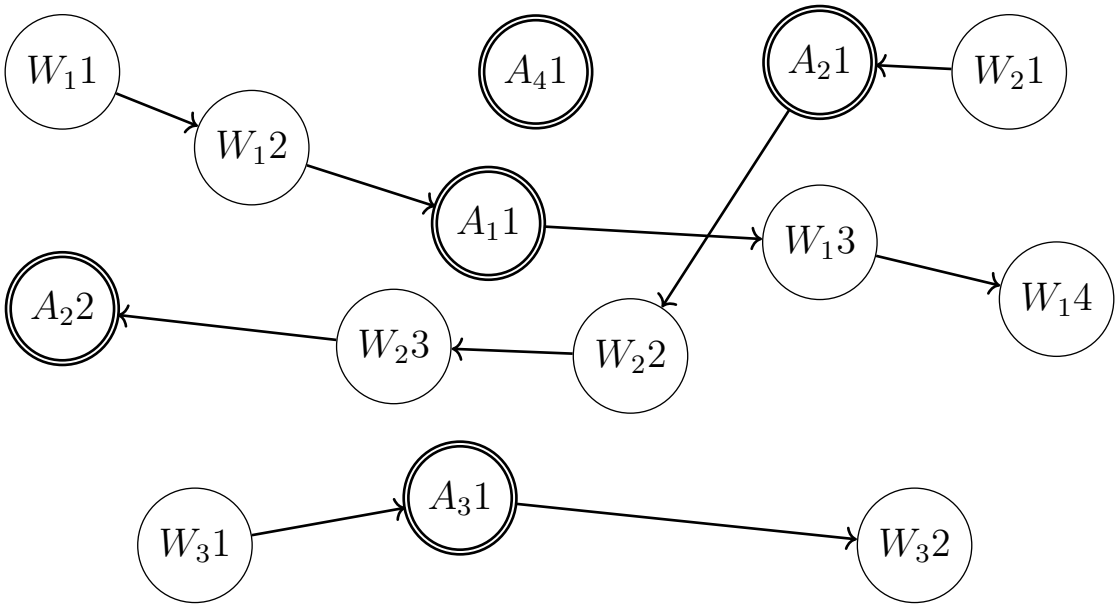
CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG



Simulation-Based Metamorphic Testing for System Verification of Cyber-Physical Systems

A Systematic Assessment of Seamless EV Charging Systems Under Dynamic Conditions

Master's Thesis in Computer science and engineering

Oscar Cronvall, Linus Pettersson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's Thesis 2026

Simulation-Based Metamorphic Testing for System Verification of Cyber-Physical Systems

A Systematic Assessment of Seamless EV Charging Systems Under Dynamic Conditions

Oscar Cronvall, Linus Pettersson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Simulation-Based Metamorphic Testing for System Verification of Cyber-Physical Systems
Oscar Cronvall, Linus Pettersson

Supervisor: Linda Erlenhov, Department of Computer Science and Engineering
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Oscar Cronvall, Linus Pettersson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The increasing adoption of electric vehicles has driven demand for reliable public charging infrastructure, as well as the backend software systems that support it. Seamless charging services, which link charging sessions to vehicles by correlating asynchronous event streams from charge point operators and vehicle systems, introduce a verification challenge that conventional testing approaches are not well suited to address. There is no precomputable ground truth against which each matching decision can be evaluated, and reproducing relevant adversarial conditions through physical testing is not feasible at scale.

This thesis presents the design, implementation, and evaluation of a simulation-based metamorphic testing framework that addresses these constraints. The framework combines discrete-event simulation, which generates realistic and adversarial event sequences in a reproducible execution environment, with metamorphic relations, which act as a substitute oracle by defining behavioral properties that must hold across related executions. Eight metamorphic relations that characterize correct vehicle-to-session matching behavior were derived through a four-step process consisting of source code analysis, brainstorming, peer review with the development team, and consolidation. The framework was evaluated against the Seamless Charging Service operated by WirelessCar AB across 18 scenarios.

The simulator generated 467 validation reports. Six of the eight metamorphic relations held across the dataset, while two, matching exclusivity and match quality monotonicity, revealed 43 violations across 11 reports. These violations were traceable to structural inconsistencies in the system's observed behavior, despite the system producing zero incorrect matches across 998 attempted matches. No confirmed false positives were observed across 397 control executions. The findings demonstrate that simulation-based metamorphic testing can detect meaningful behavioral deviations in event-driven cyber-physical systems without access to a traditional test oracle. At the same time, the strength of this conclusion is bounded by the conditions the framework was able to execute: the densest concurrency and event-ordering scenarios did not complete within the configured execution budget, leaving the system's behaviour under the highest-stress conditions unverified.

Keywords: metamorphic testing, discrete-event simulation, cyber-physical systems, verification, oracle problem, electric vehicle charging, event-driven systems.

Acknowledgements

We thank our academic supervisor, Linda Erlenhov, for her guidance and feedback throughout this thesis. Her questions and steady input helped us turn a broad idea into a focused piece of work, and we are grateful for her support along the way. We also thank our examiner, Gregory Gay, whose comments and advice pushed us to sharpen our reasoning and improved the final result.
We thank WirelessCar AB for the opportunity to carry out this thesis in an industrial setting. In particular, we thank the Seamless Charging development team for their time, technical insight, and honest feedback while we developed and refined the metamorphic relations. Without their input and assistance, starting this project would be close to impossible; therefore, we are beyond grateful for all the assistance we have been granted along the course of this project.

Oscar Cronvall & Linus Pettersson, Gothenburg, June 2026

Contents

List of Figures

List of Tables

1

Introduction

With the growing adoption of Electric Vehicles (EVs) as alternatives to conventional internal combustion vehicles, demand for easy, reliable charging infrastructure has grown substantially. This progress has contributed to the emergence of a rapidly expanding and technologically diverse charging infrastructure market, characterized by a broad range of Charge Point Operators (CPOs) and eMobility Service Providers (eMSPs). These actors include traditional fossil fuel companies, energy providers, and newly established companies focused specifically on EV charging services.

Alongside this expansion, the processes surrounding user authentication, authorization, and payment have caused considerable complexity to the public charging ecosystem. In practice, EV users are frequently required to register with multiple CPOs and eMSPs to access charging stations across different networks. Consequently, initiating a charging session at an unfamiliar charging station often requires several manual steps, such as connecting the vehicle, downloading a dedicated application, creating an account, and authorizing the charging session. This fragmented experience reduces usability and may discourage users from utilizing unfamiliar charging infrastructure.

To confront these challenges, WirelessCar AB has developed Seamless Charging, a service intended to simplify and automate the charging process. The service, referred to as Seamless Charging System (SCS), initiates charging sessions automatically when an EV is connected to a charging station. This is achieved by correlating telemetry data exchanged between vehicles and charging points, including positional information and charging-state data. Depending on the intended deployment architecture, SCS can either integrate into existing CPO systems by replacing the identification and authorization stages of the charging workflow, or operate as a unified business-to-consumer platform responsible for user authentication, payment handling, and session management across multiple charging networks.

However, systems such as SCS require extensive verification to ensure reliable operation under diverse, non-deterministic conditions. This thesis explores the feasibility of using simulation to generate synthetic event data in combination with Metamorphic Relations (MRs) to validate the behaviour and correctness of SCS.

1.1 Problem Description

The growing adoption of EVs has driven demand for reliable, interoperable public charging infrastructure. As a consequence, the software systems responsible for managing charging sessions have grown in complexity, operating within heterogeneous networks of CPOs, Original Equipment Manufacturers (OEMs), and backend services providers that communicate through independent asynchronous data streams [14]. In this environment, it becomes important to ensure that a charging session is correctly attributed to the right vehicle. A problem of session matching is both safety-critical from a billing perspective and technically difficult because to the non-deterministic nature of the event streams involved, even when scoped to single national markets.

From a software engineering perspective, this class of systems constitute a fundamental verification challenge. The SCS developed by WirelessCar AB illustrates the problem: it must correctly match a vehicle to a charging session by matching asynchronous Open Charge Point Protocol (OCPP) and Open Charge Point Interface (OCPI) events from CPOs with telemetry from OEM APIs, without any direct cable-level communication between the vehicle and the charger. The correctness of a match depends on the interaction of events whose arrival order, timing, and content vary non-deterministically in production. This means there is no pre-computable ground truth output for any given event sequence. The system faces what is formally known as the oracle problem [42, 6].

Compounding the oracle problem is the infeasibility of physical testing at scale. The SCS operates in a fragmented charging ecosystem that includes various implementations of the CPO network, varying network statuses, and concurrent sessions across multiple vehicles and chargers. Reproducing the full range of conditions the system encounters in production, particularly simultaneous session initiations, delayed event arrivals, and out-of-order message sequences, would require either prohibitively large physical test environments or a systematic simulation-based approach. Neither the oracle problem nor the physical testing constraints have been addressed in the existing literature for event-driven backend systems of this type. Metamorphic Testing (MT) has been established as a principled methodology for addressing the oracle problem [29, 36], and has recently been extended into the domain of Cyber-Physical System (CPS) with formal design assumptions [27]. Its application to event-driven backend systems, where correctness is defined relationally across asynchronous, non-deterministic event streams, remains unexplored.

1.2 Purpose of the Study

The purpose of this study is to design, implement and evaluate a simulation-based verification framework for SCS that addresses both the oracle problem and the physical testing constraint as described in Section 1.1. The primary artifact is a discrete-event simulator that generates realistic and adversarial event sequences, combined with a set of Metamorphic Relation (MR)s derived from direct inspection for the Software Under Test (SUT) that serve as a substitute oracle for assessing correctness. Together, these constitute a simulation-based metamorphic testing framework evaluated against SCS in an industrial context at WirelessCar AB.

With the above purpose the following research questions are to guide this study:

RQ1 How can discrete-event simulation be used to construct a systematic verification framework for an event-driven seamless EV charging system?

RQ2 What metamorphic relations characterize correct behaviour in a vehicle-to-session matching system, and how effectively do they detect faults under different and adversarial event conditions?

RQ3 To what extent does simulation-based metamorphic testing constitute a viable verification strategy for event-driven cyber-physical systems facing the oracle problem?

RQ1 addresses the design and implementation of the simulation framework as a Design Science Research (DSR) artifact, examining the architectural and methodological decisions required to make discrete-event simulation a practical verification environment for an industrial event-driven system. RQ2 addresses the identification and evaluation of MRs specific to vehicle-to-session matching, connecting the theoretical MT methodology to the concrete behavioural properties of the SUT. RQ3 addresses the generalizability of the joint approach, evaluating whether the method constitutes a practical strategy for the larger class of event-driven CPS systems that share the architectural properties identified in this study.

1.3 Significance of the Study

Apart from its immediate industrial application, this study tackles a methodological gap in the software testing literature. MT has been demonstrated as effective throughout scientific computing, web services, and most recently CPSs with physical invariants [36, 27]. However, no prior work has applied MT to event-driven backend systems where correct behaviour is defined by relational consistency across asynchronous event streams from independent, concurrently operating actors. This structural property characterizes not only the SCS but a larger class of systems in domains such as IoT and distributed transaction processing. By extending simulation-based MT to this class of systems, this thesis contributes a reusable methodology and an empirical case study that expand the known boundaries of MT applications.

1.4 Scope and Limitations

This study appraises the proposed framework within a single industrial case study at WirelessCar AB. The SCS is treated as an unmodified production system; no changes were made to the SUT during the course of this work. The simulation is constrained to the service operated by WirelessCar, meaning that faults originating from external actors such as malformed OCPP messages from CPOs or incorrect OEM telemetry are outside the scope of the verification framework. The observational evaluation of RQ3 is based on results from this single case study, and while the method is argued to be transferable to structurally similar systems, independent replication in other industrial scenarios is identified as future work.

2

Background

This chapter provides the background needed to understand the technologies and concepts discussed in this thesis. It first introduces the key stakeholders in the EV charging ecosystem, including the actors that provide charging infrastructure, vehicle data, and backend services. It then describes the architecture of EV charging systems, including charging hardware, connector standards, and communication protocols. Finally, it introduces different charging procedures, including conventional charging, Plug and Charge, and the service in focus for this thesis: SCS.

2.1 Stakeholders

Several stakeholder organisations are relevant to the development and verification of SCS. These stakeholders provide different parts of the charging ecosystem, ranging from physical charging infrastructure and backend services to vehicle telemetry and end-user services.

2.1.1 WirelessCar AB

WirelessCar AB is a Swedish connected-vehicle services company that develops and sells white-label software solutions for the automotive industry. That is, software produced by WirelessCar but rebranded and deployed under each vehicle manufacturer's own name rather than WirelessCar's. The company specialises in cloud-based API platforms that enable vehicle connectivity services such as Smart EV Routing, Digital Key Management, Call Center Services, Data Access Management, and related services [44]. In the context of this study, WirelessCar develops and operates SCS, a solution that aims to improve the convenience of EV charging by automating verification and session initialisation. This allows the user to plug the vehicle into a charger and have the charging session begin without manually initiating authentication through a separate application or token.

2.1.2 Charge Point Operators (CPO)

A CPO is an organisation responsible for operating electric vehicle charging infrastructure, including the Electric Vehicle Supply Equipment (EVSE) installed at charging locations[1]. In addition to operating physical EVSE, CPOs typically runs backend software that manages charging sessions and communication with other actors in the EV charging ecosystem.

[1]In the EV charging ecosystem, the term Charge Point Owner, also abbreviated as CPO, is often used to refer to the entity that owns the physical charging infrastructure. Since this study is concerned with the data supplied by chargers, the abbreviation CPO is used to refer to the operator rather than the owner.

For this reason, CPOs are important data providers for protocols such as OCPP and OCPI, which generate and manage data during charging sessions. Together with OEMs, CPOs provides the data required by the SCS to identify and manage charging sessions.

2.1.3 Original Equipment Manufacturers (OEM)

In this thesis, OEM refers specifically to vehicle manufacturers. OEMs are relevant because they provide vehicle-side telemetry, such as vehicle identity, position, and charging-related state information. Together with data from CPOs, this telemetry enables the SCS to identify which vehicle should be associated with a given charging session.

2.2 EV Charging Architecture

EV charging introduces system complexity throughout every stage of the charging process. The charging market is highly fragmented and involves various CPOs, OEMs, and eMSPs, requiring end users to interact with multiple vendors and service providers. This landscape requires solutions targeting the EV charging market to support high levels of interoperability to achieve wide market use.

2.2.1 AC and DC Charging

Both Alternating Current (AC) and Direct Current (DC) can be utilised during charging sessions; however, the two current types require different charging hardware and infrastructure. The current supplied from the electrical grid is AC. AC chargers deliver this current to the vehicle, where the onboard charger converts AC to DC before it is stored in the battery. DC chargers convert AC power from the grid inside the charger itself, allowing power to be delivered directly to the vehicle battery. Therefore, DC chargers generally allow greater power transfer compared to AC chargers, since they are not constrained by the size and heat dissipation of the onboard charger [11]. This necessitates different connector types, as the charging architecture and communication requirements of AC and DC charging differ [12].

2.2.2 Connector Standards

Connector standards vary based on manufacturer-specific designs and the requirements of different current types. While legislative efforts have been made to standardise this interface, as in the European Union where Type 2 (AC) and Combined Charging System 2 (CCS2) (AC and DC) are mandated, no global consensus is in place. Certain regions, such as the US, have historically been characterised by the presence of multiple competing connector standards, resulting in the need for adapters between plugs and connectors. In recent years, the US has moved toward the North American Charging Standard (NACS) as a common connector standard [39, 7, 35]. However, this low-level signalling is limited to conveying state transitions and the maximum current the charger can supply, encoded as a PWM duty

cycle. Richer interactions, such as battery parameter exchange, charging schedule negotiation, and authentication, require high-level communication protocols layered on top of the connector, such as ISO 15118 over HomePlug Green PHY power-line communication, or CAN-based protocols used in CHAdeMO and GB/T [17, 4].

2.2.3 Charge Point and Vehicle Communication

Compared to refuelling Internal Combustion Engine (ICE) vehicles, EV charging requires greater coordination between the vehicle and the charger, since the power that can be accepted by a vehicle varies between different EVs and changes during a charging session [17]. Failure to adapt the charging session accordingly could damage equipment or pose severe safety risks, such as thermal runaway [18]. Standards such as ISO 15118 [17] and communication frameworks such as OCPP [31] enable charger-vehicle and charger-backend communication to support controlled charging operations [18].

2.2.4 Communication Protocols (OCPP and OCPI)

To facilitate communication between chargers and CPO backend systems, OCPP is used as a protocol for managing charging sessions. It supports functions such as starting and stopping charging sessions, monitoring station health, and providing firmware updates [31]. To ensure interoperability between eMSPs and CPOs, OCPI is used. OCPI provides a unified interface for roaming, telemetry sharing, and transaction management [30]. The difference between OCPI and OCPP is summarised in Table 2.1.

Feature	**OCPP [31]**	**OCPI [30]**
Connection	Hardware ↔ CPO Software	CPO Software ↔ eMSP Software
Main goal	Management & control	Roaming & interoperability
Typical data	Voltages, errors, start/stop	Pricing, locations, tokens, CDRs

Table 2.1: Comparison of OCPP and OCPI protocols.

2.2.5 ISO 15118

ISO 15118 is an international standard, jointly developed by the International Organisation for Standardisation (ISO) and the International Electrotechnical Commission (IEC), that defines communication between electric vehicles, charging stations, and the electrical grid. The standard enables the exchange of information related to charging control, authentication, billing, and smart energy management, allowing charging and discharging operations to be optimised according to economic or energy-efficiency objectives. In addition, ISO 15118 enables Plug and Charge, which, similar in purpose to SCS, allows vehicles to be authenticated and charging sessions to be initiated automatically when the vehicle is connected to a charging station. This communication is typically performed through Power Line Communication (PLC) over the charging cable. However, deployment of Plug and Charge

requires both the vehicle and the charging station to support the necessary ISO 15118 hardware and software capabilities [17].

2.3 EV Charging Process

This section describes different technologies and methods related to EV charging processes, including how a charging session is initiated, maintained, and terminated. In particular, the conventional EV charging procedure, Plug and Charge, and the technology in focus for this thesis—SCS—are introduced.

2.3.1 Conventional EV Charging

Without a unified charging experience, an EV driver initiating a public charging session must navigate a fragmented and often unintuitive process. A 2025 report by the Fédération Internationale de l'Automobile (FIA) identifies convenience and reliability of public infrastructure as the highest priorities for EV drivers, while also highlighting fragmented payment solutions and a lack of pricing transparency as the most significant friction points in the current user journey [14].

This fragmentation is reflected directly in the charging procedure itself. The driver must first identify a charging station operated by a CPO whose service they have access to, which typically requires cross-referencing multiple apps or membership cards. Upon arrival, the driver physically connects the vehicle using a connector that may require an adapter depending on the station and vehicle combination. Authentication is then required before charging can begin. Depending on the CPO, this is handled either by scanning an Radio Frequency Identification (RFID) tag that must be pre-ordered and registered with that specific operator, or through a CPO-specific mobile application. This step alone introduces a significant friction point, as a driver without the correct RFID tag or app account for that particular CPO cannot initiate a session regardless of the physical connection being established [13].

Once authentication has been attempted, the driver depends on the communication between the vehicle, charger, and CPO backend functioning correctly. In the best case the session starts without issue. In the worst case the session fails silently or with a cryptic error, leaving the driver to troubleshoot at the charging station with limited feedback. Ending the session introduces further complexity, as different stations handle disconnection differently. Some require re-authentication to unlock the connector, while others have dedicated stop buttons or application-based stop flows. This fragmented experience stands in direct contrast to refuelling an internal combustion engine vehicle, where the process is standardised and operator-agnostic [14].

Beyond procedural friction, the FIA report identifies limited technical and operational interoperability across the charging ecosystem as a critical underlying barrier. This is particularly relevant to automated charging functionality, where uncertainty

in the digital communication between the vehicle, charger, and backend service provider can affect session initiation and user experience [14].

