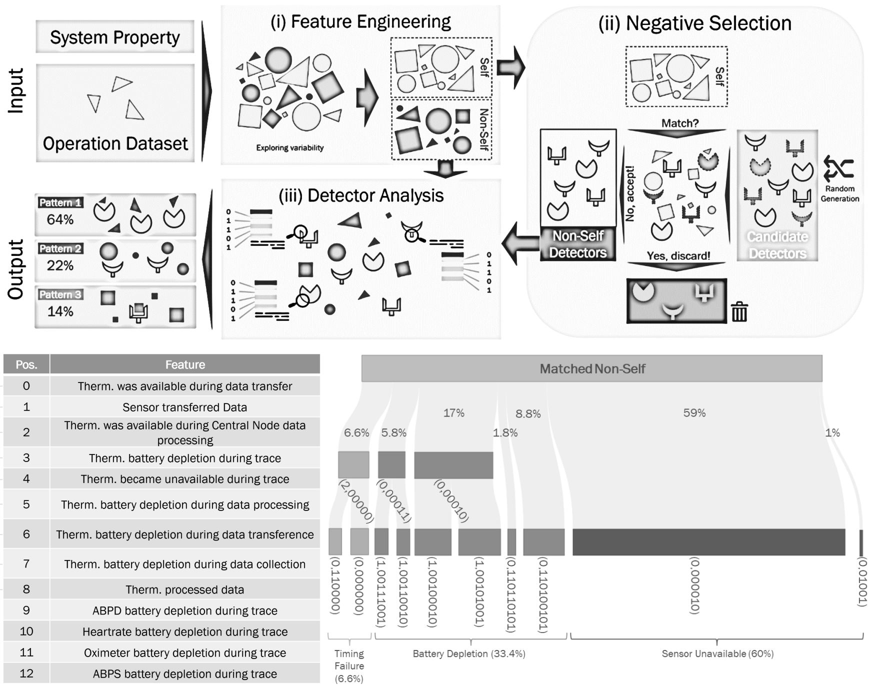
Figure 3.5: Simulation of the robotic arm UR5 and EzSkiROS. Note: originals in Chapter 8

Paper E solely realizes the strategy **STG5** and, jointly with **Papers D, F, G**, answers to the research question **RQ3**.

Individual Contribution: The author of this thesis contributed to the work led by another researcher, namely M. Rizwan from the University of Lund, Sweden. The thesis author participated in conceptualizing the ideas that resulted in the paper, supported by pair programming in some of the core features of the work. The thesis author also participated in planning and running the evaluation section of the work, delving into data analysis, reporting, and visualizations of the approach. The author also supported paper writing, especially the related work section, and revisions during peer review.

Paper F– J. P. C. de Araujo, G. N. Rodrigues, M. Carwehl, T. Vogel, L. Grunske, R. Caldas, P. Pelliccione„ “Explainability for Property Violations in Cyber-Physical Systems: An Immune-Inspired Approach”, In: IEEE Software 41(5): 43-51, IEEE, 2024

This paper proposes an immune-inspired approach to automated explainability derivation from traces. Figure 3.7 illustrates the approach in three steps: (i) Feature Engineering, (ii) Negative Selection, and (iii) Detector Analysis. This pipeline follows the Negative Selection Algorithm [266]. The input to the automated explanation analysis tool is a system property specified in first-order logic, and the operational dataset consists of (multiple) execution traces derived from simulated system runs. This paper analyzes the feasibility of the solution in a body sensor network system (from paper H). Our analysis diagnosed that features related to the sensor’s unavailability, battery depletion, and timing failure were relevant factors that contributed to anomalous behaviors regarding the property at hand. Check the original work in Chapter 9.



Pos.	Feature
0	Therm. was available during data transfer
1	Sensor transferred Data
2	Therm. was available during Central Node data processing
3	Therm. battery depletion during trace
4	Therm. became unavailable during trace
5	Therm. battery depletion during data processing
6	Therm. battery depletion during data transference
7	Therm. battery depletion during data collection
8	Therm. processed data
9	ABPD battery depletion during trace
10	Heartrate battery depletion during trace
11	Oximeter battery depletion during trace
12	ABPS battery depletion during trace

Figure 3.6: Overview of the pipeline to automated explanation using the negative selection algorithm. Note: originals in Chapter 9

Paper F jointly with paper G realizes the strategy **STG6** and, jointly with **Papers D, E, G**, answers to the research question **RQ3**.

