Validating Autonomous Driving: Metamorphic Testing for Autonomous Vehicles and Simulators

Thesis submitted to the University of Nottingham for the degree of
Doctor of Philosophy, 2025.

Yifan Zhang

20318993

Supervised by

Prof. Dave Towey
Dr. Matthew Pike
Prof. Xu Sun

Abstract

The development of autonomous driving systems (ADSs) represents a significant evolution in transportation, with simulation being a crucial factor in their development. However, one of the challenges for these ADS-related systems, due to their complexity and critical nature, is the oracle problem. In software testing, an oracle is a mechanism used to systematically verify the correctness of the outputs for any given input. The oracle problem arises when it is difficult or impossible to find or use such an oracle. Metamorphic testing (MT) has proven to be an effective approach to alleviate the oracle problem. It uses metamorphic relations (MRs)—relations among multiple inputs and their corresponding outputs—to verify test results. The process of generating MRs remains a significant challenge in the application of MT, especially in complex systems like ADSs. Traditionally, MRs are produced by domain experts, while approaches like metamorphic relation patterns (MRPs) and metamorphic relation input patterns (MRIPs) provide a structured approach to the generation of MRs. Additionally, the rise of large language models (LLMs), such as ChatGPT, provides opportunities to reduce the manual effort involved in the MT process. Metamorphic exploration (ME), as an extension of MT, employs hypothesized MRs (HMRs) to assess the system under test (SUT). A violation of these relations may not indicate a defect but can reveal the user's misunderstanding of the SUT, prompting further exploration of the system.

This thesis aims to enhance MT in ADS testing by alleviating the oracle problem

in these systems, improving testing efficiency, and promoting educational practices. Through a series of experiments, it demonstrates how MT can be effectively applied to tackle challenges in ADS testing. Key contributions include applying ME and MT to the Baidu Apollo ADS, which leads to enhanced system comprehension and the identification of conflicting obstacle-detection results in the perception-camera module. An ADS-based test harness is designed to improve testing efficiency and is validated through an industry case study.

While MT is proven effective in testing ADSs, it still heavily depends on human knowledge, particularly in the MR generation process. This thesis introduces the use of ChatGPT for generating and evaluating MRs, along with a set of evaluation criteria for objective assessments of MR quality. A GPT evaluator was also developed, demonstrating AI's potential to assist beginners and enhance MT practices. Comparative studies with human-generated MRs further underscore the potential of LLMs and highlight areas for educational improvement. Additionally, an Open Educational Resource (OER) is introduced to provide guidelines and templates for teaching beginners about scenario and MR generation.

To address the challenges of identifying whether anomalies originate from the ADS or the simulator, this thesis applies MT to Autonomous Driving (AD) simulators to enhance ADS testing, revealing critical bugs in NIO and CARLA simulators, and demonstrating MT's effectiveness in ensuring simulator reliability. A scenario-driven MT framework integrating ME and MT is proposed to enhance defect discovery and reporting, and is validated through an industry case study. Additionally, MRPs and MRIPs tailored for AD systems are introduced, enabling effective defect detection. These are further incorporated into a human-AI hybrid MT framework with a test harness to streamline MR generation and automate test case execution, enhancing testing efficiency.

In summary, through practical experiments and methodological improvements, the thesis fills significant gaps in MT and ADS testing. It demonstrates the practical utility of MT in ADS testing, introduces templates, integrates AI into the MT process, and develops educational resources along with advanced frameworks and tools. These contributions enhance testing efficiency, reliability, and educational practices in the field of MT and ADSs.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Prof. Dave Towey, Dr. Matthew Pike, and Prof. Xu Sun, for their invaluable guidance and support throughout my PhD project. Their expertise and mentorship have been crucial in the development and completion of this work.

I am also thankful to my family for their unwavering support, both emotionally and financially. Their encouragement provided me with the strength and motivation to overcome the challenges of this journey.

Lastly, I wish to express my heartfelt thanks to my fiancée, Ding, for her constant support and understanding during this time. Her patience and belief in me have been indispensable in helping me reach this milestone.

Publications

As of now, the author has published ten academic papers during his PhD, which includes nine conference papers and one journal paper. Furthermore, two journal papers are currently under review.