2.3.2 Plug and Charge

Plug and Charge is a standardised approach to EV charging that directly addresses the authentication friction described in Section 2.3.1. Rather than requiring the driver to present an RFID tag, mobile application or contactless payment, the vehicle and charger communicate directly upon physical connection, automatically handling authentication, session initiation, and billing without any driver intervention. The experience is reduced to a single action: plugging in [17].

The technical foundation of Plug and Charge is defined in ISO 15118, which specifies a TLS-secured communication layer between the vehicle and the charger. Each vehicle is issued a contract certificate by its OEM or eMSP, which is stored in the vehicle and presented to the charger automatically upon connection. The charger validates this certificate against a trusted authority, and if successful, the session begins. This certificate-based authentication removes the dependency on operator-specific credentials entirely.

However, the practical adoption of Plug and Charge remains inconsistent across the charging ecosystem. While ISO 15118 defines the protocol, its implementation requires alignment between the vehicle manufacturer, the CPO, and the eMSP — all of which must support the same certificate infrastructure. Where this alignment is absent, the Plug and Charge handshake fails and the driver is forced to fall back to conventional authentication methods [14].

2.3.3 Seamless Charging

Seamless Charging, as developed by WirelessCar, addresses the gap created by the absence of widespread adoption of Plug and Charge through a different approach. Rather than relying on a direct ISO 15118 certificate exchange between vehicle and charger, it correlates proxy data from OEM APIs and CPO-provided OCPP and OCPI events to achieve the same outcome — automatically matching a charging session to the correct vehicle and handling the associated billing. This makes Seamless Charging functional in environments where full ISO 15118 support is not yet in place, broadening the reach of Plug and Charge-like functionality across the fragmented charging landscape.

2.4 Software Testing

System-level software testing aims to verify that a complete system behaves correctly under conditions that represent those it will encounter in production. This is opposed to unit or integration testing, which validates isolated components or their immediate interactions. [33]

While unit and integration tests are valuable for catching implementation errors early, they operate on controlled inputs and cannot capture the emergent behaviour that arises when a full system processes real-world data streams. For a system such as the SCS, whose correctness depends on the interplay of asynchronous events from multiple external sources, system-level testing is both an essential verification layer and one of the most difficult forms of testing to execute effectively [33].

2.4.1 Testing of Event-Driven Software

Event-driven software is architecturally distinct from request-response systems in that its behaviour is entirely determined by the arrival, ordering, and content of incoming events rather than by synchronous function calls with predictable return values[28]. This architectural model introduces testing challenges that conventional example-based testing — in which the tester manually authors each input together with its expected output — is not designed to address, since the space of relevant event sequences is far too large to enumerate by hand. Techniques such as model-based testing[40], property-based testing [10, 2], and feedback-directed random generation [32] were developed in part to relieve exactly this burden: rather than requiring the tester to write individual input–output pairs, they generate inputs automatically and check them against a model, a general property, or an implicit reference oracle. Metamorphic testing, the approach adopted in this thesis, is itself a form of property-based testing in this sense. What distinguishes the SCS is therefore not input construction but the evaluation step: correctness is defined over an accumulated event history produced by independent external actors and is itself non-deterministic, so there is no model or oracle against which any individual output can be checked. This is the oracle problem, and it is the reason a relational technique is required rather than a purely generative one.

The first challenge is asynchrony. In the SCS, events originating from OCPP status notifications, OCPI session updates and OEM API callbacks arrive independently and are not guaranteed to appear in any fixed sequence. A charging session that produces events in one order during a manual test may produce them in a different order in production, depending on network conditions and CPO-specific implementation behaviour. A test suite that only validates fixed event sequences therefore provides limited coverage of the system's real operational envelope.

Closely related is the problem of non-determinism. Even when the physical charging process is identical, the event stream it produces can vary. Two sessions at the same charger with the same vehicle may yield different event timings, different field values in intermediate messages, or different numbers of status transitions depending on transient grid conditions or backend processing delays [26, 21]. This means there is often no single correct output that a test can assert against. The system must instead be validated on whether its behaviour remains logically consistent across varying inputs, which is precisely the challenge the oracle problem describes.

A further challenge is state explosion. An event-driven session accumulates state across its lifetime, and the correctness of any given event's processing depends on the

history of events that preceded it. As session length grows, the number of possible event orderings grows combinatorially, making exhaustive enumeration of test cases infeasible. Manual construction of test scenarios can only cover a small fraction of this space, and the scenarios most likely to expose matching failure are often the edge cases least likely to be anticipated by a test designer. [28]

Finally, there is a structural gap between unit-level test coverage and production behaviour. Individual components of the SUT's event parsers, session state machines, and matching logic may pass their respective unit tests while the integrated system fails on event sequences that no single component test ever exercised. This gap is not a failure of unit testing but a consequence of the combinatorial complexity that only emerges at the system level. Bridging this gap requires a testing approach that can generate realistic, varied, and high-volume event sequences automatically, which is the role the simulation framework developed in this thesis is designed to fill.

2.4.2 Metamorphic Testing

The challenges described in Section 2.4.1: Asynchrony, non-determinism, state explosion, and the gap between unit-level coverage and production behaviour collectively manifest as an instance of the oracle problem: the absence of a mechanism by which the correctness of a system's output can be determined for a given input. For event-driven systems like the SCS, this means that standard input-output testing cannot be applied, since there is no pre-computable expected output against which observed behaviour can be verified.

MT was developed specifically to address this class of problem [29]. Rather than asserting that a system produces a specific correct output for a given input, MT defines Metamorphic Relations (MR)—properties that must hold across related inputs and their corresponding outputs, regardless of the specific values involved. Formally, if a system f is executed on input x_1 to produce output y_1, and then executed on a transformed input x_2 to produce output y_2, an MR specifies the relationship that must hold between y_1 and y_2 given the known relationship between x_1 and x_2. A violation of this relationship constitutes evidence of a fault in f, even in the absence of a known correct output for either execution. For example a relationship could be defined as $sin(x) = sin(\pi - x)$.

This approach has two properties that make it particularly suited to the verification problem in this thesis. First, it is oracle-free in the classical sense: correctness is assessed relationally rather than absolutely, which remains tractable even when no ground truth is available [36]. Second, it is composable with simulation: by executing the SUT repeatedly across simulation-generated inputs and evaluating each pair of executions against a set of MRs, verification can be scaled across a large and varied input space without requiring manually specified expected outputs for each test case. The process by which MRs are identified for the SCS, and how they are applied within the simulation framework, is described in Sections 4.3 and 4.3.5, respectively.

3

Related Work

This chapter presents the foundational research which our work is built on. It aims to synthesise the core concepts underpinning both our methodology and the resulting artifacts. By providing the necessary theoretical context without delving into the exhaustive process of how these paths were selected we instead provide a baseline of terminology and their relationship to each other and our work.

3.1 Verification and Validation

The foundation of any robust testing strategy for complex systems lies in a structured Verification and Validation (V&V) process. As defined by Robert G. Sargent [41] and emphasised by Balci [5], V&V is not a single step but a continuous life cycle that spans the entire simulation study, from problem formulation through the presentation of results. This process ensures that a model or system is both internally consistent and externally accurate. Verification focuses on the transformation process, ensuring that the conceptual logic has been correctly translated into the SUT, in other words, building the model right. Validation, in contrast, ensures that the system's behaviour aligns with the intended requirements of the real-world domain, or building the right model [5].

In many event-driven and data-intensive systems, however, traditional validation becomes impractical because no explicit ground truth exists against which outputs can be compared. To address this limitation, MT is employed as an operational validation technique. By focusing on MRs rather than individual ground-truth results, MT enables systematic verification of system behaviour across transformed input sequences. This provides a scalable way to confirm that the system remains consistent even when the environment is non-deterministic or high-dimensional.

V&V also presupposes that the inputs reaching the SUT are well-formed in the first place. Input validation enforces the structural and semantic constraints on incoming data so that only valid input is accepted for further processing. It is essential to a large class of data-intensive systems and constitutes a distinct concern from the verification of system logic [25]. In the context of an event-driven backend such as the SCS, this distinction matters: input validation addresses the integrity of individual events at the ingestion layer, while MT addresses the relational consistency of system behaviour across sequences of valid events. The verification framework developed in this thesis assumes that input validation is handled separately at the system boundary, consistent with the scope established in Section 1.4, and focuses on the relational properties that MT is uniquely positioned to evaluate.

3.2 Oracle Problem

A fundamental challenge in software testing is the existence of a reliable test oracle — a mechanism by which the correctness of a system's output can be determined for a given input. As formally identified by Weyuker [42], many programs are inherently non-testable in the classical sense: even when a system produces an output, there is no tractable or affordable way to verify whether that output is correct. This is the oracle problem, and it arises whenever the expected output cannot be specified in advance, is too costly to compute independently, or depends on environmental conditions that are themselves non-deterministic.

Barr et al. [6] provide a comprehensive treatment of the oracle problem in software testing, identifying it as one of the most pervasive and under-addressed challenges in the field. They distinguish between cases where an oracle is entirely absent — such as in systems whose correct behaviour cannot be formally specified — and cases where an oracle exists in principle but is impractical to apply at scale. In both cases the consequence is the same: standard input-output testing cannot be used to validate the system with confidence.

In the context of SCS, the oracle problem manifests in a specific and concrete way. The matching logic that attributes a charging session to the correct vehicle processes asynchronous OCPP and OCPI events whose arrival order and timing vary non-deterministically. For any given sequence of events there is no pre-computable ground truth that can serve as an oracle, since the correct match outcome depends on the interplay of multiple data sources under conditions that cannot be fully controlled. As Chen et al. [29] establish, this is precisely the class of problem for which MT was developed — by defining MRs that must hold across transformed inputs rather than specifying absolute expected outputs, MT provides a principled way to validate system behaviour even in the absence of a traditional oracle.

3.3 Simulation as a Testing Methodology

Simulation is an established approach for studying the behaviour of complex systems that are difficult, costly, or impossible to test directly in their real-world environment. By constructing a model that replicates the essential characteristics of a target system, simulation allows controlled experimentation across a wide range of scenarios — including edge cases that would be rare or unsafe to provoke in production [41].

A particularly relevant class of simulation for software testing is Discrete-Event Simulation (DES), in which the system is modelled as a sequence of events occurring at specific points in time [20]. Between events, the system state remains unchanged — it only transitions when an event is processed. This maps naturally onto event-driven software architectures, where system behaviour is entirely determined by the arrival, ordering, and content of incoming events. In the context of SCS, each OCPP status notification, OCPI session update, or OEM API callback constitutes

a discrete event, and the correctness of a match depends on how the SCS processes these events relative to one another.

The validity of any simulation-based testing approach depends on the fidelity of the model — that is, how accurately the simulated events and their distributions reflect real-world conditions. This connects directly to the V&V framework described in Section 3.1: the simulation model itself must be validated against known system behaviour, by comparing model output against the real system under the same input conditions, before its outputs can be trusted as meaningful test evidence [5]. A simulation that fails to capture the timing characteristics or event variability of real charging sessions will produce results that do not transfer to production.

3.4 Stochastic Modelling

Real-world systems rarely behave deterministically [20]. In the context of EV charging infrastructure, both the demand placed on the electrical grid and the latency of network communication between system components vary in ways that cannot be predicted precisely in advance. Deterministic testing — where inputs are fixed and expected outputs are fully specified — cannot adequately capture this variability, and a system that passes only deterministic tests may still fail under the irregular conditions it encounters in production.

Stochastic modelling addresses this by representing uncertain or variable inputs as probability distributions rather than fixed values [15]. Instead of specifying that a network message arrives after exactly 200 milliseconds, a stochastic model might draw the arrival delay from a distribution that reflects the observed variance in real network conditions. Similarly, grid demand fluctuations can be modelled as random variables whose statistical properties are calibrated to real-world measurements. By sampling from these distributions across many simulation runs, a stochastic simulator can systematically explore the space of conditions the system may encounter rather than testing only a narrow set of hand-crafted scenarios.

In the context of this study, stochastic modelling serves two purposes. First, it generates realistic variation in the timing and ordering of OCPP and OCPI events, which is essential for testing whether the matching logic of the SCS remains correct when events do not arrive in an idealised sequence. Second, it enables the simulation to stress-test the system at the boundaries of its operational envelope, a recognised dynamic VV&T technique [5], for instance by generating latency spikes or concurrent session bursts that are statistically plausible but rare enough that they would be difficult to reproduce through manual testing alone.

3.5 Metamorphic Testing

Metamorphic Testing was originally introduced by Chen, Cheung and Yiu [9] as a technique for generating follow-up test cases for programs without a reliable oracle.

Rather than asserting that a program produces a specific output for a given input, MT specifies *relations* that should hold between the outputs of related executions, and uses violations of those relations as evidence of faults. In the two decades since, the method has matured into a general-purpose testing methodology with applications across scientific computing, machine learning, web services, and cyber-physical systems [29, 36].

As Chen et al. [29] formalise, MT serves as a dual-purpose methodology for both test case generation and test validation. It directly addresses the oracle problem described in Section 3.2 — a scenario where the expected output for a given input is unavailable or too costly to determine. Rather than relying on traditional, deterministic input-output mappings, MT utilises MRs. These relations describe how the output should change (or remain invariant) when the input is transformed. By executing the SUT multiple times across these transformed inputs, MT provides insights into system behaviour under varying circumstances. This approach is particularly suited for complex, non-deterministic systems such as event matching, where predicting a specific output in advance is often impossible. By executing the SUT multiple times across these transformed inputs, MT provides insights into system behaviour under varying circumstances [36].

3.5.1 Metamorphic Testing as an Oracle Solution

The maturity and breadth of MT as a methodology is well documented. Segura et al. [36] provide a comprehensive survey of MT applications, identifying the oracle problem as the unifying motivation across domains including scientific computing, security testing, web services, and autonomous systems. Their taxonomy distinguishes between MRs that specify invariant outputs, where a transformation of the input should leave the output unchanged, and MRs that specify relational outputs, where the output should change in a predictable, verifiable way. This distinction is central to how MRs are applied in practice and directly informs the MR derivation process described in Section 4.3.

Beyond its theoretical motivation, MT has been shown to be empirically effective at fault detection. Liu et al. [24] conducted a controlled study comparing MT against traditional testing on a set of subject programs and found that MT achieved fault detection rates comparable to oracle-based testing, despite operating without access to ground truth. This result is important because it establishes that the absence of an oracle does not inherently weaken MT's fault-detection capability, which is a central assumption underlying its use in this thesis.

3.5.2 MT for Distributed and Service-Oriented Systems

The applicability of MT to distributed, asynchronous, and externally-dependent services has also been demonstrated directly. Segura et al. [37] applied MT to RESTful web APIs operated by Spotify, Yelp, and YouTube, cataloguing eight abstract metamorphic relations and showing that they exposed real defects in production services. Earlier work by Chan et al. [8] established MT as viable for service-oriented

architectures, where loose coupling and asynchronous messaging make traditional oracle-based testing impractical. These results establish that MT can be applied to backend services consuming heterogeneous external data streams, and they inform the MR abstraction style adopted in Section 4.3. The Seamless Charging System shares these structural traits — backend, asynchronous, externally dependent — but, as Section 3.6 develops, differs in ways that motivate a distinct methodological treatment.

3.5.3 Metamorphic Relation Identification

A central challenge in MT is the identification of MRs that are both meaningful and practically applicable. Kanewala et al. [19] investigate this challenge in the context of scientific software, where the absence of a ground truth oracle closely mirrors the conditions under which the SCS must be tested. Their work demonstrates that MR identification is a non-trivial process requiring domain knowledge, and that even partially specified MRs can provide significant fault detection capability when applied systematically. Zhou et al. [45] further examine how MRs can serve dual purpose: not only as a test oracles but as tools for deepening understanding of system behaviour and exposing implicit assumptions embedded in the SUT. This dual role is particularly relevant in industrial contexts like this thesis, where the SUT is an existing production system whose full behavioural specification is not formally documented in a trivial mapping between inputs and outputs.

3.5.4 MT for CPS and Simulation-Based Verification

Beyond these foundational applications, MT has more recently been extended to CPS. Mandrioli et al. [27] propose an approach combining design-assumption-based MRs with genetic programming to automatically generate test cases for CPS subjects including drone controllers and robotic systems. Their work demonstrates that MT can be effective in CPS domains where physical laws and design contracts provide a basis for MR formulation. However, their approach presupposes the existence of formal design assumptions, mathematical or physical invariants that can be encoded as MRs automatically. This distinction motivates the manual, inspection-based MR derivation process adopted in this thesis, as described in section 4.3, and represents a meaningful difference in problem class between this work and that of Mandrioli et al.

Of particular relevance to this thesis is the explicit combination of simulation-based testing and metamorphic testing. Lindvall et al. [23] integrated model-based testing with metamorphic relations to verify autonomous systems in a simulated environment, establishing that simulation and MT can be composed within a single verification framework when neither a ground-truth oracle nor exhaustive physical testing is available. Their work provides direct methodological precedent for the approach taken in this thesis, though their subject systems are governed by physical dynamics rather than asynchronous event correlation. In an industrial case study closer in spirit to the present work, Ayerdi et al. [3] applied MT to an elevator control system at an industrial partner, addressing quality-of-service properties and timing

constraints. Their evaluation framework offers a structural template for reporting industrial MT applications and informs the evaluation design in Section 4.6.

3.6 Research Gaps

The preceding sections establish the theoretical and empirical foundations on which this thesis builds. Taken together, they reveal a set of interconnected gaps in the existing literature that motivate the research questions posed in Chapter 1.

3.6.1 Gaps in MT for Event-Driven Systems

The oracle problem, as established by Weyuker [42] and elaborated by Barr et al. [6], is a well-recognised challenge in software testing. While metamorphic testing has emerged as a principled methodology for addressing it [29], the domains in which MT has been systematically applied have historically centred on scientific computing, machine learning, and web services [36]. Recent work by Mandrioli et al. [27] demonstrates that MT can be extended to cyber-physical systems. However, their approach assumes the presence of formal design assumptions grounded in physical laws or mathematical contracts. Prior work has also applied MT to distributed backend services — Chan et al. [8] to service-oriented architectures, and Segura et al. [37] to RESTful web APIs at scale — establishing that MT is viable for backend systems consuming external data. However, these applications share a structural assumption that does not hold for the class of system examined in this thesis: correctness is evaluated relative to a request-response pairing in which each follow-up test case is anchored to an identifiable initiating request. The Seamless Charging System differs in three ways that place it outside this structural assumption. First, its correctness is defined across *concurrent, independent* event streams from OCPP, OCPI, and OEM APIs that are not paired by any shared request context. Second, there is no initiating request from which follow-up inputs can be systematically derived; the "input" to any matching decision is an accumulated event history whose composition is itself non-deterministic. Third, the actors producing these events operate independently and asynchronously, so metamorphic transformations must be defined over event sequences rather than over single API calls. To the authors' knowledge, no prior work has applied MT to systems exhibiting all three of these properties, and no established MR identification methodology targets them directly.

3.6.2 Gaps in Simulation-Based Metamorphic Testing

Simulation has been established as a viable environment for testing complex systems that are infeasible to exercise exhaustively in production [41]. Discrete-event simulation in particular maps naturally onto event-driven architectures, as described in Section 3.3. However, the combination of discrete-event simulation with metamorphic testing as a substitute oracle where simulation provides the test execution environment and MRs replace the absence of a ground truth has not been explored in the literature. Existing simulation-based testing approaches either assume the availability of a reference oracle or focus on coverage metrics rather than relational

correctness. The absence of this combined approach represents a methodological gap that is directly relevant to systems like the SCS, where neither a tractable oracle nor exhaustive physical testing is available.