Individual Contribution: The author of this thesis contributed to the work led by another researcher, J. P. C. Araujo, from the Humboldt University Berlin, Germany. The thesis author contributed by discussing the adoption of the negative selection algorithm. He also assisted with writing and reviewing the manuscript.

Paper G– G. Araujo, R. Caldas, F. Formica, G. Rodrigues, P. Pelliccione, C. Menghi, “Search-based Trace Diagnostic” *(submitted)*

Paper G proposes a search-based trace diagnostics method. Figure 3.7 illustrates a scenario that violates a safety property, and the explanation is automatically derived using the search-based trace diagnostics method. Departing from a violated requirement, specified in hybrid logic signals [158], and a trace, the method exploits genetic algorithms to generate valid mutations of the violated requirement and, consequently, derive diagnostics to the property violation. Significantly, the genetic algorithm relies on the Smith-Waterman algorithm [267] as the fitness function. The C4.5 algorithm [268], available at Weka [269] that builds a decision tree containing the diagnostics, the semantics of the nodes on the root of the tree are more likely to have strongly influenced the property violation. Empirically, we evaluated our solution by performing 34 experiments involving 17 trace-requirement combinations that led to a property violation from two systems from the automotive domain and one from the robotic domain. Our results show that Diagnosis can produce informative diagnoses within a practical time (47 hours) for most of our experiments (33 out of 34). Check the original work in Chapter 10 for more information.

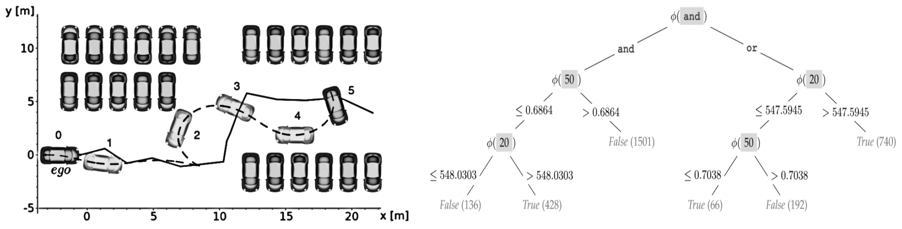


Figure 3.7: Scenario violating the requirements and derived diagnostics.
Note: originals in Chapter 10

Paper G jointly with paper F realizes the strategy **STG6** and, jointly with **Papers D, E, F**, answers to the research question **RQ3**.

Individual Contribution: The author of this thesis contributed to the work led by another researcher, namely G. Araujo from the University of Brasilia, Brazil. The thesis author contributed with the initial ideas of the work and the concrete solution design. Particularly, he assisted with designing and framing the solution within the genetic algorithm paradigm by participating in initial technology spikes and discussing the implementation with the leading author. In addition, he contributed to interpreting the diagnostics results in the format of decision trees. The thesis author also assisted with the design of the experiments for validation, data analysis, and reporting. More specifically, it concerns the experimentation goals, the semantics of effectiveness and efficiency, and the comparison to human-driven diagnostics. He also assisted with writing the related work section and revising the manuscript.

Paper H– E. B. Gil, R. Caldas, A. Rodrigues, G. L. G. da Silva, G. N. Rodrigues, P. Pelliccione "Body Sensor Network: A Self-Adaptive System Exemplar in the Healthcare Domain" In: Proceedings of the 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE/ACM, 2021

Paper H proposes the BSN, a self-adaptive body sensor network exemplar in healthcare. The BSN features in the list of self-adaptation exemplars of the community on engineering adaptive and self-managing systems. Interestingly, the paper contributes the only exemplar that leverages a replaceable, self-adaptation based on control theory [175]. In addition, the BSN was implemented in the robot operating system (ROS) [270], which ties the application to the de facto development environment used in robotics. Finally, it embeds a node dedicated to uncertainty injection for experimentation in the self-adaptive domain with the capacity to simulate all four kinds of uncertainty sources: changing system goals, environmental changes, and changes in the system itself [84]. Our exemplar is easy to install and deploy, offering instructions on building locally and using a prepared virtual machine or docker. The exemplar from Paper I was used in papers D and G for validation purposes. Check the original work in Chapter 11 for more information.