3.6.3 Gaps in Verification of Seamless EV Charging Systems

Finally, at the domain level, no prior work addresses the systematic verification of seamless EV charging systems or similar session-attribution backend. The fragmented and asynchronous nature of the EV charging ecosystem, characterised by heterogeneous CPO implementations, variable network conditions, and concurrent session initiations across independent actors creates verification challenges that are specific to this domain but representative of a broader class of connected vehicle and IoT backend services. The absence of a published verification framework for this class of system constitutes a practical gap that complements the methodological gaps identified above.

3.6.4 Thesis Positioning and Research Contributions

This thesis addresses all three gaps through the design, implementation, and evaluation of a simulation based metamorphic testing framework for the SCS. RQ1 addresses the methodological gap in simulation-based testing by constructing a discrete-event verification framework. RQ2 addresses the MT application gap by deriving and evaluating MRs for an event-driven matching system without formal design assumptions. RQ3 addresses the generalisability gap by evaluating whether the combined approach constitutes a viable verification strategy for the broader class of event-driven CPS systems facing the oracle problem.

4

Methods

This chapter describes the methodology used to develop and evaluate the simulation-based framework for verification of the SCS. The approach combines discrete-event simulation with metamorphic test generation and evaluation to enable repeatable and deterministic assessment of system behaviour.

The simulation is used to model interactions and execute test scenarios [41], while metamorphic testing addresses the oracle problem by evaluating correctness in the absence of explicit expected outputs [6]. Together, these methods support systematic evaluation of system behaviour under the environmental uncertainty and dynamic conditions typical of CPSs [22].

4.1 Research Design and Method Approach

This study adopts a Design Science Research (DSR) methodology within a case study context at WirelessCar AB. DSR is well suited to research that produces and evaluates concrete artifacts in response to an identified problem in a real-world setting [16]. While Hevner et al. establish DSR's general principles in information systems research, Wieringa [43] develops an engineering-cycle formulation tailored to software engineering, consisting of problem investigation, treatment design, and treatment validation. This structure maps directly onto the design, implementation, and evaluation activities carried out in this thesis.

Rather than passively observing an existing system, DSR frames the research as the iterative construction and evaluation of artifacts that address a genuine organizational need. In this thesis, the primary artifact is a simulation-based verification framework designed to systematically test the SCS under conditions that are infeasible to reproduce through physical testing alone.

To answer RQ1, the study designed and implemented a discrete-event simulation framework that models asynchronous event streams from CPOs and OEMs. To answer RQ2, the study derived and implemented metamorphic relations that define the expected correctness properties of the matching process. To answer RQ3, the study analysed the results of these simulation-based metamorphic tests to assess the viability of the proposed approach as a verification strategy for event-driven cyber-physical systems affected by the oracle problem.

Figure 4.1 provides an overview of how the methodological activities in this thesis are organized within the iterative Design Science Research process.

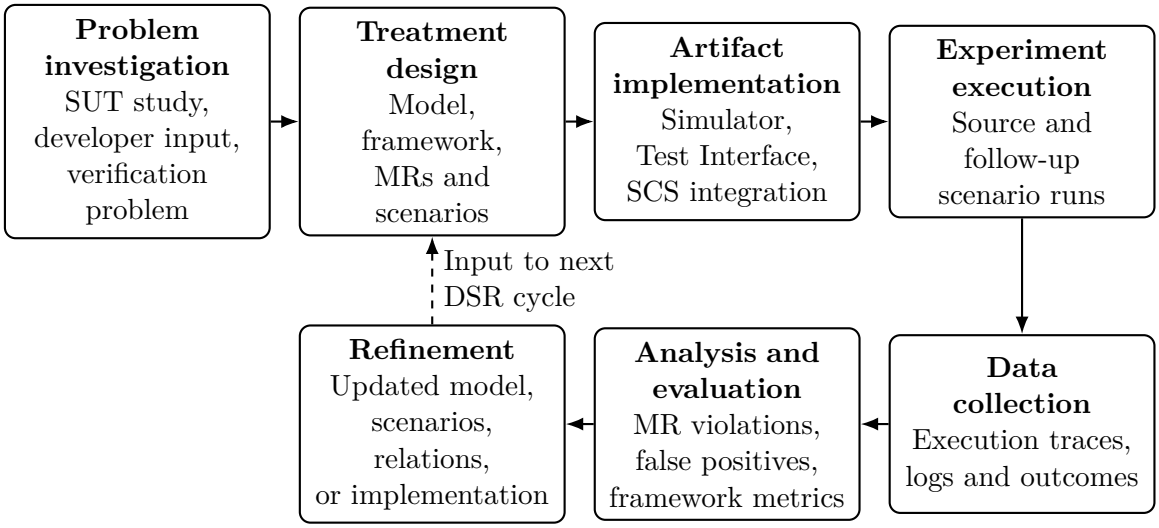


Figure 4.1: Overview of the iterative Design Science Research process used in this thesis.

4.1.1 Mapping to Research Questions

The metrics above address the research questions as follows. RQ1, which concerns the construction of the simulation framework, is addressed primarily through the framework performance metrics together with the validation process described in Section 4.2.3. RQ2, which concerns the effectiveness of the MRs at detecting faults, is addressed through the verification effectiveness indicators above — the per-relation observed-violation counts and coverage figures, evaluated against the false positive rate on control executions. The mutation testing pass that was originally planned to serve as the primary controlled answer to the RQ2 fault-detection sub-question was not feasible to produce within the project's time budget; its absence bounds the strength of the RQ2 claim and is reported as a scope limitation in Section 1.4 rather than as a metric in this section. RQ3, which concerns the viability of simulation-based metamorphic testing as a verification strategy for the broader class of event-driven CPSs, is addressed through a combination of the empirical results from RQ1 and RQ2 and a qualitative discussion in Chapter 6 of the methodological assumptions that would transfer to structurally similar systems. As noted in Section 1.4, the empirical basis for RQ3 is a single industrial case study, and independent replication in other contexts remains future work.

4.1.2 Problem Formulation

The motivation behind this research is the absence of a systematic verification method for the SCS. Current solutions revolve around a handful of vehicles and cannot reliably or deterministically test complex interactions and scenarios. Thus, this thesis addresses the need for a solution that scales beyond a handful of vehicles and reduces the work hours required for each manual real-world testing session.

As described in Chapter 1, the SCS operates as part of a CPS, where software services interact with vehicles, charging infrastructure, and external data sources. This creates a non-deterministic, event-driven environment in which the correctness of vehicle-to-charging-session matching depends on asynchronous data streams from multiple sources. This makes conventional test approaches inadequate: there is no tractable oracle for expected outputs, the input space is combinatorially large, and the conditions most likely to expose matching failures are precisely those most difficult to reproduce manually.

The artifact developed in this work addresses this problem by combining discrete-event simulation with metamorphic testing. These two methodological choices are justified in the following sections, with simulation and metamorphic testing discussed in Sections 4.2 and 4.3, respectively.

4.1.3 Methodological Choices

The choice of simulation and metamorphic testing follows directly from the theoretical analysis in Chapter 3. The oracle problem, as established by Weyuker [42] and elaborated by Barr et al. [6], makes standard input-output testing insufficient for the SCS. The correct match outcome for a given event sequence cannot be precomputed, because it depends on the non-deterministic interplay of asynchronous OCPP, OCPI, and OEM events whose arrival order and timing vary in production.

Metamorphic testing was selected because it was developed to address this class of problem [29]. Rather than asserting absolute expected outputs, it evaluates relational consistency across transformed inputs, which remains tractable even in the absence of a traditional oracle.

Simulation was selected as the execution environment because the SCS operates as part of a CPS in a distributed, stochastic environment where exhaustive physical testing is infeasible [41]. As described in Section 3.3, discrete-event simulation maps naturally onto event-driven architectures, allowing a broad space of realistic and adversarial event sequences to be explored in a controlled and reproducible setting.

Together, the two methods are complementary by design: simulation addresses the generation problem of producing realistic, varied, and high-volume test inputs, while metamorphic testing addresses the oracle problem of assessing correctness in the absence of ground truth. Neither method alone would be sufficient; their combination constitutes the core methodological contribution of this work.

4.1.4 Empirical Evaluation

The empirical dimension of the study is realized through the systematic execution of simulation scenarios and the quantitative measurement of results against formally defined metamorphic relations. Each simulation run produces observable session outcomes that are evaluated for relational consistency rather than against absolute expected values. This allows correctness to be assessed at scale across a varied

input space without requiring manually specified ground truth for each test case. The metrics used for this evaluation are defined in Section 4.6.

4.1.5 Validity Considerations

The validity of this study is discussed along the dimensions established by Runeson and Höst [34] for case study research in software engineering: internal validity, external validity, construct validity, and reliability.

To support internal validity, three design decisions were made deliberately. First, the simulation is executed under reproducible and configurable conditions, so that differences in outcomes between scenarios can be attributed to variation in input parameters rather than uncontrolled environmental factors. Second, the metamorphic relations were derived through a structured process involving direct inspection of the SUT and peer review by the development team, grounding the relations in the system's actual semantics rather than in purely theoretical assumptions, as described in Section 4.3. Third, the simulation model itself was subject to iterative validation through review during and after development, following the V&V principles established by Sargent [41] and applied throughout Section 4.2.3.

External validity is more limited by design. This study evaluates a specific industrial system in a specific deployment context, and its findings are not intended to generalize directly to other event-matching systems without adaptation. However, the methodology, which combines simulation-based test execution with metamorphic relations as a substitute oracle, is presented as a transferable approach for verifying systems that share the same structural challenges: non-deterministic outputs, event-driven architecture, and an absence of tractable ground truth.

4.2 Simulation

Since the SUT is a CPS operating in a highly complex environment, real-world testing is challenging. Achieving deterministic behaviour requires large-scale coordination and control over variables that may be impractical to manage in real-world settings [22]. Simulation provides an alternative by offering a controlled and testable environment that supports effective V&V [41].

Simulation does not inherently require a visual component, but visualization and animation can support verification and validation by making system behaviour easier to inspect [41]. This is particularly important in contexts where determining correctness is non-trivial due to the absence of a reliable test oracle [6]. The purpose of simulation in this thesis is therefore twofold: to act as a test scenario runner and to serve as a visual V&V tool.

4.2.1 Conceptual Model

Sargent [41] defines a conceptual model as the logical representation of the problem entity. A conceptual model is developed during the design phase of a simulation study and updated as new requirements and assumptions emerge. It is used to ensure that the assumptions, theories, and limitations of the model are valid with respect to the intended purpose of the simulation.

In this thesis, the conceptual model defines the entities, events, state transitions, and assumptions needed to represent the SCS matching process at the level relevant for verification. The model has been derived for this purpose and has been validated as outlined in Section 4.2.3.

4.2.1.1 Entities and Relationships

The conceptual model comprises vehicles, chargers, SCS, charging sessions, and communication events. Vehicles and chargers interact through temporally ordered events that, depending on the state of the system, result in the establishment of a charging session.

4.2.1.2 Event-Driven Behavioural Model

The SCS determines matching outcomes based on independent, asynchronous connection events from OEMs and CPOs. The matching algorithm considers vehicle and charger events, including their temporal and geographical relationships, when determining matches. This matching process is modelled in a simplified and abstracted manner in Equation 4.1[1].

$$SCS_{\text{match}} = \begin{cases} \arg\min_{m \in M} C\Big(E^{\text{connected}}_{\text{vehicle}}(t_v, p_v), E^{\text{connected}}_{\text{charger}}(t_c, p_c)\Big) & \text{if } C < \theta \\ \varnothing & \text{otherwise} \end{cases} \tag{4.1}$$

where:

$$\begin{aligned} t &= \text{time at which the event was sent} \\ p &= \text{position associated with the event} \\ E^{\text{connected}}_{\text{vehicle}}(t_v, p_v) &= \text{vehicle connected event sent at time } t_v \text{ and position } p_v \\ E^{\text{connected}}_{\text{charger}}(t_c, p_c) &= \text{charger connected event sent at time } t_c \text{ and position } p_c \\ C(\cdot) &= \text{matching cost function} \\ M &= \text{set of candidate matches} \\ \theta &= \text{matching acceptance threshold} \end{aligned}$$

[1]The presented model is intentionally abstracted, as the implementation details of the production matching algorithm are proprietary to WirelessCar AB.

The equation describes the abstracted matching algorithm used by the matching service in the SCS. Candidate vehicle–charger pairs are evaluated based on temporal and geographical proximity. A match is determined by selecting the pair with the lowest matching cost that also satisfies the acceptance threshold θ. The candidate set M consists of all eligible vehicle–charger event pairs available to the matching service at a given point in time.

This means that the simulation must not artificially couple the position of an entity with the time at which an event is sent. Doing so would compromise the validity of the model, since the SCS receives vehicle and charger events as independent asynchronous inputs. Consequently, the behavioural model is based on independent vehicle and charger events that become coupled only when matching operations are performed by the SCS. Vehicles, chargers, and their associated events are therefore modelled as independent entities within the simulation.

4.2.1.3 State Representation and Transition

To capture relevant real-world behaviour, the simulator maintains operational states for vehicles and chargers, including idle, matching, and charging. The idle state represents entities that are not currently participating in a matching process or an active charging session. The matching state represents entities that have sent a connection event but have not yet received a matching response. The charging state represents an active charging session.

The transition from idle to matching can only be initiated by vehicle or charger events. In contrast, the transition from matching to charging can only occur through a charging-session event generated by the SCS. This means that vehicles and chargers can initiate the conditions for a charging session, while the actual establishment of the session is determined by the SCS. The only allowed transition from charging is back to idle, and this transition may be initiated by vehicles, chargers, or the SCS. As shown in Figure 4.2, charging sessions are initiated by vehicle or charger events, while the transition to the charging state is determined by the SCS.

Current State	Next State	Trigger Source
Idle	Matching	Vehicle or Charger Events
Matching	Charging/Idle	SCS Event
Charging	Idle	Vehicle, Charger, or SCS Events

Figure 4.2: Allowed state transitions in the SCS behavioural model.

4.2.1.4 Conceptual Assumptions and Scope

The model is abstracted to operate at the input boundary of the SCS. It generates synthetic events and the required contextual fields as they would be received by the SCS, rather than modelling the exact real-world conditions from which the data originate.

Physical actions, such as a user connecting the charging cable, are therefore not modelled directly. Instead, their resulting system-level events, such as charger-connected events, are simulated and forwarded to the SCS.

Consequently, the geographical position associated with a vehicle event does not necessarily represent the true physical location of the vehicle, but rather the location information available to the SCS at the time of processing. The model therefore does not aim to distinguish between physically correct and incorrect vehicle positions; it targets the position information received and processed by the SCS, rather than the vehicle's physical ground-truth location.

The purpose of the simulation is not to reproduce the complete real-world charging ecosystem or all underlying physical conditions. Instead, the goal is to simulate the event and data representations observed by the SCS, since these are the inputs that influence SCS behaviour and can be directly affected by WirelessCar AB.

4.2.2 Computerized Model

Sargent [41] defines the computerized model as the implementation of the conceptual model in executable form. It consists of the logical structure and behaviour of the conceptual model translated into software. In the context of this thesis, the computerized model supports verification of the SCS through executable scenarios that are evaluated against metamorphic relations, as described in Section 4.3.

4.2.2.1 System Overview

The simulation consists of three primary components: the Simulator, the Test Interface, and the SCS. The Simulator orchestrates and executes scenarios, manages user input and configuration, and visualizes results. The Test Interface acts as an intermediary layer that standardizes communication and augments requests with the fields and artifacts required by the SCS, thereby improving Separation of Concerns (SoC). The SCS acts as the SUT; it processes incoming events and requests from the simulation via the Test Interface and determines matching outcomes.

During execution, the simulation generates events based on predefined scenario definitions. These events are propagated to the SCS through the Test Interface, which translates simulator-level events into SCS-compatible requests and adds the required fields. This triggers internal components of the SCS, such as the matching service, which determines the outcome of charging requests. The sequence of interaction is shown in Figure 4.3, which provides a high-level illustration of the interaction between the Simulator, Test Interface, and SCS during scenario execution.

The responsibilities of the Simulator, Test Interface, and SCS can be summarized as follows:

Simulator

- Executing simulation scenarios and advancing time
- Simulating vehicle and charger behaviour
- Maintaining entity state, including identifiers and positions
- Propagating events to the SCS via the Test Interface

Test Interface

- Translating simulation events into SCS-compatible requests
- Standardizing communication between the simulation and the SCS
- Decoupling simulation logic from SCS implementation

SCS

- Processing incoming requests from the Test Interface
- Performing matching between vehicles and charging sessions
- Managing the charging-session lifecycle

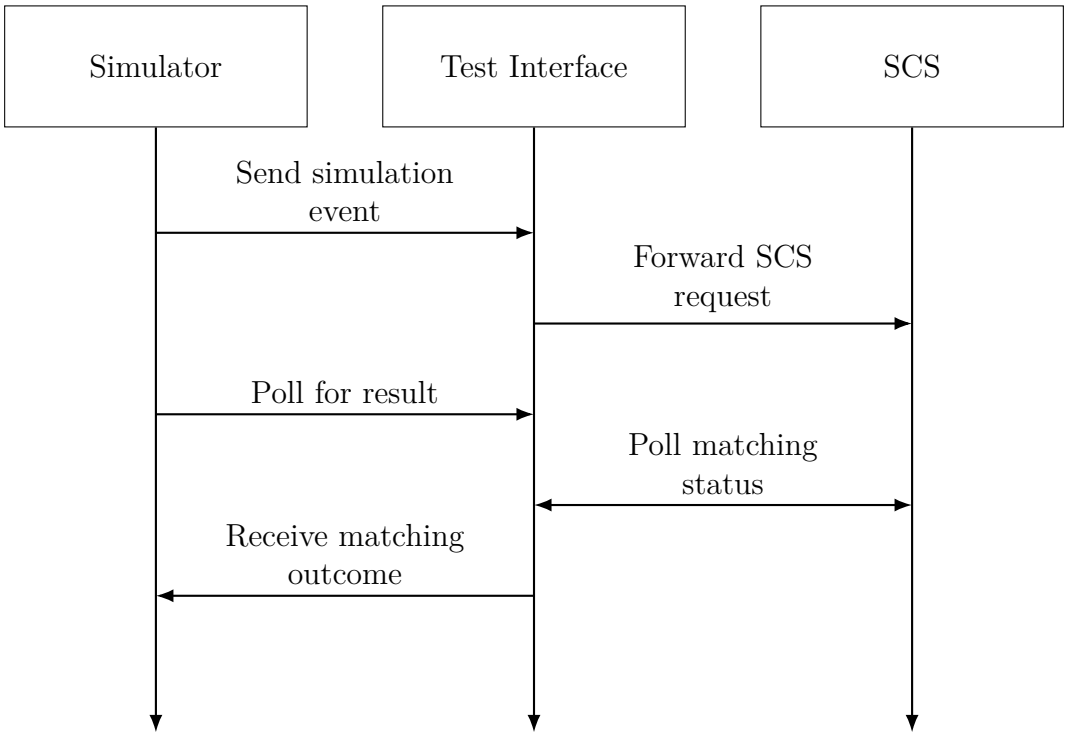


Figure 4.3: High-level simulation architecture illustrating the interaction between the Simulator, Test Interface, and Seamless Charging System (SCS) during scenario execution.

4.2.2.2 Simulator

The primary role of the Simulator is to generate synthetic OCPP and OCPI events, as well as synthetic vehicle telemetry that would, in production environments, originate from CPOs and OEMs. To achieve this, the Simulator tracks the states of chargers and vehicles, their geographical positions, and their identifiers. This makes it possible to simulate relevant vehicle and charger behaviour while maintaining traceability throughout a scenario.

The simulator controller acts as the internal coordination layer between the user interface and the simulator execution logic. It handles starting, pausing, restarting, and modifying test scenario runs, as well as visualization settings such as map controls and overlays. The controller does not communicate directly with the SCS. Instead, all communication passes through the Test Interface, which forms the external boundary between the simulation and the SUT.