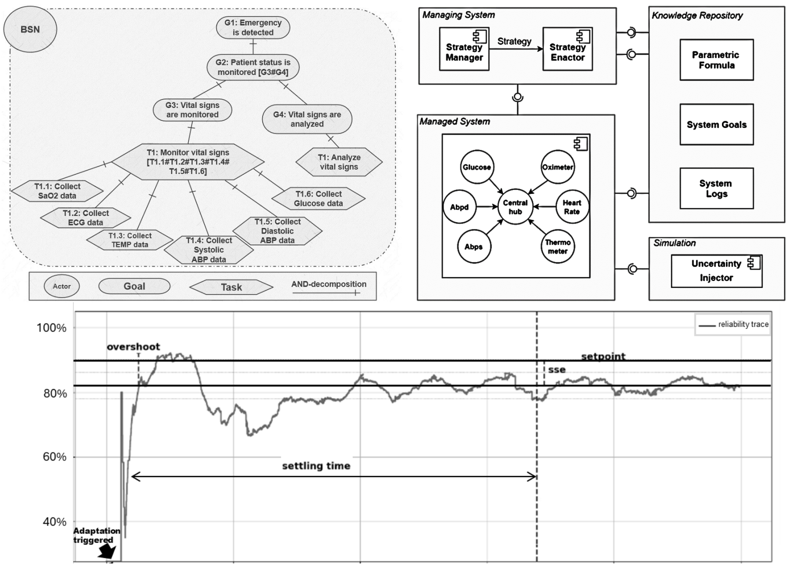


Figure 3.8: Overview of the requirements and architecture of the body sensor network. Note: originals in Chapter 11

Paper H jointly with paper I realizes the strategy **STG7** and, jointly with **Papers I**, answers to the research question **RQ4**.

Individual Contribution: The author of this thesis led this work. The thesis author conceptualized the idea and implemented the design artifact, including all available components. He led the writing, reporting, visualizations, and packaging of the artifact for sharing.

Paper I– M. Askarpour, C. Tsigkanos, C. Menghi, R. Calinescu, P. Pelliccione, S. García, R. Caldas, T. J. von Oertzen, M. Wimmer, L. Berardinelli, M. Rossi, M. M. Bersani, G. S. Rodrigues, "RoboMAX: Robotic Mission Adaptation eXemplars" In: Proceedings of the 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE/ACM, 2021

Paper I proposes RoboMAX, a series of natural language mission specifications to support the development of robotic applications subject to uncertainty. RoboMAX serves as a repository of scenarios collected from academics and practitioners working in domains such as hospitals and warehouses. The written scenarios are further classified concerning sources of uncertainty that may affect the conclusion of the mission, depicted in Fig. 3.9. The repository is extensible and available in an online repository. Check the original work in Chapter 12 for more information.