4.2.2.3 Test Interface

The main purpose of the Test Interface is to reduce coupling between the Simulator and the SCS, thereby improving SoC. This separation is necessary because the SCS requires implementation-specific information, such as registered users, vehicle associations, and payment details. Although these aspects are not inherently part of the matching simulation, they are required by the production system in order for matching to be performed.

During a charging session, vehicles, chargers, and the SCS exchange requests and responses. In this implementation, parts of this interaction are offloaded to a mock server that simulates responses to SCS requests. In production environments, telemetry from CPOs is communicated through OCPI and OCPP. However, embedding protocol-specific system logic directly in the Simulator would violate SoC. Therefore, conversion between the simulator schema and SCS-compatible request formats is handled by the Test Interface.

4.2.2.4 Seamless Charging System (SCS)

The SCS represents the SUT. It was not modified during development and is therefore equivalent to the production system within the scope of this thesis. In the simulation, the SCS operates as the matching service, producing matching outcomes after receiving synthetic vehicle and charger requests from the Simulator through the Test Interface.

4.2.2.5 State Representation and Transitions

The state transitions follow the rules established in Figure 4.2 and are triggered either by synthetic events created by the Simulator or by responses from the SCS. For example, a vehicle connection event generated by the Simulator may move the vehicle and/or charger from an idle state to a matching state, while a successful matching outcome generated by the SCS may result in a transition to the charging

state. These transitions are recorded together with relevant identifiers, timestamps, and log entries to support traceability.

The state model is important for verification and validation because the final outcome of a scenario is not sufficient on its own. The sequence of intermediate states must also be inspectable in order to determine whether the system behaviour is consistent with the intended charging process.

4.2.2.6 Execution Model

Building on the state transition model, the simulator executes scenarios using a scenario-driven discrete-event execution model. Simulation scenarios are defined through structured scenario descriptions that specify entity behaviour, event timing, and interaction conditions. During execution, vehicles progress through predefined coordinates, referred to as Way Points. Movement between Way Points is performed at a fixed configurable speed along the shortest linear path between coordinates.

Way Points can be configured to trigger independent connection events. These specialized Way Points are referred to as Action Points and are introduced to decouple movement progression from interaction logic, thereby simplifying scenario construction.

An Action Point can, for example, transition vehicle and/or charger states from idle to matching by sending vehicle-connected and/or charger-connected events to the SCS. Depending on the matching outcome, the entities may subsequently transition either to charging or back to idle. After a charging session is completed, or unless otherwise specified, the vehicle resumes its route progression and proceeds to its next Way Point or Action Point. Events use the entity states at the time of transmission, meaning that event positions and timestamps correspond to the simulated position and time at which the event is sent.

In complex scenarios, vehicle arrival times are difficult to predict due to dependencies on previous interactions, such as charging sessions or matching delays. By triggering events upon arrival at predefined positions, the simulation removes ambiguity related to arrival timing. Since real-world charging occurs while vehicles are stationary, associating event generation with predefined positions provides both an intuitive and realistic representation of the charging process within the simulation.

Examples of the Action Point model are shown in Figures 4.4–4.6. Figure 4.5 illustrates a scenario containing a single Action Point and two generated events. Figure 4.7 shows how multiple independent vehicles can trigger asynchronous event streams, while Figure 4.8 illustrates how Way Points and Action Points are positioned conceptually within the simulation environment.

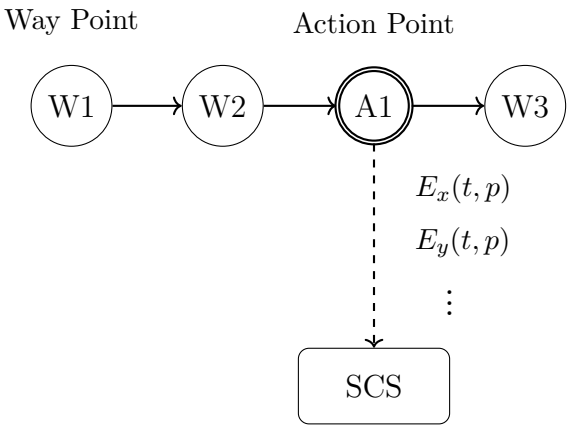


Figure 4.4: Scenario-driven execution where vehicles traverse predefined Way Points. W_n represents a normal Way Point, while A_n represents an Action Point — a specialized Way Point that triggers asynchronous events toward the SCS. $E(t,p)$ denotes time- and position-dependent events generated during scenario execution.

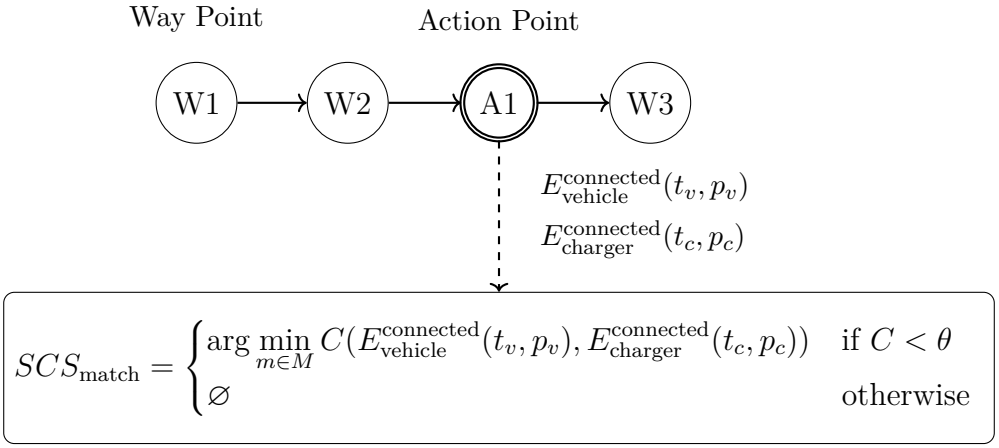


Figure 4.5: Example scenario containing one vehicle connection event and one charger connection event. The events are propagated to the SCS, which determines matching outcomes according to the model defined in Equation 4.1.

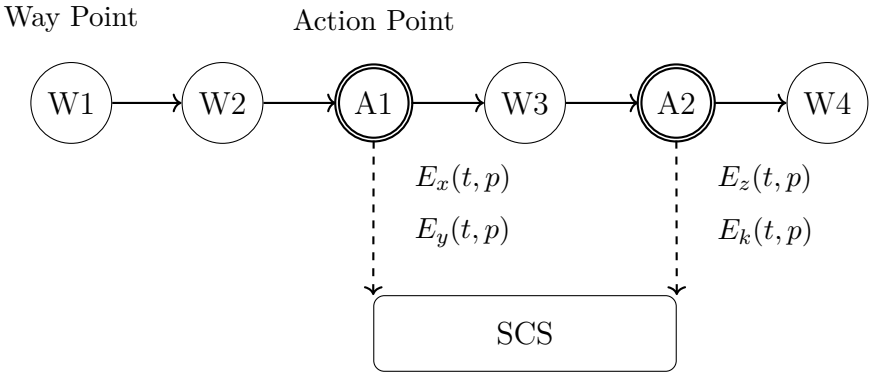


Figure 4.6: Multiple Action Points can be added to a scenario, illustrating that the scenario structure is modular and can be extended or varied by changing the Way Points, Action Points, and associated event definitions.

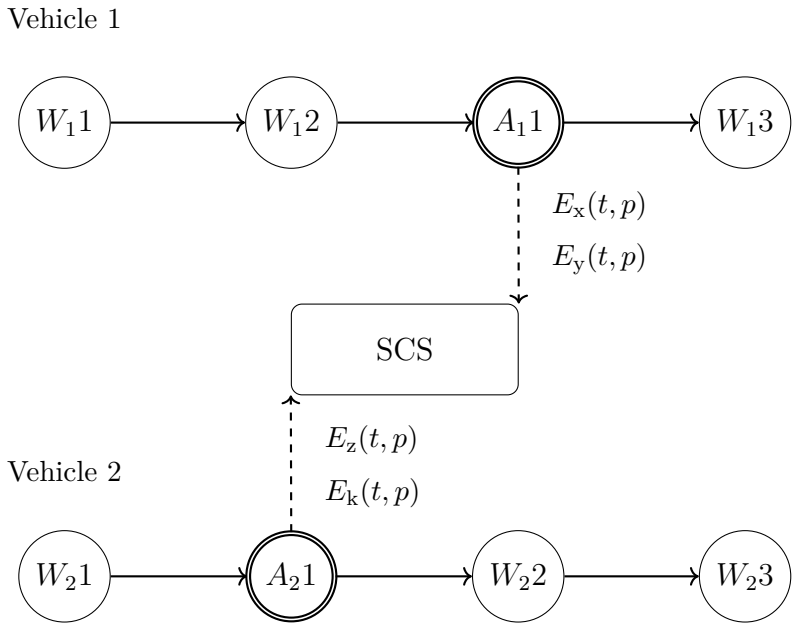


Figure 4.7: Less abstract scenario-driven discrete-event execution model showing how multiple independent vehicles traverse Way Point sequences and trigger asynchronous events at Action Points. The generated event streams are propagated to the SCS, which determines charging-session matches according to Equation 4.1.

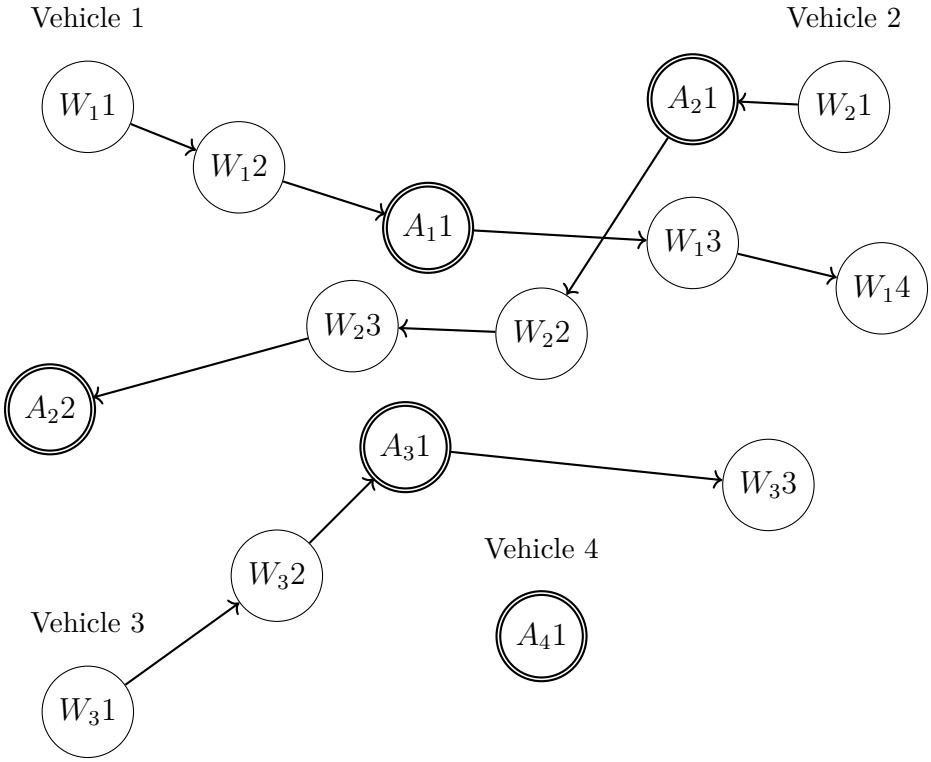


Figure 4.8: Example of a waypoint network where multiple vehicles traverse independent but spatially intertwined paths. The spatial placement of waypoints and action points represents their conceptual geographical positions within the simulation environment.

4.2.2.7 State Ownership and Traceability

State ownership in the system is divided between the experimental control provided by the Simulator and the system behaviour managed by the SCS. The Simulator maintains ownership of the simulated environmental state, including entity positions, identifiers, and simulation time, while the SCS maintains its internal business state related to matching and charging-session processing. The simulation therefore controls only scenario inputs and environmental conditions, whereas the SCS remains responsible for producing behavioural outcomes.

To support verification and validation, the simulation records structured execution traces of events, requests, responses, and state transitions, enabling traceability, behavioural analysis, failure localization, and scenario comparison.

4.2.3 Model Validation and Verification

According to Sargent [41], a simulation model should be developed for a specific purpose, and simulation can support V&V when the model is appropriately designed and implemented. However, if the simulation model itself is incorrect, it may produce misleading system behaviour and unreliable results. The simulation therefore requires validation and verification throughout its development.

Sargent [41] further notes that when a simulation model is developed by a small team, users and domain experts can be closely involved during development. In this thesis, validation of both the conceptual model and the computerized model was supported through interviews and review sessions with the development team behind the SUT and related test software. This validation was performed iteratively throughout the development of the simulation.

4.2.4 Simulation Interface

Figure 4.9 shows a sample of the simulation interface used for configuration and monitoring of test scenarios. The map view (1) shows the simulation environment, including vehicles, chargers, and their geographical positions. Simulation scenarios can be configured in the control panel (2). SCS responses and other relevant events are captured in the simulation logs (3), including examples such as connection and charging events. Additionally, scenarios can be forked through the UI by dragging and dropping vehicles and chargers, as well as by modifying Action Point variables. The vehicle and charger state panels (4–5) provide runtime information on event timelines, battery level, and charging status. Icons representing the different vehicle and charger states are shown in Figure 4.10.

The simulator can also be run without the graphical interface using a smaller Terminal User Interface (TUI). The TUI provides less runtime observability than the GUI, but it is useful for systematic execution of predetermined scenarios, for example in a CI/CD pipeline.

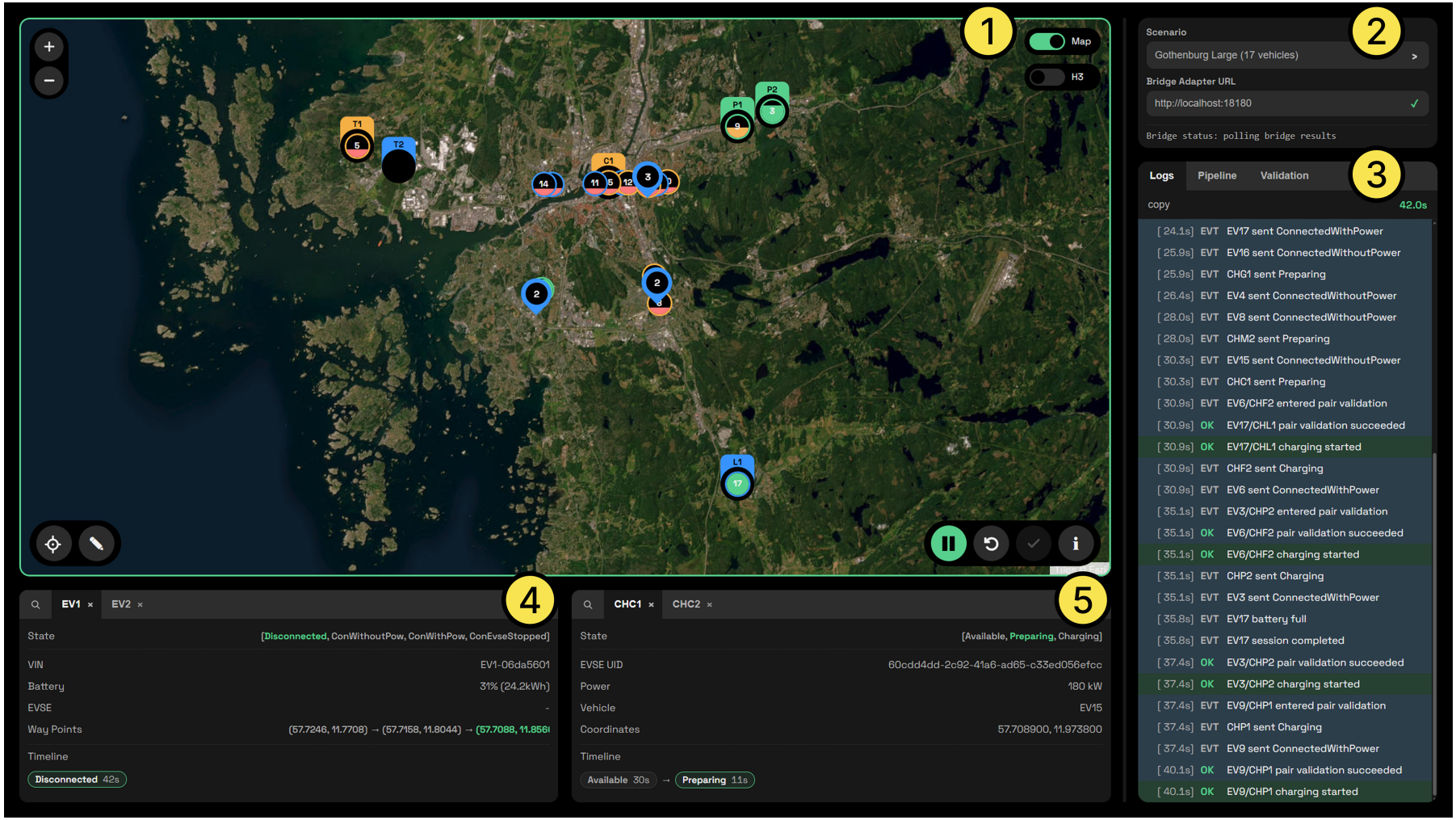


Figure 4.9: Graphical user interface used to configure, execute and monitor Seamless Charging test scenarios.

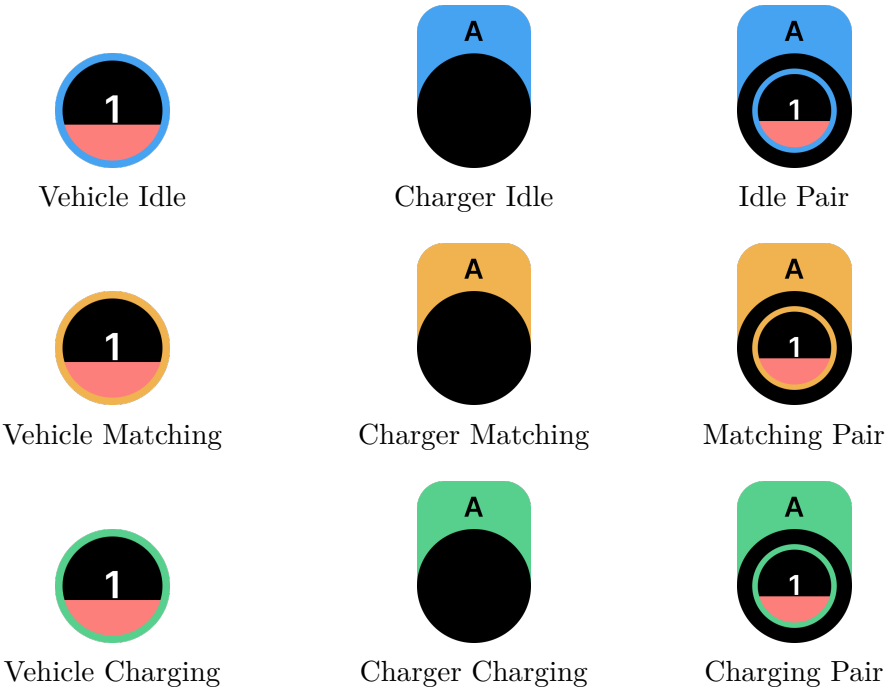


Figure 4.10: Icons representing vehicle (named 1), charger (named A), and combined states across different simulation states.

4.2.5 Support for Metamorphic Testing

The simulation environment supports metamorphic testing by enabling deterministic execution of source scenarios and transformed scenarios under controlled conditions. This is important because the correctness of SCS matching behaviour is not always easily determined using a traditional test oracle. Instead of relying only on fixed expected outputs, related scenario executions can be compared according to predefined metamorphic relations.

A source scenario defines an initial configuration, event sequence, and expected behavioural properties. A transformed scenario modifies selected aspects of the source scenario while preserving or changing specific properties according to the metamorphic relation being tested. Because the Simulator controls the initial state, event ordering, and simulation time, differences between related executions can be attributed to the intended transformation rather than uncontrolled environmental variation.

The Test Interface is important in this process because it ensures that conceptual scenario transformations are translated consistently into SCS-compatible requests. This helps separate potential failures in the SCS from failures caused by inconsistent request construction. The resulting execution traces, state transitions, and SCS responses provide the basis for evaluating whether the metamorphic relation holds.

In this way, the simulation acts as the execution environment for metamorphic testing. It provides repeatable scenario execution, observable state progression,

and traceable system responses, which together make it possible to evaluate SCS behaviour in situations where a complete test oracle is unavailable.

4.3 Metamorphic Test Generation

Metamorphic testing requires the identification of MRs—properties of the SUT that must hold across related inputs and outputs, regardless of the specific values involved. Since no automated oracle exists for our SUT, MRs serve as a tractable substitute, allowing correctness to be assessed through relational consistency rather than absolute expected outputs.

The generation of MRs for this thesis followed a manual, inspection-based process grounded in direct study of the SUT and collaboration with its developers. This choice reflects the consensus in the MT literature that MR identification is fundamentally a creative task requiring domain knowledge and system understanding [38, 29]. While automated MR generation techniques exist, they are effective primarily in mathematical and scientific software where input-output relationships can be expressed as formal invariants [1]. For event-driven systems like the SCS, where correctness depends on relational consistency across asynchronous, nondeterministic event streams, no comparable automation exists, and manual identification grounded in domain expertise remains the standard approach [19, 3]. The process followed four steps, described below.

4.3.1 Understanding the SUT

The first step involved a thorough reading of the SUT source code and documentation in order to develop a working understanding of its internal logic, architectural decisions, and inherent trade-offs. This was necessary to reason about which behavioural properties were both meaningful to test and plausible to specify as formal relations. Without this foundation, brainstormed relations risk being either trivially satisfied or disconnected from the system's actual semantics. This grounding step is consistent with established MT practice, where MR identification is recognized as requiring deep knowledge of both the application domain and the system's specification [38, 19]. Zhou et al. [45] further argue that the process of deriving MRs through SUT study serves a dual purpose: it produces test artifacts and surfaces implicit assumptions embedded in the system, which is particularly valuable when working with a production SUT whose full behavioural specification is not formally documented.

4.3.2 Brainstorming Candidate Relations

With a sufficient understanding of the SUT established, a broad set of candidate MRs was generated through an input-driven brainstorming strategy. Following the approach described by Segura et al. [38], candidate relations were derived by considering how systematic transformations of the SUT's input events—such as reordering,

duplication, insertion, temporal shifts, and substitution of identifiers—could be expected to affect its matching outcomes.

This step was conducted collaboratively by both authors. First, each author independently reviewed the relevant SUT behaviours and noted potential metamorphic relations based on the identified input transformations. The independently generated candidates were then compared in joint sessions, where overlapping relations were merged, unclear relations were clarified, and disagreements were resolved through discussion with reference to the expected SUT behaviour.

The goal at this stage was breadth rather than precision, capturing as many potentially relevant behavioural properties as possible without premature filtering. Prior work suggests that MR sets are more effective at fault detection when they exercise diverse input parameters and constraints [38], which motivated the deliberately wide scope of this step. Relations were recorded informally, prioritizing coverage of different aspects of SUT behaviour.

4.3.3 Peer Review and Validation

The roughly 25 brainstormed relations were shared with two software engineers on the development team, who provided written feedback. This step served a dual purpose. It acted as a domain-level validation that the identified relations were relevant and grounded in the actual behaviour of the system, and it informed the subsequent consolidation by surfacing redundancies, gaps, and relative priorities that were not apparent from a purely analytical perspective.

Practitioner involvement in MR validation is an established pattern in industrial MT case studies. Ayerdi et al. [3] derived MRs for an elevator dispatch system at Orona in direct collaboration with the development team, and their experience informed how the MR set was consolidated and prioritized for empirical evaluation. More broadly, the MT literature recognizes that domain expertise is essential not only for generating MRs but for selecting which candidates are worth retaining [36, 45]. Involving the engineers actively developing the SUT therefore strengthens both the credibility and the operational relevance of the final relation set.

4.3.4 Consolidation into a Final Test Set

Drawing on both the initial brainstorm and the feedback received, the candidate relations were reduced to a manageable final set. This consolidation involved merging relations that captured the same underlying property in different forms, and discarding relations deemed redundant, overly narrow, or impractical to evaluate. The resulting set represents a compact but representative collection of behavioural invariants against which the SUT can be systematically verified.

4.3.5 Application for Verification

The finalized metamorphic tests are used as the basis for simulation-based verification. The simulator is executed repeatedly with scenario variants, and each run is evaluated against the metamorphic relations to determine whether the SUT exhibits the expected relational behaviour. This approach allows verification to scale across a large and varied input space without requiring manually specified expected outputs for each test case or scenario.

4.4 Scenario Generation

While the MRs define the behavioural properties the SUT must satisfy, they must be exercised across a range of realistic and varied conditions in order to support meaningful verification. This is achieved through a set of simulation scenarios, each representing a distinct deployment context for the system. The scenarios were designed to cover both the relations identified in the preceding step and the edge cases surfaced during the initial code analysis.

4.4.1 Definition of Base Scenarios

The base scenarios were manually defined based on real-world use cases, with deliberate variation in environmental complexity. Each scenario represents a plausible deployment environment for the SUT, ranging from relatively straightforward settings such as a supermarket parking lot with predictable traffic patterns, to more demanding environments such as a multi-level parking facility in a dense urban area. This spectrum of difficulty ensures that the SUT is evaluated not only under nominal conditions but also in environments where the demands on the system are substantially higher.

4.4.2 Base Scenario Overview

The base scenarios were designed to represent a range of operating conditions relevant to the SCS. For clarity, they are grouped into logical categories according to the primary verification challenge they address.

Single-Vehicle Baseline. This category consists of the *single-1-vehicle* scenario. It contains a single vehicle and a single charger, producing an unambiguous charging session. It serves as a baseline reference for validating nominal matching behaviour and correct session lifecycle progression.

Multi-Vehicle Matching Scenarios. This category consists of the *city-2-vehicles*, *metro-3-vehicles*, *three-hex*, *same-hex*, *same-hex-10-pairs*, *same-hex-10-staggered*, and *gothenburg-large-16-vehicles* scenarios. These scenarios introduce multiple vehicles and chargers operating concurrently and evaluate the SCS under increasing levels of competition between candidate matches. By varying the number of entities, their geographical distribution, and event timing, these scenarios evaluate matching determinism, exclusivity, match quality, and robustness under increasing concurrency.

Ambiguity Scenarios. This category consists of the *adjacent-hex*, *adjacent-hex-crossed-fail*, *adjacent-hex-crossed-timeout*, *adjacent-hex-single-crossed*, and *shared-charger-2-vehicles* scenarios. These scenarios create situations where multiple candidate matches appear plausible, increasing the risk of crossed, conflicting, or otherwise incorrect associations. Their purpose is to evaluate spatial-temporal soundness, eligibility filtering, and the ability of the SCS to correctly distinguish between competing candidates.

Negative Matching Scenario. This category consists of the *far-apart-no-match* scenario. Vehicles and chargers are placed beyond the expected matching threshold. The correct behaviour is the absence of a match, making this scenario useful for evaluating rejection behaviour and false positives.

Lifecycle Scenario. This category consists of the *single-return-charge* scenario. A vehicle completes a charging session and subsequently initiates another session. This scenario evaluates session lifecycle management and state progression across multiple charging events.

Large-Scale Stress Scenario. This category consists of the *stress-grid-50-vehicles* scenario. It is intended to expose scalability and concurrency-related issues by increasing the number of simultaneously active vehicles and chargers beyond what is practical to evaluate through manual testing.

Metamorphic Validation Scenarios. This category consists of the *mr2-isolated-stability* and *mr6-ordered-bridge-evidence* scenarios. These scenarios were constructed specifically to evaluate individual metamorphic properties. The former introduces additional non-competitive entities to test match stability under benign environmental changes, while the latter applies event-ordering transformations to evaluate robustness against reordered and duplicated event streams.

Together, these scenarios provide coverage of nominal and adversarial operating conditions, including ambiguity, concurrency, lifecycle progression, stability, rejection behaviour, and event-ordering effects. This ensures that metamorphic relations are evaluated across a diverse set of conditions representative of those encountered by the SCS in production.

4.4.3 Coverage of Metamorphic Relations and Edge Cases

Together, the base scenarios and their generated variants are designed to provide coverage across two dimensions. First, each MR should be exercisable within at least one scenario, ensuring that the relational properties of the SUT are tested under realistic conditions. Second, the scenarios incorporate edge cases identified during the code analysis phase, targeting boundary conditions and implementation trade-offs that may not arise naturally during typical usage or manual charging sessions.

4.5 Experiment Execution Procedure

This section describes how the components introduced in Sections 4.3–4.4 are combined into a repeatable experimental procedure. Each experiment instance consists of a source run and one or more follow-up runs executed against the SUT, whose outcomes are compared against the MRs defined in Section 4.3.

4.5.1 Source and Follow-Up Run Construction

Metamorphic testing requires each test case to consist of at least two related executions of the SUT: a *source run*, which establishes a baseline outcome for a given scenario, and one or more *follow-up runs*, which execute the same scenario under an input transformation specified by an MR [29]. For each scenario generated through the process described in Section 4.4, the simulator first executes the unmodified scenario to produce the source outcome. The corresponding input transformation is then applied, for example by reordering events, inserting additional events, perturbing timing, or substituting identifiers, depending on which MR is under evaluation. The transformed scenario is then executed as the follow-up run.

4.5.2 Execution Flow of a Single Experiment

A single experiment instance proceeds through four stages. First, the scenario definition is loaded into the Simulator, which initialises the state of all vehicles and chargers and establishes the initial event schedule. Second, the Simulator advances through the scenario, propagating events to the SCS via the Test Interface as described in Section 4.2. Third, the SCS processes the events and produces matching outcomes, which are captured by the Simulator along with any intermediate state transitions relevant to the MRs under evaluation. Fourth, the captured outcomes are paired with their source or follow-up counterpart and passed to the evaluation procedure described in Section 4.6, which determines whether the expected relational property holds.

4.5.3 Reproducibility and Stochastic Control

Because the scenarios generated in Section 4.4 incorporate stochastic variation in event timing and ordering, individual runs are not deterministic by default. To support reproducibility, the random seed governing each simulation run is recorded alongside the scenario definition and experiment outcome, so that any run can be re-executed exactly. For source and follow-up pairs within a single metamorphic test, the same seed is used across both runs except where the MR itself specifies a transformation of the stochastic inputs. This ensures that any observed difference between source and follow-up outcomes is attributable to the MR transformation rather than to uncontrolled variance.

4.5.4 Scale of Execution

Each MR is evaluated across all scenarios in which it is applicable, and each scenario is executed with multiple stochastic variations to reduce the influence of any single event ordering on the results. The total number of experiment instances is therefore the product of the number of applicable MRs, the number of base scenarios, and the number of stochastic variations per scenario.

4.6 Evaluation Metrics

This section defines the metrics used to evaluate the simulation-based metamorphic testing framework against the research questions stated in Section 1.2. The metrics are organised around two levels: verification effectiveness, which assesses whether the approach detects faults in the SUT, and framework performance, which characterises the operational properties of the simulation environment itself.

4.6.1 Verification Effectiveness

The primary evaluation target is whether the MRs defined in Section 4.3, when applied through the simulation framework, surface behavioural deviations in the SCS. The evaluation was originally planned to use mutation testing as the primary controlled fault baseline, following prior industrial MT case studies [3]. In practice, however, producing and analysing enough mutants to constitute meaningful evidence proved far more time-intensive than anticipated. Covering the core scenarios at a comparable volume would have required at least doubling the available development time. This made a mutation pass infeasible within the project's time budget.

The strategy was therefore pivoted to an observational design based on repeated, seeded execution of the scenarios in Section 4.4. Under this design, each relation is exercised across many runs of the scenarios that target it. The random seed is recorded so that source and follow-up runs differ only in the transformation the relation specifies. The resulting violations and false positives are then counted against the SCS. This pivot is reported as a scope limitation in Section 1.4 and revisited in Chapter 6.

Three observational indicators are recovered from this design. The first, the *observed-violation count per relation*, is the number of validation reports in which an MR fails on the unmodified SCS, broken down by relation and by the scenario class that produced it. This indicator carries the fault-surfacing claim, since each violation is a concrete behavioural deviation rather than a synthetic fault detection. The second, the *per-relation coverage figure*, is the number of completed validation reports against which each MR was actually evaluated, derived from the scenarios it is associated with. It is reported alongside the violation count so that the absence of violations on a relation can be read in the context of how much evidence that relation was exposed to. A relation with zero violations across a hundred reports and one with zero violations across two reports support meaningfully different claims, and this metric makes that distinction explicit. The third, the *false positive rate*,

is the proportion of MR evaluations that flag a violation on control executions of the unmodified SCS under conditions where the relation is expected to hold. A non-zero false positive rate would indicate that the relations themselves produce spurious failures, which would invalidate the observed-violation findings.

Read together, these three indicators stand in for the mutation score and per-mutant detection rate that the planned mutation pass would have produced. They measure what the framework actually surfaces and how exposed each relation was to evidence, rather than what fraction of a seeded fault population is caught. The implication of this trade-off for the strength of the RQ2 claim is treated in Section 1.4 and Chapter 6.

4.6.2 Framework Performance

In addition to verification effectiveness, the operational characteristics of the simulation framework are measured to assess its practicality as a verification environment. Three indicators are recorded. The *execution time per experiment instance* captures the end-to-end duration of a single source–follow-up run pair, from scenario loading to outcome capture, and is used to assess whether the framework scales to the volume of experiments required for meaningful verification coverage. The *reproducibility rate* is the proportion of paired re-executions that produce identical event sequences and state transitions when re-run with the same recorded seed and scenario definition; a reproducibility rate below 100 % would indicate uncontrolled non-determinism in the framework itself, which would need to be investigated before the verification results can be trusted. The *completion rate* is the proportion of scheduled runs that produce a validation report rather than reaching the configured per-run timeout without completing; this is recorded because it is a known property of the execution environment (Section 5.2.2) rather than of the SCS, and it directly bounds the operational envelope the framework is able to exercise.

4.6.3 Use of Artificial Intelligence Tools

AI tools were used in two distinct capacities. First, for the software development of the simulator and its graphical user interface, the authors used Claude (Anthropic), Codex, and ChatGPT (OpenAI) as coding and exploratory support throughout all stages of the thesis work. These tools served as assistants for tasks ranging from initial scaffolding of boilerplate code and digesting unfamiliar APIs, to continuous feature implementation and debugging. All AI-suggested code used for the results was reviewed and functionally tested by the authors to ensure behavioural correctness. The overall architecture, design decisions, and metamorphic relations underlying the simulator remain the authors' own work

Second, for the written report, ChatGPT, Overleaf AI Assist and Grammarly were used for language refinement, including grammar correction, spelling, and stylistic improvements to text first drafted by the authors. AI tools were also used as a sounding board during the consolidation and refinement of the authors' own ideas, including discussions of candidate formulations, structure, and presentation. This

support was used to clarify and organise material, not to delegate the generation of research ideas, the formulation of metamorphic relations, the design of experiments, the analysis of results, or the conclusions of the thesis. All final research decisions, interpretations, and conclusions were made by the authors.

5

Results

This chapter presents the findings of the simulation-based metamorphic testing framework developed in Chapter 4, applied to the Seamless Charging System (SCS) as a representative event-driven cyber-physical system (CPS). The results are organized around the three research questions stated in Section 1.2: 1. The construction of the verification framework (RQ1). 2. The effectiveness of the metamorphic relations (RQ2), 3. the viability of the combined approach for the broader CPS class (RQ3). Findings are reported factually here, with interpretation deferred to Chapter 6.

5.1 Overview

The evaluation consists of simulation scenarios designed to exercise the metamorphic relations (MRs) of the matching system. Each simulator scenario targets at least one MR but usually multiple by varying a specific aspect of the input, such as vehicle density, measured charger proximity, physical ambiguity, or event ordering, and checking whether the matching system's behaviour remains consistent with the corresponding property. Rather than treating scenarios as isolated cases, they are grouped by the property they stress. The properties covered are:

- **Determinism (MR1):** identical inputs must yield identical matching outcomes. A violation indicates non-deterministic behaviour in a system that is required to attribute charging sessions reproducibly for billing.
- **Non-competitive match stability (MR2):** when a new vehicle or charger is introduced that is not a viable candidate for an existing match, the existing match must remain unchanged. A violation indicates that the matching engine is sensitive to inputs that should be inert.
- **Matching exclusivity (MR3):** a single charger must not be assigned to more than one vehicle during overlapping windows, and vice versa. A violation indicates that the system can produce conflicting billing attributions.
- **Spatial and temporal soundness (MR4):** matches must respect the physical and temporal constraints of the underlying scenario (e.g. a vehicle far from a charger cannot match it). A violation indicates that the matching cost function does not enforce its own thresholds.
- **Eligibility filtering (MR5):** entities outside the matching criteria must be rejected rather than matched. A violation indicates that ineligible candidates leak into the matching set.
- **Event idempotency and ordering (MR6):** duplicated or reordered events

that represent the same underlying state must not change the matching outcome. A violation indicates fragility to the asynchronous, out-of-order event streams the SCS receives in production.

- **Match quality monotonicity (MR7):** A validation with a higher score should not yield the match to a validation with a weaker score. A violation indicates an evidence-propagation gap within the matching pipeline.

- **Session lifecycle and observability (MR8):** every session must traverse the state transitions defined in the behavioural model (Figure 4.2), and its full trace must be inspectable — meaning the ordered sequence of states the session passed through, together with the events that triggered each transition, is recorded in the execution trace and can be reconstructed after the run. A violation indicates that intermediate states are lost, skipped, or never recorded, leaving the session's history unverifiable.

Each property is therefore both a behavioural invariant of the SCS and a verification criterion against which the framework can produce evidence.

The dataset analysed in this chapter consists of 467 complete validation reports across 18 scenarios. A small subset of these reports (the 20 runs of `mr2-isolated-stability`) was collected later in a targeted top-up to ensure MR2 had at least one dedicated scenario; it is treated as part of the same dataset throughout this chapter rather than as a separate experiment. A dedicated MR6 ordering scenario (`mr6-ordered-bridge-evidence`) was also attempted, but each of its scheduled runs reached the configured 180 s simulated timeout before producing a validation report; this scenario is retained in the scheduling tables below as an incomplete coverage attempt rather than excluded.

A *validation report* is the output of one simulation run that ran end-to-end and was checked against its associated MRs. A second figure, the number of *scheduled runs*, also appears in the tables below. Scheduled runs include simulation runs that the framework attempted to execute but did not finish, most commonly because the run exceeded the 180 s simulated timeout under dense conditions on the development hardware used to execute the experiments. The gap between scheduled and completed runs is therefore primarily a property of the execution environment rather than of the SCS itself, and is revisited as a limitation in Section 5.2.2 and Chapter 6. All MR-violation counts in this chapter are derived from validation reports, not from scheduled runs.

Some scenarios are intentionally constructed so that the expected outcome is the absence of a match (for example, the `far-apart-no-match` scenario, where vehicles and chargers are placed beyond the spatial matching threshold). In these scenarios, a report containing zero successful matches is the correct outcome, not a failure. The MR pipeline distinguishes between MR violations and zero-match scenarios accordingly, and only the former are counted as violations in the tables below.

Scenario categories and their targeted MRs are summarised in Table 5.1. The two

categories that test event ordering and idempotency (the staggered dense category and the dedicated MR6 ordering category) produced zero validation reports between them, and the implications of this gap are discussed in Section 5.2.2.