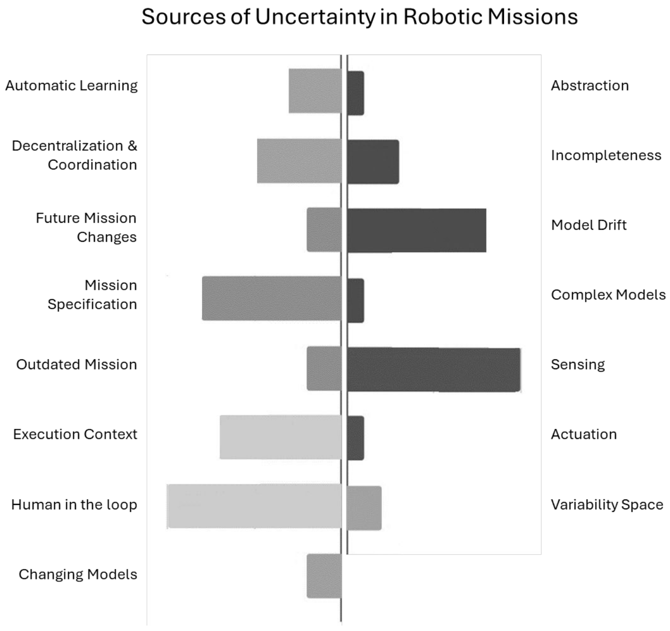


Figure 3.9: Classifications of sources of uncertainty affecting the 13 initial scenarios from RoboMAX.

Paper I jointly with paper H realizes the strategy **STG7** and, jointly with **Paper H**, answers to the research question **RQ4**.

Individual Contribution: The author of this thesis contributed to this work led by another researcher, namely M. Askarpour from McMaster University, Canada. The author of this thesis contributed by designing and writing three specifications (i.e., robot missions) that compose the collection of artifacts that amount to the repository. He assisted with classifying the specifications according to the required and implemented framework.

3.2 Discussion and Research Impact

In this section, we revisit the research questions and their answers, outlining our contributions to research and practice in resilience attainment in cyber-physical systems. Moreover, when possible, we reflect on the research impact of the contributions and papers appended to the thesis.

RQ1. How are the current practices, methods, and tools enabling runtime assessment to attain resilience in cyber-physical systems?

RQ1 is answered by the strategy STG1 and supported by paper A. The strategy STG1 argues for enacting a catalog of best practices on runtime verification and field-based testing to equip development and quality assurance teams with runtime introspection and modification capacities. With this objective, paper A gathered the state-of-the-art through a systematic literature review, enhanced by the state-of-the-practice with a repository mining and analysis. It synthesized 20 guidelines with recommendations that support the development and quality assurance of robotic systems, a particular kind of cyber-physical system. In this context, we discuss two outstanding aspects of our study: a methodological contribution and a technological contribution.

Methodologically, our study bridges the gap between theory and practice by combining a systematic literature review on fundamental software engineering technologies, i.e., runtime verification and field-based testing, and the practice of such in a framework widely used in practice, both in academia and industry, the robot operating system (ROS). In this sense, the closest comparison to our work is the work from Malavolta et al. [271], which relies purely on mining ROS repositories to collect architectural decisions. Their mapping to a theory on software architecture is implicit in the paper and limits the task of finding and exposing gaps in the literature. In addition, we believe that the guidelines should be extended to include theoretical and practical advances in the field. Therefore, we released a live version of the guidelines, hosted on an open-source website (https://ros-rvft.github.io/) with reproducibility and extensibility instructions. The protocol for extension is described and transparent and aims to maintain the quality of the guidelines catalog. As far as we know, there is no similar practice.

Technologically, our study also delivers the first recommendations for robotics testing to support such challenging tasks in robotics software development concretely. In contrast, other contributions to guiding testing in the context of robotics focus on developing practices [272], methods [273, 274], tools [275], and concrete recommendations are not evident from these works. Our proposed, loosely coupled guidelines are agnostic from the development process and can be employed by any roboticist using the robot operating system. Although not supported by concrete examples, the guidelines can be generalized to robotics and cyber-physical systems.

RQ2. How to attain resilience in cyber-physical systems with diverse and smart agents?

RQ2 is answered by strategies STG2 and STG3, supported by papers B and C, respectively. Strategy STG2 advocates separating concerns between coordination and task execution to attain robust operation, with the automated task (re-)allocation for cyber-physical systems with heterogeneous components. Accordingly, paper B promotes the automated formation of coalitions, i.e., teams of agents, focusing on ensemble formation for robust execution. The strategy STG3 argues for behavior reuse when specifying test cases for cyber-physical systems with smart agents and, thus, collects assurance cases for robust cyber-physical systems' behavior. Accordingly, paper C develops a model that supports the implementation of other dynamic and smart agents as part of the test scenario. In this context, we highlight this study's contributions compared to similar works for attaining resilience in heterogeneous and smart cyber-physical systems.