Table 5.1: Base case scenario categories and their targeted metamorphic relations.

Category	**# Scenarios**	**Reports**	**Primary MRs**
Baseline (single pair)	1	100	MR4, MR8
Dense / multi-pair / co-located	7	123	MR1, MR3, MR7
Staggered dense (ordering stress)	1	0	MR1, MR3, MR6, MR7
Ambiguity (adjacent / crossed)	5	179	MR3, MR4, MR5
Negative (no-match cases)	1	4	MR4, MR5
Lifecycle (repeated sessions)	1	41	MR8
Stability (isolated multi-pair)	1	20	MR2, MR4, MR8
Ordering / idempotency (dedicated)	1	0	MR6, MR1, MR3, MR7

Across all scenarios, the system was evaluated based on its ability to both correctly produce valid matches and correctly reject invalid or ambiguous candidates. A per-scenario breakdown of scheduled versus completed executions and the MRs each scenario targets is given in Table 5.2. This breakdown is the source of the scheduled and completed counts cited throughout the RQ1 and RQ2 analyses below.

Table 5.2: Scenario execution counts used for MR association.

Scenario	Scheduled	Reports	Primary MRs
adjacent-hex	57	16	MR3, MR4, MR5
adjacent-hex-crossed-fail	70	41	MR3, MR4, MR5
adjacent-hex-crossed-timeout	73	25	MR3, MR4, MR5
adjacent-hex-single-crossed	64	41	MR3, MR4, MR5
city-2-vehicles	55	55	MR1, MR3, MR7
far-apart-no-match	100	4	MR4, MR5
gothenburg-large-16-vehicles	55	7	MR1, MR3, MR7
metro-3-vehicles	55	16	MR1, MR3, MR7
same-hex	54	24	MR1, MR3, MR7
same-hex-10-pairs	55	0	MR1, MR3, MR7
same-hex-10-staggered	56	0	MR1, MR3, MR6, MR7
shared-charger-2-vehicles	56	56	MR3, MR4, MR5
single-1-vehicle	100	100	MR4, MR8
single-return-charge	100	41	MR8
stress-grid-50-vehicles	55	1	MR1, MR3, MR7
three-hex	55	20	MR1, MR3, MR7
mr2-isolated-stability	20	20	MR2, MR4, MR8
mr6-ordered-bridge-evidence	20	0	MR6, MR1, MR3, MR7

5.2 RQ1: Constructing the Simulation-Based Verification Framework

RQ1 asks how discrete-event simulation can be used to construct a systematic verification framework for an event-driven seamless EV charging system. The evidence relevant to this question is operational rather than relational: it concerns whether the framework, once built, exhibits the properties required of a verification environment for a CPS. Three indicators are reported, namely reproducibility, coverage, and aggregate execution performance.

5.2.1 Framework Reproducibility

In the 467 validation reports, when performing repeated executions of the same scenario with the same random seed generated identical event sequences and state transitions. No instances of uncontrolled non-determinism were observed within the simulation layer itself. The one execution-level exception is the dedicated MR6 ordering scenario, which reached its configured timeout in all 20 scheduled runs;

this is a coverage limitation rather than a determinism failure, and is treated in Section 5.2.2.

5.2.2 Framework Coverage

The framework executed scenarios spanning the behavioral categories of Table 5.1, from a single-vehicle baseline to a 50-vehicle stress configuration and as already mentioned with varying conditional changes. Of the 1,100 scheduled runs (Table 5.2), 467 (42.5%) produced validation reports. The remaining 633 scheduled runs did not complete, and the asymmetry between scheduled and completed runs is large enough that it warrants direct treatment rather than being left implicit.

The incomplete runs are concentrated in the densest scenarios: `same-hex-10-pairs`, `same-hex-10-staggered`, and `mr6-ordered-bridge-evidence` produced zero validation reports, and `stress-grid-50-vehicles` produced only one out of 55 scheduled. By contrast, every smaller scenario (single-vehicle baseline, two- and three-vehicle metro scenarios, shared-charger scenarios) completed all or nearly all of its scheduled runs. Two factors are involved. First, each simulated run is bounded by a 180 s simulated-time timeout configured in the framework; when the number of concurrent vehicles and chargers grows, the event processing required to advance simulated time slows correspondingly, and runs reach the timeout before reaching the matching outcomes the report is meant to capture. Second, the experiments reported here were executed on a laptop rather than a dedicated test cluster, which places a ceiling on the level of concurrency the framework can drive before the host itself becomes the bottleneck. The gap between scheduled and completed runs is therefore primarily a property of the execution environment used to run the experiments, not a property of the SCS under test. This observation is carried forward to Chapter 6, since it directly bounds the strength of the RQ3 viability claim.

5.2.3 Framework Performance

The simulation framework demonstrated the deterministic, reproducible, and observable execution properties identified in Chapter 4 as prerequisites for CPS verification. The headline figures over the full dataset of validation reports are reported in Table 5.3.

Table 5.3: Dataset summary across all validation reports.

Metric	Value
Validation reports	467
Reports passing all MRs	456
Reports failing at least one MR	11
Pass rate	97.64%
Total MR violations recorded	43

The simulator and framework were capable of executing the MR test suite under controlled conditions and produced traceable execution evidence for each completed

run. However, as established in Section 5.2.2, the inability to complete the densest ordering and stress scenarios on the available execution hardware indicates that the framework's coverage of high-density temporal stress remains an open operational limitation. This limitation is carried forward to the discussion of RQ3 in Chapter 6.

5.3 RQ2: Metamorphic Relation Effectiveness

RQ2 asks what metamorphic relations characterize correct behaviour in a vehicle-to-session matching system, and how effectively they detect faults under varied and adversarial event conditions. Two complementary sources of evidence were planned: observed violations on the unmodified SCS, which surface deviations from the relational properties during normal execution, and mutation-based fault detection, which would provide a controlled fault baseline. The mutation-based component was not feasible to produce at the scale required to constitute meaningful evidence within the resources available to this study, and is reported as a scope limitation in Section 5.3.5. The fault-detection evidence reported in this section is therefore drawn from observed violations on the unmodified SCS under varied and adversarial event conditions.

5.3.1 Per-Relation Validation Outcomes

Each of the eight MRs was evaluated against the validation reports of the scenarios in which it was applicable. The number of reports each MR was evaluated against, derived from the scenarios it is associated with, is given in Table 5.4. This table reports the primary scenario-design associations; the MR harness may still evaluate additional relations on each completed artifact, and that broader per-MR validation result is reported in Table 5.5.

Table 5.4: MR test counts from scenario associations.

MR	Associated scenarios	Scheduled	Reports
MR1	9	460	123
MR2	1	20	20
MR3	14	780	246
MR4	8	540	303
MR5	6	420	183
MR6	1	20	0
MR7	9	460	123
MR8	3	220	161

The per-MR validation summary is shown in Table 5.5. MR2 evidence comes primarily from the dedicated isolated-stability scenario, while MR6 was evaluated as an oracle on every completed artifact even though the dedicated MR6 ordering scenario did not itself complete and is therefore not counted as targeted MR6 evidence.

Six of the eight MRs were preserved across all evaluated scenarios with no observed

Table 5.5: Metamorphic relation validation summary.

MR	Property	Reports	Violations
MR1	Matching determinism	467	0
MR2	Match stability under non-competitive changes	265	0
MR3	Matching exclusivity	467	39
MR4	Spatial & temporal soundness	463	0
MR5	Eligibility filtering	463	0
MR6	Event idempotency & ordering	467	0
MR7	Match quality monotonicity	442	4
MR8	Session lifecycle & observability	463	0

violations. MR2 was confirmed in a targeted non-competitive multi-pair scenario. Meaning the scenario is built on a one-to-one relationship between the number of matching vehicles and matching chargers where no vehicle or charger share a matching room. MR6 had no observed oracle violations across the reports on which it was evaluated, but the dedicated MR6 ordering scenario remains inconclusive because all 20 scheduled runs timed out before producing validation reports. The remaining two relations, MR3 (matching exclusivity) and MR7 (match quality monotonicity), exhibited violations concentrated in two specific scenarios; these are characterized in Section 5.3.3.

5.3.2 Representative Scenario Outcomes

To illustrate the relationship between scenario design and MR outcome, Table 5.6 reports representative evidence from selected scenarios, highlighting system behaviours without reproducing all execution data.

Table 5.6: Representative scenario results.

Scenario	Purpose	Expected	Observed	MRs
Single pair	Baseline correctness	1 match / run	100 / 100 runs	MR4, MR8
Far-apart	Spatial rejection	0 matches / run	0 / 4 runs	MR4, MR5
Crossed fail	Ambiguity rejection	0 verified / run	82 / 41 runs	MR3, MR4, MR5
Isolated stability	Non-competitive stability	3 matches / run	60 / 20 runs	MR2, MR4, MR8
Staggered dense	Ordering robustness	N matches	Not evaluated (0 reports)	MR1, MR6
Ordered bridge	Event ordering / idempotency	4 matches / run	Timed out (0 reports)	MR6, MR1, MR3, MR7

These scenarios were selected as representative cases exposing distinct risks, including false positives, duplicate assignment, temporal disordering, ambiguous matching,

and non-competitive match stability. The isolated stability scenario completed all 20 of its scheduled runs and produced 60 successful session matches, confirming MR2 behaviour on this evidence. The staggered dense scenarios (`same-hex-10-pairs` and `same-hex-10-staggered`) did not yield validation reports, and the dedicated MR6 ordered-bridge scenario timed out before producing any. The absence of evidence from these scenarios is therefore retained as an execution limitation rather than treated as a passing result.

5.3.3 Violation Analysis

Two scenarios produced MR violations across the dataset. These are summarised in Table 5.7, and the broader set of tested failure modes and their outcomes is given in Table 5.8.

Table 5.7: MR failures observed across the dataset.

Scenario	MR	Failed reports	Violations	Finding
Gothenburg large	MR3	7	39	Multiple chargers were assigned to more than one vehicle during overlapping windows.
Far-apart	MR7	4	4	High-quality validation evidence did not reach charging evidence.

Table 5.8: Tested failure modes and outcomes.

Failure Mode	Scenario	Observed
False positive (distance violation)	Far-apart	No
Duplicate charger assignment	Gothenburg large	Yes: 7 reports, 39 MR3 violations
Crossed pairing	Adjacent crossed	No
Session merge error	Return charge	No
Non-competitive drift	MR2 isolated stability	No: 20 reports, 0 MR2 violations
Order-dependent matching	MR6 ordered bridge	Inconclusive: 20 scheduled runs timed out before validation reports

The MR3 violations in the Gothenburg large scenario indicate that, under dense multi-vehicle conditions, the matching system did not consistently enforce charger exclusivity across overlapping assignment windows. Seven reports contained a total of 39 individual exclusivity violations, in which a single charger was assigned to more than one vehicle within an overlapping time window.

The MR7 violations in the far-apart scenario, comprising four reports and four violations, reflect a different class of finding: a propagation gap between high-quality

validation evidence and downstream charging evidence, rather than an incorrect match decision. The MR2 runs did not produce non-competitive match drift on the evidence available.

Critically, the violations observed across both MR3 and MR7 relate to structural properties of the matching process (exclusivity and evidence propagation) rather than to the system producing pairings that should not exist. This distinction is quantified against the aggregate matching outcome in Section 5.4. It is also worth noting up front, since the discussion in Chapter 6 returns to it, that MR violations identify the *circumstances* under which the SCS behaves inconsistently but do not by themselves explain *why*; in both the MR3 and MR7 findings, the root cause was only localized through repeated debugging runs against the violating scenarios, with the MRs serving as the entry point rather than the diagnosis.

The violations reported here were localised against internal SCS state during follow-up debugging (Section 6.4.1) but were not subjected to a structured confirmation review by the development team within the timeframe of this study.

5.3.4 False Positive Behaviour

The false positive analysis is restricted to control executions, that is, scenarios in which the expected outcome is the absence of violations. The isolated-stability scenario contributed 20 passing control executions to this analysis.

Table 5.9: False positive summary.

Metric	Value
Total control executions	397
Flagged control failures	7
Confirmed false positives	0
False positive rate	0.00%

Of the 397 control executions, seven were initially flagged. On manual review, all seven were traced to genuine matching behaviour rather than oracle artifacts, leaving no confirmed false positives (Table 5.9). The MRs therefore exhibit zero spurious failures on the unmodified SCS within the evaluated dataset, which is a precondition for treating the observed MR3 and MR7 violations as genuine behavioural deviations.

5.3.5 Mutation-Based Fault Detection: Scope Limitation

The validation dataset does not include a mutation testing pass. Mutation testing was planned in Chapter 4 as the controlled fault baseline most directly aligned with the RQ2 fault-detection sub-question, but producing a population of mutants large enough to constitute meaningful fault-detection evidence proved infeasible within the project's time budget. Building the mutation pipeline, curating a mutant population, and analysing the resulting outcomes at a volume comparable to the core

scenarios reported above would have required at least a doubling of the available development time, and was therefore not pursued within the resources of this study. The fault-detection evidence reported above is consequently restricted to observed violations on the unmodified SCS under varied and adversarial event conditions, which provides a partial but non-trivial answer to RQ2. The implications of the missing mutation baseline for the strength of the RQ2 claim are discussed in Chapter 6.

5.4 RQ3: Indicators of Viability for Event-Driven CPS

RQ3 asks to what extent simulation-based metamorphic testing constitutes a viable verification strategy for event-driven cyber-physical systems facing the oracle problem. This question is broader in scope than RQ1 and RQ2 and is addressed primarily through qualitative interpretation in Chapter 6. The empirical evidence reported here, which the discussion will draw on, can be summarised in four observations, anchored by the aggregate matching metrics in Table 5.10.

Table 5.10: Aggregate matching metrics.

Metric	Value
Total matches attempted	998
Successful matches	994
Correct rejections	4
Incorrect matches	0

First, the combined approach detected real behavioural deviations in a production CPS without any pre-computed ground truth: 43 MR violations across 11 reports, localised to two scenarios and traceable to structural properties of the matching process. At the same time, no incorrect matches were observed in the aggregate matching outcome across 998 attempted matches (994 successful, 4 correct rejections; Table 5.10). The oracle problem was therefore navigable in practice for this class of system: the framework characterised the SCS as functionally correct in its primary matching decisions while still exposing structural inconsistencies under dense and adversarial conditions.

Second, the relational properties most informative for fault detection, MR3 exclusivity and MR7 quality monotonicity, are properties that would not be expressible in a traditional input-output oracle, because they describe relationships across asynchronous event streams from independent actors rather than functional mappings. This is consistent with the structural argument developed in the related-work discussion: the SCS, as a representative event-driven CPS, sits outside the request-response assumption that underlies prior MT applications to backend services.

Third, the MR violations identified the *circumstances* under which the SCS pro-

duced inconsistent behaviour but did not on their own explain *why*. In both the MR3 and MR7 findings, the violation evidence indicated that a particular class of scenario (dense multi-vehicle layouts for MR3, distance-rejected pairings for MR7) consistently exposed the inconsistency, but localising the underlying cause within the SCS required follow-up debugging runs in which the violating scenario was replayed repeatedly while internal state was inspected. The MRs were therefore most useful as a triage signal (pointing at where to look) rather than as a diagnostic mechanism. This is itself a finding about how a relational oracle can be used in practice on a production CPS: the verification framework establishes *that* something is wrong and *under what conditions*, while standard debugging tools complete the picture of *why*.

Fourth, the operational limitations encountered, particularly the inability to complete the dense ordering scenarios within the configured timeout, indicate that the viability claim is bounded by the simulation environment's own capacity to execute the most adversarial conditions. A verification strategy that cannot reach its own stress boundary is viable only up to that boundary, and the discussion in Chapter 6 returns to what this implies for transferring the approach to other event-driven CPS contexts.

Full per-scenario outputs, detailed logs, and raw data tables are provided in Appendix A. The interpretation of these findings, and their implications for the three research questions, is developed in Chapter 6.

6

Discussion

This chapter interprets the results presented in Chapter 5 in light of the research questions stated in Section 1.2 and the research gaps identified in Section 3.6. The chapter is organised around the three research questions, with each section drawing together the relevant empirical evidence and assessing what it does and does not support. Sections on validity, implications for practice and research, and limitations and future work follow.

6.1 RQ1: Constructing a Simulation-Based Verification Framework

RQ1 asked how discrete-event simulation can be used to construct a systematic verification framework for an event-driven seamless EV charging system. The evidence for this question is operational: the framework was built, it executed reproducibly under controlled conditions, and it produced traceable execution evidence for every completed run (Section 5.2). On that basis, the construction itself can be answered affirmatively.

The more interesting question is what the construction *cost* in terms of coverage, and what that cost implies for the framework's standing as a verification environment. Of the 1,100 scheduled runs, 467 (42.5 %) produced validation reports. The remaining 633 did not complete, with the gap concentrated almost entirely in the densest scenarios (`same-hex-10-pairs`, `same-hex-10-staggered`, `mr6-ordered-bridge-evidence`, and `stress-grid-50-vehicles`). The interpretation that follows from this pattern is that the framework is reproducible and observable across the scenario classes it can execute, but that its coverage of high-density temporal stress is bounded by the execution environment rather than by the SCS itself. This is a finding about the framework's operational envelope, not about the SCS being un-testable under those conditions.

A practical implication follows from this for any future replication. Two parameters bound the coverage gap, and both are replaceable. The first is the 180 s simulated-time timeout applied to each run, which determines how much event processing the framework can absorb before terminating a scenario. The second is the single-laptop execution host on which the experiment suite was run, which sets a ceiling on the level of concurrency the framework can drive before the host becomes the bottleneck rather than the SCS. A dedicated test cluster, a longer per-run timeout, or both would likely close most of the gap; the coverage limitation is therefore a property of the execution environment used in this study and not an inherent limit of the methodology.

A second implication is architectural rather than operational. The reproducibility

result reported in Section 5.2, in which repeated executions of the same scenario with the same random seed produced identical event sequences and state transitions across all 467 validation reports, was made possible by the deliberate separation between the Simulator, the Test Interface, and the SCS established in Section 4.2.2. Without that separation, simulation logic would have leaked into protocol-specific request construction, and uncontrolled variation in the request layer would have made it impossible to attribute behavioural differences between runs to the intended scenario transformation. The reproducibility result therefore depends on an architectural decision and not only on seed control, and this is worth recording as an architectural finding rather than as a description of how the framework happens to be structured.

6.2 RQ2: Metamorphic Relations for Vehicle-to-Session Matching

RQ2 asked what metamorphic relations characterise correct behaviour in a vehicle-to-session matching system, and how effectively they detect faults under varied and adversarial event conditions. The eight relations developed in Section 4.3 were evaluated against the validation reports of the scenarios in which they were applicable. Six of the eight produced no violations across the dataset. The remaining two, MR3 (matching exclusivity) and MR7 (match quality monotonicity), exposed 43 violations across 11 reports concentrated in two scenarios.

6.2.1 What the Violations Indicate

The MR3 and MR7 violations are notable not for their volume but for their character. Neither violation corresponds to the SCS producing an incorrect match: across 998 attempted matches, 994 were successful and 4 were correct rejections, with zero incorrect matches recorded (Table 5.10). What the violations describe is structural inconsistency in the matching process. MR3 violations show that, under dense multi-vehicle conditions, the SCS did not consistently enforce charger exclusivity across overlapping assignment windows. MR7 violations show that high-quality validation evidence did not always propagate into the downstream charging evidence. The relations therefore exposed a class of fault that an input-output oracle would not have surfaced: one in which the primary matching decision is correct, but the surrounding structural guarantees on which that decision depends are not consistently maintained. This distinction is the bridge between the RQ2 evidence and the RQ3 viability argument, because it is precisely the class of fault that motivated the use of a relational oracle in the first place.