Our study uniquely prepares cyber-physical systems for operating with heterogeneous and smart agents. From the architecture side (STG2), we promote the automated composition of heterogeneous nodes in a dynamic setting that, differently from other multi-agent system architectures, enables task (re-)allocation in the face of uncertainty [276–278], our architecture is system agnostic, as long as the system's architecture allows for skill descriptions and the descriptors follow MissionControl's interface. Our study heavily inspired further work on the specification and automated decomposition of missions for robotic systems [279], an approach that integrates the human-on-the-loop for coordination of mission reconfigurations in self-adaptive systems [280], automated plan recovery in dynamic environments [281], and uncertainty handling for multi-robot applications [163].

From the assurance case collection side (STG3), we allow for a multi-layered model that unites precise, lower-level task specifications with high-level mission coordination. In contrast to less flexible approaches to implementing test scenarios, for instance, using models that do not scale as well [282], are capacity-driven (e.g., driver following maneuver [78]), or are not controllable or modifiable since they are data-driven [72]. This study motivated work on complementary models that extrapolate human driver's behavior, specifically lane change maneuvers, by evaluating available safety margins from a personal minimal acceptable gap size [283]; it also motivated the development of human (pedestrian) models [284].

RQ3. How to attain resilience in cyber-physical systems prone to uncertainty?

RQ3 is answered by the Strategies (STG4-STG6) and supported by Papers D-G. Strategy STG4 argues for embedding control-based self-adaptation in cyber-physical systems. Accordingly, paper D presents a two-layered adaptation approach with high-level strategy management and lower-level strategy enactment (using a PID controller) to reduce the effect of exposure to uncertainties. Strategy STG5 argues for preparing the codebase to catch errors early. Accordingly, paper E implements four design patterns to prepare the codebase to catch errors that are subject to environmental uncertainty early. Strategy STG6 argues for using tools to automate the derivation of diagnostics and explanations to support swift recovery. Accordingly, papers F and G propose trace-checking techniques that are either focused on collecting multiple traces or generating variations of the requirement specification to find likely causes for the failures.

In this context, we discuss the studies' contributions to mitigating uncertainty to attain resilience in cyber-physical systems.

The strategy STG4 aims to systematically mitigate uncertainty by collecting execution logs and feeding them back to tune the control-based self-adaptation engine. Differently from other approaches to automatically construct controllers for managing the software system's adaptation needs [285–287], we developed a first-of-its-kind two-layered control-based self-adaptation mechanism. Our work inspired a wave on better adaptive systems combining the mape architecture, control theory, and machine learning [239], the design of controller synthesis to deep-learning perception components [288], and the design of hybrid control algorithms for enhanced driving performance [289]. As a counterpart, there was some criticism of using the NSGA-II algorithm for tuning the controller [290], the authors of the *arXiv* document write: *"For example, a very recent work in SBSE (Search-Based Software Engineering) for SASs (Self-Adaptive Systems) [(a reference to paper D)], [...], has wrongly adapted NSGA-II to optimize a single-objective problem for SAS."*. Although the authors' reasoning is right, our solution does not solely stand on NSGA-II and would work with any other search-based algorithm.

Strategy STG5, on the other hand, aims to mitigate uncertainty by preparing the codebase to detect errors in the implementation earlier. Embedding environment assumptions in the codebase typically follow model-driven approaches [291–293]; they, however, do not always lie on the potential of embedded domain-specific languages such as employing early checking techniques or leveraging the constructs of the host language as means to ease the introduction of advanced language mechanisms in a formalism well-known by the developers–the proposed patterns uniquely from paper E address this issue in the context of cyber-physical systems. We only recently released the papers, and there was no time yet for impact in the literature or practice. Yet, we speculate that our work will shape how practitioners write code for robotic systems by mindfully bringing environmental assumptions to code and checking environmental assumption violations before runtime.