6.2.2 Which Relations Carried the Detection Load

Of the eight relations, only two produced violations on the available evidence. This concentration warrants comment, because it admits two readings. Either the SCS is genuinely robust on the six non-violating properties (MR1, MR2, MR4, MR5,

MR6, MR8), or those relations were not stressed hard enough by the scenarios that were able to complete. The honest reading is that the evidence supports different strengths of claim for different relations.

MR1 (determinism), MR4 (spatial soundness), MR5 (eligibility filtering), and MR8 (lifecycle) were exercised across hundreds of completed reports under varied conditions, and the absence of violations on this evidence base is meaningful: these four relations are confirmed to hold across the operational envelope the framework was able to exercise. MR2 (non-competitive stability) was exercised only in its dedicated isolated-stability scenario, which contributed 20 reports and 60 successful session matches with no violations. The result is consistent with MR2 holding, but the evidence base is small enough that the relation should be described as suggestively confirmed rather than fully established. MR6 (event idempotency and ordering) is the weakest case. The dedicated MR6 ordering scenario timed out in all 20 of its scheduled runs, so the zero-violation figure reported for MR6 reflects evaluation on artifacts that were not designed to stress ordering. MR6 should therefore be described as inconclusive on the available evidence rather than confirmed.

The honest aggregate framing is that MR3 and MR7 carried the detection load, four relations were meaningfully confirmed, and two (MR2 and MR6) have partial or inconclusive evidence respectively. The MR set as a whole was therefore not redundant: each retained relation either contributed a violation or covered a behavioural property that the others did not address, and the relations that did not fire are not evidence that they should be removed.

6.2.3 The Meaning of the False Positive Result

The zero confirmed false positives on 397 control executions (Table 5.9) is a precondition for treating the MR3 and MR7 findings as genuine. Seven control executions were initially flagged during validation and, on manual review, were traced to genuine matching behaviour rather than to oracle artifacts. This is worth recording explicitly, because it indicates that the MR oracles themselves are well-formed: they did not produce spurious failures on the unmodified SCS, and the seven initial flags were resolved as real behaviour rather than as noise in the relations. Without this property, the violations on MR3 and MR7 could be dismissed as oracle artifacts of the relations rather than as findings about the SCS. The false positive rate is therefore a property of the relations themselves, not of the SCS, and it is what makes the violation findings actionable rather than suggestive.

6.2.4 The Absence of a Mutation Baseline

The validation dataset does not include a mutation testing pass. This was identified in Chapter 4 as the controlled fault baseline that would most directly answer the RQ2 fault-detection sub-question, and its absence bounds what RQ2 can claim. Two things are worth stating clearly about that absence.

What it does not mean is that the framework is incapable of detecting faults. The

observed violations on MR3 and MR7 constitute evidence that the framework detects real behavioural deviations in a production system under varied and adversarial conditions, which is in some respects a stronger result than detecting seeded mutants: the violations correspond to inconsistencies in a system that engineers are actively responsible for, rather than to artificial faults introduced for measurement purposes. What it does mean is that the question "what fraction of seeded faults does this framework catch?" cannot be answered from the evidence in this thesis. Anyone reading the contribution must therefore treat the fault-detection claim as supported by observational evidence on a production system rather than by a controlled baseline against a known fault population. This is a credibility move rather than a weakness in the work: the claim is bounded honestly, and the boundary is named directly so that subsequent replication can establish the controlled rate that this thesis can only point at.

6.2.5 MRs as Triage Signal, Not Diagnostic Mechanism

A finding that emerged during result analysis but was not anticipated in the methods chapter is the role MRs played in the debugging workflow. In both the MR3 and MR7 cases, the relation identified the *conditions* under which the SCS behaved inconsistently but did not on its own localise the cause within the system. Root-cause analysis required follow-up debugging runs in which the violating scenario was replayed while internal SCS state was inspected, with the MRs serving as the entry point rather than the diagnosis.

This is worth elevating from an observation to a methodological point. The MR is doing the job a unit test cannot do on a system without an oracle: it tells the engineer that something is wrong, and under what scenario class, even though there is no pre-computed correct output to compare against. Standard debugging tools then complete the picture by localising the cause within the SCS. This is an honest framing of what MT contributes to a production engineering workflow. It is not a complete fault-localisation methodology, and it does not replace conventional debugging; what it provides is an entry point that did not exist before, in the sense that without the relational oracle there would have been no signal indicating that the dense-multi-vehicle and distance-rejection scenarios warranted further investigation in the first place. This contribution to the practical understanding of MT, that the relations function as a triage signal rather than as a self-contained diagnostic mechanism, is one of the more useful results to carry forward into the implications discussion below.

6.3 RQ3: Viability for Event-Driven Cyber-Physical Systems

RQ3 asked to what extent simulation-based metamorphic testing constitutes a viable verification strategy for event-driven cyber-physical systems facing the oracle problem. This question is broader than RQ1 and RQ2 and rests on a combination

of the empirical results above and a qualitative assessment of which assumptions transfer beyond the SCS.

6.3.1 What the Evidence Supports

The case for viability rests on three steps, each anchored directly in the results.

First, the combined approach detected real behavioural deviations in a production CPS without access to any pre-computed ground truth. The 43 MR violations across 11 reports were traceable to two structural properties, exclusivity and evidence propagation, and both findings emerged from a system that was processing asynchronous event streams under conditions where no input–output oracle could have been constructed. The oracle problem was navigable in practice for this class of system, and the framework characterised the SCS as functionally correct in its primary matching decisions while still exposing structural inconsistencies under dense and adversarial conditions.

Second, the relational properties most informative for fault detection, MR3 and MR7, could not have been expressed as input–output oracles because they describe relationships across asynchronous event streams from independent actors rather than functional mappings from a single request to a single response. This is consistent with the structural argument developed in Section 3.6: the SCS sits outside the request–response assumption underlying prior MT applications to backend services [37, 8], and the relations that carried the detection load are precisely those that exploit the system's event-driven structure. The fault findings are therefore not incidental to the methodology; they are findings that the methodology made visible and that conventional testing approaches could not have surfaced.

Third, the methodology is composed of two elements, discrete-event simulation and metamorphic testing, that are each well-established individually but had not previously been combined as a verification environment in which neither alone would have been sufficient. The transferable claim from RQ3 is therefore methodological rather than artefactual: the composition of DES and MT transfers to systems with the same structural properties (asynchronous, externally-dependent event streams, no tractable oracle), even though the specific MRs and the specific scenario library developed for the SCS do not.

6.3.2 What the Evidence Does Not Support

A viability claim must be honest about its boundary, and two boundaries are worth naming explicitly.

The first is the empirical base. The evidence in this thesis comes from a single industrial case study at a single organisation. The argument that the methodology transfers to structurally similar systems (event-driven backend services, IoT platforms, distributed transaction processors, other connected-vehicle services) is a structural argument made from one data point, not a result demonstrated across

multiple replications. The structural argument is defensible, but it should not be read as more than that until independent replication has been performed.

The second is the operational envelope. The framework's inability to complete the densest ordering and stress scenarios on the execution hardware available to this study means that the viability claim is bounded by what the simulation environment could actually exercise. A verification strategy that cannot reach its own stress boundary is viable up to that boundary, and whether the approach scales to genuinely high-concurrency CPS deployments remains an open question. This is a limitation that the thesis can identify but not resolve, and the discussion of future work below treats it accordingly.

6.3.3 Positioning Relative to Prior Work

The viability argument ties back to the three gaps identified in Section 3.6.

The first gap concerned MT for event-driven systems where correctness is relational across concurrent, independent streams. The MR3 and MR7 evidence shows that MT can be applied to such systems, with the important caveat that the MR identification process is manual and SUT-specific. Prior MT work on backend services [37, 8] relied on a request–response anchor for follow-up test construction; the MRs developed for the SCS demonstrate that relations can also be defined over event sequences when no such anchor exists, but this required direct study of the SUT rather than mechanical derivation from an interface specification.

The second gap concerned the combination of discrete-event simulation with MT as a substitute oracle. The framework itself, together with the reproducibility evidence reported in Section 5.2, supports the claim that DES and MT can be composed coherently in a single verification environment. To the authors' knowledge, this combination was not previously reported in the literature in the configuration adopted here, and the framework is presented as the concrete methodological artifact that closes this gap.

The third gap concerned the verification of seamless EV charging systems. The case study addresses this gap directly: the framework is, to the authors' knowledge, the first published verification framework for the domain, and the WirelessCar deployment provides the empirical grounding. The gap closure here is complete in the sense that no prior verification framework existed for this domain, and partial in the sense that one case study does not establish the methodology as standard practice for the domain.

6.4 Threats to Validity

This section discusses the threats to the validity of the findings along the dimensions established by Runeson and Höst [34]: internal validity, external validity, construct validity, and reliability. The validity framework was introduced in Section 4.1.5.

This section addresses what actually threatens the findings now that the results are known.

6.4.1 Internal Validity

Two confounds in the results warrant direct treatment.

The first concerns the exclusion of 56 MR3 violations from the shared-charger scenarios, documented in Appendix A.5. These violations were reviewed during result analysis and treated as simulator-side findings rather than as faults in the SCS, on the basis that the violation pattern was traceable to how the simulator constructed the shared-charger configurations rather than to inconsistent behaviour in the SCS itself. The exclusion was made deliberately during analysis and is recorded in the adjustment ledger so that the decision is auditable, but it should nonetheless be acknowledged that any post-hoc exclusion creates a confound: a reviewer could reasonably ask whether the retained 39 MR3 violations are themselves robust to a similar review. The mitigation here is that the retained Gothenburg large violations were reviewed against internal SCS state during follow-up debugging runs and confirmed to correspond to genuine exclusivity-enforcement inconsistencies in the SCS, rather than to simulator-side artifacts.

The second confound concerns the peer review of the MRs by the development team during MR consolidation (Section 4.3). Grounding the relations in the system's actual semantics through peer review was an internal-validity strength of the methodology, but the same process introduces the risk that the relations were implicitly tuned to behaviours the team already knew were correct, with the consequence that relations capable of detecting unknown faults could have been filtered out. The principal mitigation is that the relations were derived from input transformations against the SUT's interface rather than from implementation review, which limits but does not eliminate this risk. The MR3 and MR7 findings, both of which surfaced behaviours the development team had not previously characterised, provide some indirect evidence that the relations retained sufficient independence from team expectations to surface genuinely new findings.

6.4.2 External Validity

The most significant external validity threat is the reliance on a single industrial case study at a single organisation. As noted in Section 1.4, the findings are not intended to generalise directly to other event-matching systems without adaptation, and the strength of any transferability claim depends on being explicit about which elements of the work transfer and which do not.

Three elements transfer. The overall methodology of composing DES with MT as a substitute oracle is independent of the specific domain and applies to any system with the same structural properties (asynchronous event streams, no tractable oracle). The MR derivation process documented in Section 4.3 (SUT study, brainstorming, peer review, consolidation) is a procedure rather than a deliverable and

can be applied in other industrial contexts. The architectural separation between Simulator, Test Interface, and SUT, which underwrote the reproducibility result, is reusable in any simulation-based verification environment for an event-driven backend.

Three elements do not transfer. The specific MRs encode SCS-specific behavioural properties (charger exclusivity, evidence propagation across the matching pipeline) and would need to be re-derived for any other SUT. The specific scenario library encodes EV-charging-specific deployment contexts (waypoint networks, hex-based co-location, dense urban geometries) and is similarly SUT-specific. The absolute fault-detection numbers depend on the SCS's actual fault distribution under the scenarios exercised and cannot be projected onto other systems. Independent replication in other industrial contexts is the most direct way to test these distinctions, and is treated as future work below.

6.4.3 Construct Validity

The main construct validity threat is that the fault-detection claim rests on observed violations on the unmodified SCS rather than on a mutation baseline (Section 5.3.5). This affects which construct the evidence supports, and the distinction matters.

Observed violations on a production system are in some respects a stronger construct than seeded mutants: they correspond to real inconsistencies that production engineers are responsible for addressing, rather than to syntactic modifications whose representativeness of real faults is itself an assumption. However, observed violations cannot establish a fault-detection *rate*; they can only establish existence proofs. The claim "the framework detects faults" is supported by the evidence; the claim "the framework detects $X\%$ of faults" is not. This thesis makes the former claim and not the latter, and the distinction is recorded explicitly here so that the construct boundary is unambiguous.

A second construct validity question concerns whether the MRs measure what the thesis claims they measure. MR3 violations are interpreted as exclusivity faults in the SCS, but they could in principle be artifacts of how scheduling windows are defined in the simulator. The shared-charger MR3 exclusion documented in Appendix A.5 is the concrete evidence that this distinction was drawn deliberately during analysis: the simulator-side findings were separated from the SCS-side findings precisely because the construct validity of MR3 depends on the violations corresponding to behaviour in the SCS rather than in the simulation environment. The retained MR3 violations are those that survived this distinction.

6.4.4 Reliability

Reliability is the dimension on which the evidence is strongest. Repeated executions of the same scenario with the same random seed produced identical event sequences and state transitions across all 467 validation reports, as reported in Section 5.2. No instances of uncontrolled non-determinism were observed within the simulation layer

itself, and the reproducibility result therefore holds across the operational envelope the framework was able to exercise.

Two residual sources of variance are worth naming. The 633 incomplete scheduled runs reflect host-machine performance under high-concurrency scenarios rather than non-determinism in the framework; the same scenarios on different hardware would likely complete more often, but the runs that did complete were deterministic across replays. The MR identification process is reproducible in principle (the four-step procedure is documented in Section 4.3) but not deterministic in practice: different teams applying the procedure to the same SUT would likely arrive at overlapping but non-identical MR sets, since the brainstorming and consolidation steps involve human judgement. This is an honest limit on full reliability and applies to MT methodology generally rather than to this work specifically, but it should be recorded here so that the reliability claim is correctly scoped.

6.5 Implications for Practice

Three implications can be defended from the results.

The first is specific to WirelessCar. The framework provides a verification layer that the existing manual testing process cannot supply at any feasible scale. The 39 MR3 violations under dense conditions, in particular, are findings that would be very difficult to surface through physical testing alone: they emerged only under multi-vehicle configurations whose reproduction in a physical environment would require coordinating multiple vehicles, multiple chargers, and overlapping session timings under controlled conditions that the company does not currently have the operational capacity to produce. The framework should therefore be understood as an addition to, rather than a replacement for, manual testing: it covers a region of the operational envelope that manual testing cannot reach, and surfaces a class of finding (structural inconsistency under concurrency) that manual testing is poorly suited to expose.

The second is for teams building structurally similar event-driven backend systems. The MR derivation process documented in Section 4.3 is reusable. The specific MRs are not, but the procedure (study the SUT, brainstorm transformations against its inputs, peer-review with the development team, consolidate into a tractable set) is portable to any system where the oracle problem applies and where direct collaboration with the development team is possible. For teams in this position, the methodology offers an entry point into systematic verification that does not depend on the existence of a tractable input–output oracle.

The third concerns cost. Developing the framework was not free: the simulator, the test interface, the scenario library, and the MR identification process together represented substantial engineering effort, and any team adopting this approach should weigh that cost against the alternative. For systems with the oracle problem, however, the alternative is typically no systematic verification at all rather than a

cheaper systematic verification, and the comparison should be framed accordingly. The framework is not free, but the alternative for systems in this class is worse.

6.6 Implications for Research

This section reconnects the findings to the research gaps identified in Section 3.6.

The first gap concerned MT for event-driven systems where correctness is relational across concurrent, independent event streams without an initiating request. The MR3 and MR7 findings constitute an existence proof that MT can be applied to this class of system, and the peer-reviewed MR derivation process documents how relations can be identified for such systems without formal design assumptions of the kind required by prior CPS work [27] and without the request–response anchor that prior backend MT work relied on [37, 8]. The contribution to this gap is therefore both an empirical result (relations of this kind can be defined and they detect faults) and a methodological one (a process for deriving them when the standard anchors are absent).

The second gap concerned the combination of discrete-event simulation with MT as a substitute oracle. The framework itself, together with the reproducibility evidence, supports the claim that this combination is methodologically coherent. To the authors' knowledge, this combination was not previously reported in the literature in the configuration adopted here, and the framework is presented as the concrete methodological artifact addressing the gap. What the thesis does not establish is that this combination is the only viable approach to the oracle problem in event-driven CPS, or that it dominates alternatives that future work might develop; what it establishes is that the combination works in at least one industrial context and is therefore worth taking seriously as a candidate methodology.

The third gap concerned the verification of seamless EV charging systems. The case study addresses this gap directly. This is, to the authors' knowledge, the first published verification framework for the domain, and the domain itself is representative of a broader class of connected-vehicle and IoT backend services facing similar verification challenges (asynchronous event correlation across independent actors, no cable-level handshake, no pre-computable ground truth). The contribution to this gap should be read as opening a domain rather than closing it: the framework demonstrates that systematic verification is achievable in this space, but it does not establish a standard practice and does not preclude alternative approaches.

6.7 Limitations and Future Work

The scope limitations set out in Section 1.4 bound what this thesis could address empirically. Beyond those, several additional limitations emerged during the work and point toward directions for future research.

Limitations

Three limitations are worth naming explicitly, beyond what the scope statement already covers.

First, mutation testing was not produced within the project's resource budget, so the fault-detection claim rests on observational evidence rather than on a controlled baseline (Section 5.3.5). The framework detects faults, but the rate at which it detects faults relative to a known fault population cannot be established from this thesis.

Second, the framework could not execute the densest ordering and stress scenarios within the configured 180 s timeout on the available execution hardware (Section 5.2.2). The viability claim for RQ3 is therefore bounded by what the framework could actually exercise, and the most adversarial conditions remain unevaluated.

Third, MR identification relied on direct collaboration with the development team responsible for the SCS. In contexts where such collaboration is not available, the four-step derivation process documented in Section 4.3 would need adaptation, and the relations produced would likely be less grounded in the SUT's actual semantics. The MR derivation process is portable in principle, but its effectiveness depends on inputs that are not always available in industrial settings.

Future Work

Four directions follow directly from the limitations above.

The first is mutation-based evaluation. Producing a population of mutants and re-executing the framework against it would establish the fault-detection rate that this thesis can only point at, and would provide the controlled baseline that the RQ2 sub-question on detection effectiveness was originally designed to answer. This is the most direct way to convert the existence-proof result into a rate-of-detection result.

The second is execution scaling. Running the framework on dedicated test infrastructure rather than a development laptop would likely close most of the scheduled-versus-completed coverage gap and allow the dense ordering scenarios, in particular the dedicated MR6 scenario, to be evaluated meaningfully. This would also convert the MR6 result from inconclusive to either confirmed or violating, and would clarify whether the operational envelope of the framework is bounded by methodology or by hardware.

The third is replication in other event-driven backend systems. This is the most direct test of the RQ3 transferability claim, and the candidate domains identified in Section 3.6 (IoT backend services, distributed transaction processing, other connected-vehicle services) all share the structural properties that motivated the methodology in the first place. A replication in any one of these domains would convert the structural argument for transferability into an empirically supported

one.

The fourth is automated or semi-automated MR generation. The manual MR identification bottleneck is the principal obstacle to applying this methodology at industrial scale, and it is also the element of the methodology that is least amenable to mechanical replication. Investigating whether MRs for event-driven systems can be derived from system specifications, log analysis, or learning-based methods is a substantial open research direction. Prior work on automated MR generation has focused on mathematical and scientific software where input–output relationships can be expressed as formal invariants; extending those approaches to event-driven systems whose correctness is relational across asynchronous streams would address one of the most significant scalability barriers identified in this work.

7

Conclusion

This thesis set out to address a verification problem that conventional testing approaches are not equipped to solve: how to systematically validate the behaviour of an event-driven cyber-physical system whose correctness depends on the interplay of asynchronous event streams from independent actors, and for which no precomputable ground truth exists. The Seamless Charging Service operated by WirelessCar AB was taken as the representative case, and the oracle problem together with the infeasibility of physical testing at scale defined the verification gap this work was designed to close.

The response was a simulation-based metamorphic testing framework combining discrete-event simulation as the test execution environment with a set of metamorphic relations as a substitute oracle. Eight relations characterising correct vehicle-to-session matching behaviour were derived through a four-step process of source-code study, brainstorming, peer review with the development team, and consolidation. The framework was evaluated against the unmodified Seamless Charging Service across 18 scenarios spanning baseline correctness, dense multi-vehicle configurations, ambiguous spatial layouts, and lifecycle progressions.

The empirical results support three claims. The framework executed reproducibly across the 467 validation reports it produced, made possible by the architectural separation between Simulator, Test Interface, and System Under Test. Six of the eight relations were preserved across the dataset, while matching exclusivity and match quality monotonicity exposed 43 violations across eleven reports, traceable to structural inconsistencies in a system that nonetheless produced zero incorrect matches across 998 attempted matches. No confirmed false positives were observed across the 397 control executions, establishing that the violation findings are attributable to the system under test rather than to oracle artifacts.

The contributions map directly onto the gaps identified in the related-work analysis. The framework demonstrates that discrete-event simulation and metamorphic testing can be composed into a single reproducible and observable verification environment (RQ1), that metamorphic relations can be defined over event sequences from concurrent independent streams without the request-response anchor prior backend work relied on and without the formal design assumptions presupposed in prior cyber-physical applications (RQ2), and that the combined approach constitutes a viable verification strategy for at least one industrial system facing the oracle problem (RQ3). A methodological observation that emerged during result analysis warrants restatement: the metamorphic relations functioned in practice as a triage signal rather than as a self-contained diagnostic mechanism, identifying the conditions under which the system behaved inconsistently while root-cause analysis required follow-up debugging against internal state. The relational oracle does not replace conventional debugging tools; it provides an entry point that did not exist

before.

The boundaries of the work are part of the contribution. The fault-detection claim rests on observed violations on a production system rather than on a mutation-based baseline, which was not feasible to produce at scale within the available resources. The framework's coverage of the densest ordering and stress scenarios was bounded by the execution hardware and the configured timeout, leaving the most adversarial conditions partially unevaluated. The transferability argument rests on one industrial case study and on a structural reading of the methodology rather than on independent replication. Each limitation defines a direction for future work: a controlled mutation-based evaluation, execution on dedicated test infrastructure, and replication in other event-driven backend domains.

The Seamless Charging Service is treated here as a representative member of a broader class of systems (connected vehicle services, distributed transaction processors, Internet of Things backend platforms) that share asynchronous event correlation across independent actors, no pre-computable ground truth, and an operational envelope too large to reproduce through physical testing alone. For systems in this class, the alternative to a verification framework of the kind developed here is typically no systematic verification at all rather than a cheaper one. The framework, the relations, and the derivation process are presented as a starting point for that broader class of work, with the case study at WirelessCar AB as the empirical grounding from which the argument proceeds.

Glossary

CCS2 Combined Charging System 2; a standardized EV charging connector supporting AC and DC charging, mandated in the European Union. 6

Charging Infrastructure The combined hardware, software, and network systems that enable EV charging. 15

CPO Charge Point Operator; an organization responsible for operating and maintaining electric vehicle charging infrastructure. 1, 2, 4–10, 19, 21, 25, 29

CPS Cyber-Physical System; a system integrating computational components with physical processes. 2, 3, 17, 19, 21–24

DES Discrete-Event Simulation; a simulation paradigm where state changes occur at discrete events. 14

DSR Design Science Research; a research methodology focused on the iterative design, development, and evaluation of artifacts to solve identified problems in real-world contexts. 3, 21

eMSP eMobility Service Provider; a company that provides services such as charging access, authentication, and billing for EV users. 1, 6, 7, 9

EV Electric Vehicle; a vehicle powered partially or entirely by electric motors using energy stored in rechargeable batteries. 1, 2, 5–9, 15

EVSE Electric Vehicle Supply Equipment; hardware and control systems that supply electrical energy to charge an electric vehicle. 5

ICE Internal Combustion Engine; a type of engine in which fuel is combusted within the engine itself to generate mechanical power. 7

ISO 15118 A standard defining communication between EVs and charging stations. 7–9

MR Metamorphic Relation; a property that must hold across the outputs of related source and follow-up executions, given a known transformation of the input. A violation constitutes evidence of a fault even when no correct output is known. 3, 11, 13, 14, 16–19, 22, 36–42

MT Metamorphic Testing; a testing technique that addresses the oracle problem by verifying that metamorphic relations hold across the outputs of related

executions, rather than asserting an absolute expected output. 2, 3, 11, 13, 14, 16, 17, 19, 36, 37, 41

NACS North American Charging Standard; an EV charging connector supporting AC and DC charging, derived from Tesla's connector design and increasingly adopted as a common standard in the United States. 6

OCPI Open Charge Point Interface; a protocol enabling interoperability between CPOs and eMSPs. 2, 6, 7, 9, 10, 14, 15

OCPP Open Charge Point Protocol; a protocol enabling communication between charging stations and backend systems. 2, 4, 6, 7, 9, 10, 14, 15

OEM Original Equipment Manufacturer; an automotive manufacturer responsible for designing, producing, and integrating vehicle systems under its brand. 2, 4, 6, 9, 10, 14, 21, 25, 29

Plug and Charge A feature enabling automatic authentication and payment when connecting an EV to a charging station. 5, 7–9

RFID Radio Frequency Identification; a wireless technology used for identification and authentication via radio waves. 8, 9

SCS Seamless Charging System; The system implementing Seamless Charging in the evaluated case study. 1–8, 10, 11, 13–15, 17, 19, 21–23, 25–30, 33, 53

SoC Separation of Concerns; a design principle that separates a system into distinct components. 27, 29

SUT Software Under Test; the system or component being tested. 3, 4, 11, 13, 16, 17, 24, 27, 29, 33, 36–40

Type 2 Standard AC charging connector used in Europe. 6

V&V Verification and Validation; verification ensures a system is built correctly with respect to its specification (building the model right), while validation ensures it meets the requirements of its real-world domain (building the right model). 13, 15, 24, 33

Bibliography

[1] E. Altamimi, A. Elkawakjy, and C. Catal, "Metamorphic relation automation: Rationale, challenges, and solution directions," *Journal of Software: Evolution and Process*, vol. 35, no. 1, Art. no. e2509, 2023, doi: 10.1002/smr.2509.

[2] T. Arts, J. Hughes, J. Johansson, and U. Wiger, "Testing telecoms software with Quviq QuickCheck," in *Proc. 2006 ACM SIGPLAN Workshop on Erlang*, 2006, pp. 2–10, doi: 10.1145/1159789.1159792.

[3] J. Ayerdi, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "QoS-Aware Metamorphic Testing: an Elevation Case Study," in Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), Coimbra, Portugal, 2020, pp. 104–114. [Online]. Available: `https://doi.org/10.1109/ISSRE5003.2020.00019`, Accessed on: 2026-05-05.

[4] A. Bahrami, "EV Charging Definitions, Modes, Levels, Communication Protocols and Applied Standards," Technical Report, Jan. 2020, doi: 10.13140/RG.2.2.15844.53123/11. [Online]. Available: `https://www.researchgate.net/publication/338586995_EV_Charging_Definitions_Modes_Levels_Communication_Protocols_and_Applied_Standards`, Accessed on: 2026-05-12.

[5] O. Balci, "Validation, verification, and testing techniques throughout the life cycle of a simulation study," *Annals of Operations Research*, vol. 53, no. 1, pp. 121–173, 1994, doi: 10.1007/BF02136828.

[6] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing: A Survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, May 2015, doi: 10.1109/TSE.2014.2372785.

[7] California Energy Commission, "Statement on the North American Charging Standard—J3400," California Energy Commission, Sacramento, CA, USA, Docket 22-EVI-06, TN 252421, 2023. [Online]. Available: `https://efiling.energy.ca.gov/GetDocument.aspx?tn=252421`, Accessed on: 2026-05-05.

[8] W. K. Chan, S. C. Cheung, and K. R. P. H. Leung, "A Metamorphic Testing Approach for Online Testing of Service-Oriented Software Applications," *International Journal of Web Services Research*, vol. 4, no. 2, pp. 61–81, Apr.–Jun. 2007, doi: 10.4018/jwsr.2007040103.

[9] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Hong Kong, HKUST-CS98-01, 1998. Reprinted as

arXiv:2002.12543, 2020. [Online]. Available: https://arxiv.org/abs/2002.12543, Accessed on: 2026-05-05.

[10] K. Claessen and J. Hughes, "QuickCheck: a lightweight tool for random testing of Haskell programs," in *Proc. 5th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP '00)*, 2000, pp. 268–279, doi: 10.1145/351240.351266.

[11] P. Dini, S. Saponara, S. Chakraborty, and O. Hegazy, "System-level compact review of on-board charging technologies for electrified vehicles: architectures, components, and industrial trends," *Batteries*, vol. 11, no. 9, Art. no. 341, 2025, doi: 10.3390/batteries11090341.

[12] European Alternative Fuels Observatory (EAFO), "Recharging systems," 2026. [Online]. Available: https://alternative-fuels-observatory.ec.europa.eu/general-information/recharging-systems (accessed on: 2026-02-19).

[13] P. Fabianek and R. Madlener, "Multi-criteria assessment of the user experience at e-vehicle charging stations in Germany," *Transportation Research Part D: Transport and Environment*, vol. 121, Art. no. 103782, 2023, doi: 10.1016/j.trd.2023.103782.

[14] "Public EV charging in Europe: Market dynamics and consumer perspectives," FIA Region I, Brussels, Belgium, ED21732, 2025. [Online]. Available: https://www.fiaregion1.com/wp-content/uploads/2025/10/ED21732-FIA-Research-on-Electricity-Market-Dynamics-and-E-Mobility-Report_FINAL-1.pdf, Accessed on: 2026-04-28.

[15] K. N. Hasan, R. Preece, and J. V. Milanović, "Efficient Identification of Critical Uncertainties Affecting Small-Disturbance Stability of Power Systems Considering Parameter Correlation," in *Proceedings of the 2018 IEEE Power & Energy Society General Meeting (PESGM)*, Portland, OR, USA, 2018, pp. 1–5, doi: 10.1109/PESGM.2018.8586568.

[16] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, no. 1, pp. 75–106, Mar. 2004, doi: 10.2307/25148625.

[17] Road vehicles—Vehicle to grid communication interface—Part 1: General information and use-case definition, ISO Standard 15118-1:2019, International Organization for Standardization, Geneva, Switzerland, 2019. Available: https://www.iso.org/standard/69113.html.

[18] L. Jiang, X. Diao, Y. Zhang, J. Zhang, and T. Li, "Review of the charging safety and charging safety protection of electric vehicles," *World Electric Vehicle Journal*, vol. 12, no. 4, Art. no. 184, 2021, doi: 10.3390/wevj12040184.

[19] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph

kernels," *Software Testing, Verification and Reliability*, vol. 26, no. 3, pp. 245–269, May 2016, doi: 10.1002/stvr.1594.

[20] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed., ch. 6, "Selecting Input Probability Distributions." New York, NY, USA: McGraw-Hill, 2000. [Online]. Available: `https://staff.universitaspahlawan.ac.id/web/upload/materials/958-materials.pdf`, Accessed on: 2026-04-20.

[21] E. A. Lee, "The Problem with Threads," *Computer*, vol. 39, no. 5, pp. 33–42, May 2006, doi: 10.1109/MC.2006.180.

[22] E. A. Lee, "Cyber Physical Systems: design Challenges," in Proceedings of the 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, USA, 2008, pp. 363–369. [Online]. Available: `https://ieeexplore.ieee.org/document/4519515`, Accessed on: 2026-05-05.

[23] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET), Austin, TX, USA, 2017, pp. 35–41. [Online]. Available: `https://ieeexplore.ieee.org/document/7966963`, Accessed on: 2026-05-05.

[24] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?" *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 4–22, Jan. 2014, doi: 10.1109/TSE.2013.46.

[25] H. Liu and H. B. Kuan Tan, "Covering Code Behavior on Input Validation in Functional Testing," *Information and Software Technology*, vol. 51, no. 2, pp. 546–553, Feb. 2009, doi: 10.1016/j.infsof.2008.07.001.

[26] Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, "An Empirical Analysis of Flaky Tests," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*, Hong Kong, China, 2014, pp. 643–653, doi: 10.1145/2635868.2635920.

[27] C. Mandrioli et al., "Testing CPS With Design Assumptions-Based Metamorphic Relations and Genetic Programming," *IEEE Transactions on Software Engineering*, vol. 51, no. 6, pp. 1666–1684, Jun. 2025, doi: 10.1109/TSE.2025.3563121.

[28] A. M. Memon, "An event-flow model of GUI-based applications for testing," *Software Testing, Verification and Reliability*, vol. 17, no. 3, pp. 137–157, Sep. 2007, doi: 10.1002/stvr.364.

[29] T. Y. Chen et al., "Metamorphic Testing: A Review of Challenges and Opportunities," *ACM Comput. Surv.*, vol. 51, no. 1, Art. no. 4, Jan. 2018, doi: 10.1145/3143561.

[30] Open Charge Point Interface (OCPI) 2.3.0, EVRoaming Foundation, 2025. Available: https://evroaming.org/wp-content/uploads/2025/02/OCPI-2.3.0.pdf.

[31] OCPP 2.1, Part 0—Introduction, Technical Specification, Edition 2, Open Charge Alliance, 2025. Available: https://www.openchargealliance.org/.

[32] C. Pacheco, S. K. Lahiri, M. D. Ernst, and T. Ball, "Feedback-directed random test generation," in *Proc. 29th Int. Conf. on Software Engineering (ICSE '07)*, 2007, pp. 75–84, doi: 10.1109/ICSE.2007.37.

[33] M. Pezzè and M. Young, *Software Testing and Analysis: Process, Principles, and Techniques*. Hoboken, NJ, USA: John Wiley & Sons, 2008. [Online]. Available: https://ix.cs.uoregon.edu/~michal/book/Samples/book.pdf, Accessed on: 2026-05-20.

[34] P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Apr. 2009, doi: 10.1007/s10664-008-9102-8.

[35] J3400: North American Charging System (NACS) for electric vehicles, SAE Standard J3400, SAE International, 2024. Available: https://www.sae.org/standards/content/j3400/.

[36] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A Survey on Metamorphic Testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, Sep. 2016, doi: 10.1109/TSE.2016.2532875.

[37] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of RESTful web APIs," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1083–1099, Nov. 2018, doi: 10.1109/TSE.2017.2764464.

[38] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic Testing: Testing the Untestable," *IEEE Software*, vol. 37, no. 3, pp. 46–53, May–Jun. 2020, doi: 10.1109/MS.2018.2875968.

[39] Tesla, Inc., "Opening the North American Charging System," 2022. [Online]. Available: https://www.tesla.com/blog/opening-north-american-charging-system (accessed on: 2026-03-10).

[40] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, pp. 297–312, 2012, doi: 10.1002/stvr.456.

[41] R. G. Sargent, "Verification and validation of simulation models," in Proceedings of the 2010 Winter Simulation Conference, Baltimore, MD, USA, 2010, pp. 166–183. [Online]. Available: https://ieeexplore.ieee.org/document/5679166, Accessed on: 2026-05-05.

[42] E. J. Weyuker, "On Testing Non-Testable Programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, Nov. 1982, doi: 10.1093/comjnl/25.4.465.

[43] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Berlin, Germany: Springer, 2014.

[44] WirelessCar, "WirelessCar products," 2026. [Online]. Available: `https://www.wirelesscar.com/products/` (accessed on: 2026-03-10).

[45] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, Oct. 2020, doi: 10.1109/TSE.2018.2876433.

A

Appendix

A.1 Dataset Sources

Table A.1: Validation datasets used for the report

Dataset	Scheduled	Complete reports
Primary campaign	1060	447
Supplemental MR2/MR6 coverage	40	20
Blended report dataset	**1100**	**467**

A.2 Scenario-Level Data

Table A.2: Scenario execution matrix for the blended dataset

Scenario	Scheduled	Complete reports	Primary MRs
`adjacent-hex`	57	16	MR3, MR4, MR5
`adjacent-hex-crossed-fail`	70	41	MR3, MR4, MR5
`adjacent-hex-crossed-timeout`	73	25	MR3, MR4, MR5
`adjacent-hex-single-crossed`	64	41	MR3, MR4, MR5
`city-2-vehicles`	55	55	MR1, MR3, MR7
`far-apart-no-match`	100	4	MR4, MR5
`gothenburg-large-16-vehicles`	55	7	MR1, MR3, MR7
`metro-3-vehicles`	55	16	MR1, MR3, MR7
`same-hex`	54	24	MR1, MR3, MR7
`same-hex-10-pairs`	55	0	MR1, MR3, MR7
`same-hex-10-staggered`	56	0	MR1, MR3, MR6, MR7
`shared-charger-2-vehicles`	56	56	MR3, MR4, MR5
`single-1-vehicle`	100	100	MR4, MR8
`single-return-charge`	100	41	MR8
`stress-grid-50-vehicles`	55	1	MR1, MR3, MR7
`three-hex`	55	20	MR1, MR3, MR7
`mr2-isolated-stability`	20	20	MR2, MR4, MR8
`mr6-ordered-bridge-evidence`	20	0	MR6, MR1, MR3, MR7
Total	**1100**	**467**	

A.3 MR Evidence

Table A.3: Metamorphic-relation evidence matrix

MR	Associated scenarios	Scheduled	Completed	Evaluated reports	Violations
MR1	9	460	123	467	0
MR2	1	20	20	265	0
MR3	14	780	246	467	39
MR4	8	540	303	463	0
MR5	6	420	183	463	0
MR6	1	20	0	467	0
MR7	9	460	123	442	4
MR8	3	220	161	463	0

Targeted complete reports are derived from primary scenario-design associations. Evaluated reports count relation-oracle evaluations on complete artifacts, including relations that were checked outside their targeted scenario group.

A.4 Session Outcomes

Table A.4: Aggregate matching and session outcomes

Dataset	Attempted	Successful	Correct rejections	Incorrect matches
Primary campaign	938	934	4	0
Supplemental MR2 coverage	60	60	0	0
Blended report dataset	**998**	**994**	**4**	**0**

A.5 Reporting Adjustments

Table A.5: Adjustment ledger for reportable validation outcomes

Item	Count	Treatment
Primary complete validation reports	447	Base campaign reports with persisted validation output.
Primary raw failed validation reports	67	Includes shared-charger MR3 findings later treated as simulator-side.
Shared-charger MR3 reports excluded	56	Excluded from reportable failures after review.
Reportable primary failed validation reports retained	11	Seven Gothenburg large reports and four far-apart reports.
Supplemental MR2 passing reports added	20	Added to blended control evidence with no new MR violations.
Blended complete validation reports	467	Primary complete reports plus supplemental MR2 reports.
Raw primary MR3 violations	95	Relation-level total before shared-charger exclusion.
Shared-charger MR3 violations excluded	56	Relation-level simulator-side findings.
Reportable MR3 violations retained	39	Gothenburg large exclusivity violations.
Reportable MR7 violations retained	4	Far-apart evidence-propagation violations.
Blended reportable MR violations	**43**	39 MR3 violations plus 4 MR7 violations.

A.6 Findings and Gaps

Table A.6: Reportable findings and incomplete evidence

Scenario	Evidence type	Scheduled	Complete reports	Outcome
`large-16-vehicles`	MR3 finding	55	7	39 reportable exclusivity violations across seven reports.
`far-apart-no-match`	MR7 finding	100	4	Four reportable evidence-propagation violations; no incorrect matches.
`same-hex-10-pairs`	Incomplete dense stress evidence	55	0	No complete validation reports in the primary campaign.
`same-hex-10-staggered`	Incomplete ordering stress evidence	56	0	No complete validation reports in the primary campaign.
`mr6-ordered-bridge`	Incomplete MR6 target evidence	20	0	All scheduled supplemental runs timed out at 180 s before validation reports.