On the quality assurance side, STG6 aims to mitigate uncertainty by equipping engineers with automated tooling to support the explanation for runtime failures using trace-checking. The approach in paper G uniquely uses the negative selection algorithm to identify anomalous behavior in the data, leveraging flexibility to capture complex relationships between cyber and physical elements, differently from [294–296] that use opaque models as surrogates, whose less transparent reasoning requires an additional step by integrating explainability techniques to render comprehensible explanations. The approach in paper H uniquely uses genetic algorithms for mutating the violated formal specification, and thus, requires only one trace of execution, differently from other works that require many traces or extrapolating information coming from the traces [151, 152, 297]. The works are recent, and there is no evidence of their impact in the literature or practice; however, we speculate that automated explanations will heavily rely on trace-checking-based techniques that efficiently analyze field data against requirement specifications.

RQ4. How to develop exemplars to advance research on resilience attainment in cyber-physical systems?

RQ4 is answered by the Strategy (STG7) and supported by Papers H and I. Strategy STG7 argues for offering exemplars to support the development of new methods, tools, and evidence in the field of resilience in cyber-physical systems. Accordingly, papers H and I provided exemplars for the community on engineering self-management and adaptive systems. In this context, we discuss the studies' contributions to the community.

An extensive list of exemplars is used to advance research in self-adaptive systems.[1] Uniquely; the body sensor network presented the first exemplar showcasing control-based adaptation that uses the robot operating system (ROS) for a healthcare application. Other works are deployed in the healthcare domain, using the ROS [270], but none provided interfaces to control-theoretic solutions. The body sensor network exemplar has been widely used in further developments of research for control-based self-adaptation [298–300], as an inspiration for other exemplars [301], or research on normative requirements and explanability [302, 303]. RoboMAX, on the other side, stands on the scarcity of model problems that support understanding the needs of the robotics domain. Like a similar IoT exemplar [304], RoboMAX provides high-level requirements and key contextual information associated with the considered systems and their adaptation concerns. In this way, the RoboMAX repository serves a different purpose and complements other exemplars that offer simulators for simple robotics applications [98, 305, 306].

3.3 Threats to Validity

This thesis' arguments were constructed following a qualitative research paradigm – from nine research publications (Papers A-I), we synthesized seven strategies (STG1-STG7) to attain resilience in cyber-physical systems. We then briefly analyze the validity of the strategies, answering questions inspired by guidelines to a threat to validity assessment [307].

Do the strategies fully answer to the research questions?

Our strategies collectively address each research question, validated through user studies, case studies, and controlled experiments. STG1 relies on runtime verification and field-based testing, with validation aligning with established anomaly detection techniques. STG2 and STG3 address resilience in multi-agent systems by enhancing scalability and reuse, validated through extensive testing and analysis. While we recognize that resilience in multi-agent systems could be further supported by areas such as consensus and conflict management, STG2 and STG3 offer essential solutions to address RQ2. STG4–STG6 contribute to managing uncertainty in cyber-physical systems, focusing on uncertainty handling by prevention, adaptation, and automated analysis; other dimensions like incomplete knowledge and human-in-the-loop are not covered. Finally, STG7 provides resilience exemplars, expanding SEAMS resources and covering diverse aspects of uncertainty, like requirements and fault-injection.

[1] https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/

Are the strategies solid, i.e., built on solid grounding?

The strategies are based on nine peer-reviewed publications in renowned venues, ensuring credibility through rigorous expert review of the claims, data, and methods used. Most publications employ quantitative methods supporting transparency and reproducibility; Papers B, C, D, F, G, and H provide detailed experimental setups, online access to code and data, and, where applicable, data analysis algorithms. These steps enable independent verification, reinforcing the reliability of our findings. Qualitative research (Papers A, E, and I) was conducted with bias mitigation in mind, including triangulation and thorough documentation. The process for extrapolating the strategies from the publications is subject to bias since the researcher is also the author of the papers. To reduce author bias, we compared each strategy with comparable frameworks and findings in related work (Chapter 2).

Is there evidence that the strategies enhance resilience?

The impact of each strategy on resilience can vary depending on the system context. Nonetheless, the strategies in this thesis were designed as complementary methods to aid developers and quality assurance teams in improving resilience in cyber-physical systems. Although we did not test for effectiveness and efficiency, the strategies align with the challenges to resilience attainment discussed in the scientific literature. Moreover, we applied thorough documentation to reduce potential bias. Existing resilience frameworks, such as those by Andersson et al. [41], Trivedi et al. [115], and Sharma et al. [116], provide metrics that can be used to evaluate the strategies in a structured, statistically sound experimental setting in the future. However, as our resilience assessment was not conducted in real-world environments or other software architectures, the generalizability of these strategies may limited to the scenarios in the papers. We intend to follow up on this in the near future. As far as we know, there are no off-the-shelf experimental settings for testing resilience enhancement in cyber-physical systems, which remains a gap in the field.

Are the strategies suitable to all cyber-physical systems?

The strategies presented in this thesis address a range of cyber-physical systems, including mobile robots, robotic arms, autonomous vehicles, and healthcare sensor networks. Our contributions extend to widely used software environments, such as the Robot Operating System (ROS), a building block of various cyber-physical systems. While we have not specifically tested our strategies in domains like Smart grids, Industry 4.0, or the Internet of Things (IoT), the underlying theories and principles driving our approaches have shown promise in similar contexts. We grounded our strategies in well-established frameworks and empirical studies relevant to these additional domains. By doing so, we mitigate potential biases stemming from limited domain testing, enhancing the generalizability of our strategies. We acknowledge that further validation in these areas would be beneficial, yet we believe the foundational aspects of our strategies will adapt effectively across different cyber-physical systems.

3.4 Conclusions and Future Work

This thesis introduced a strategic approach to resilience development for cyber-physical systems, moving beyond incidental resilience that may yield only short-term stability. We propose seven strategies that address runtime assessment, the diversity and smartness of agents within these systems, taming uncertainty as an intentional approach to cope with runtime change, and exemplars that serve as resilience role models. The strategies are rooted in publications across software engineering, self-managed and adaptive systems, robotics, and transportation, encompassing quantitative and qualitative research that follows a design science methodology. Our strategies reflect substantial advances in the resilience of cyber-physical systems, addressing challenges in the field. In addition to future directions outlined within the individual studies, we propose two broader avenues for future work that emerged from this thesis.

Uncertainty-Aware Specification for Resilient Cyber-Physical Systems. Systematically using unanticipated changes as a key enabler to runtime reasoning and resilience attainment remains an open question. One potential research direction involves relaxing specifications [86, 161], allowing systems to be constructed and deployed despite incomplete knowledge of their operational environment. However, incorporating field data in uncertainty-aware models to guide runtime adaptations poses significant challenges, particularly with maintaining guarantees of correctness [308, 309]. This complexity increases in multi-agent systems with diverse, intelligent components [163]. Key questions arise: *"What methods best encode uncertainty in specifications for multi-agent systems?"*, *"How can uncertainty-aware specifications be leveraged at runtime to support resilience amid unanticipated changes?"*, *"What approaches ensure runtime guarantees of resilience as unforeseen changes emerge?"*. Exploring these questions would represent a relevant advance in the engineering of modern cyber-physical systems.

Experimental Support to Resilience Assessment of Cyber-Physical Systems. Standardized platforms for resilience assessment in cyber-physical systems remain a gap in the field. Achieving resilience requires an operational measure of "more" or "less" resilience, grounded in resilience metrics proposed in the literature [41, 115, 116]. However, these metrics often require controlled, fault-inducing scenarios and simulated uncertainties, which are challenging to scale for large or multi-agent systems. Effective assessment also depends on field data manipulation, constrained by the need for physical context or real-time system adaptation [89, 310, 311]. Further, designing thorough experiments requires decision-making around the timing, type, and sequence of fault injection, which are difficult to generalize yet essential for evaluating resilience accurately [65]. To advance resilience assessment, researchers should address questions like: *"What metrics best quantify resilience across varying conditions?" "How can field data be systematically incorporated to simulate realistic scenarios without losing experimental control?" "What best practices scale fault injections effectively for complex, multi-agent systems?* Progress in this area will strengthen cyber-physical system validation, ensuring resilience strategies perform under diverse, realistic conditions.