Conference Papers

1. **Y. Zhang**, M. Pike, D. Towey, J. C. Han and Z. Q. Zhou, "Preparing future SQA professionals: An experience report of metamorphic exploration of an autonomous driving system," in *2022 IEEE Global Engineering Education Conference (EDUCON)*, Tunis, Tunisia, 2022, pp. 2121-2126, doi: 10.1109/EDUCON52537.2022.9766791.

2. **Y. Zhang**, D. Towey, M. Pike, J. C. Han, G. Zhou, C. Yin, Q. Wang, and C. Xie, "Metamorphic testing harness for the Baidu Apollo perception-camera module," in *2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)*, Melbourne, Australia, 2023, pp. 9-16, doi: 10.1109/MET59151.2023.00009.

3. **Y. Zhang**, D. Towey and M. Pike, "Automated metamorphic-relation generation with ChatGPT: An experience report," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, Torino, Italy, 2023, pp. 1780-1785, doi: 10.1109/COMPSAC57700.2023.00275.

4. **Y. Zhang**, D. Towey, M. Pike, Z. Q. Zhou, and T.Y. Chen, "Enhancing ADS testing: An Open Educational Resource for metamorphic testing", in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, Osaka, Japan, 2024, pp. 138-147, doi: 10.1109/COMPSAC61105.2024.00029.

5. **Y. Zhang**, D. Towey and M. Pike, "Enabling effective metamorphic-relation generation by novice testers: A pilot study", in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMP-*

SAC), Osaka, Japan, 2024, pp. 2398-2403, doi: 10.1109/COMPSAC61105.2024.00384.

6. **Y. Zhang**, D. Towey, M. Pike, R. Qiu, A. T. Jaya, S. Huey, X. Zhang, and Y. Wu, ”Metamorphic testing of a steer-by-wire system: An intercultural students-as-partners collaboration experience,” in *Proceedings of the 9th ACM International Workshop on Metamorphic Testing (MET'24)*, Vienna, Austria, 2024, pp. 19-25. doi: 10.1145/3679006.3685069.

7. D. Towey, Z. Luo, Z. Zheng, P. Zhou, J. Yang, P. Ingkasit, C. Lao, M. Pike, and **Y. Zhang**, “Metamorphic testing of an automated parking system: An experience report,” in *Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC 2023)*, Torino, Italy, 2023, pp. 1774-1779, doi: 10.1109/COMPSAC57700.2023.00274.

8. Z. Zhang, D. Towey, Z. Ying, **Y. Zhang** and Z. Q. Zhou, “MT4NS: Metamorphic testing for network scanning,” in *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*, Madrid, Spain, 2021, pp. 17-23, doi: 10.1109/MET52542.2021.00010.

9. **Y. Zhang**, M. Pike, and D. Towey, “No pain, no gain: The necessary initial struggles to enable doctoral research work,” in *2021 International Conference on Open and Innovative Education (ICOIE 2021)*, Hong Kong, The Open University of Hong Kong, 2021, pp. 437-450.

Journal Papers

1. **Y. Zhang**, D. Towey, M. Pike, J. C. Han, Z. Q. Zhou, C. Yin, Q. Wang, and C. Xie, “Scenario-Driven Metamorphic Testing for Autonomous Driving Simulators”, *Software Testing, Verification and Reliability*, vol. 36, Art. no. e1892, Jul. 2024. doi: 10.1002/stvr.1892.

2. **Y. Zhang**, D. Towey, M. Pike, Q.-H. Luu, H. Liu, and T. Y. Chen, “Integrating Artificial Intelligence with Human Expertise: An In-depth Analysis

of ChatGPT's Capabilities in Generating Metamorphic Relations", submitted to *Information and Software Technology*, **under review**.

3. **Y. Zhang**, D. Towey, M. Pike, Q.-H. Luu, H. Liu, and T. Y. Chen, "Integrating Artificial Intelligence with Human Expertise: An In-depth Analysis of ChatGPT's Capabilities in Generating Metamorphic Relations," *arXiv preprint arXiv:2503.22141*, 2025. [Online]. Available: `https://arxiv.org/abs/2503.22141`.

Abbreviations

AD	Autonomous Driving
ADAS	Advanced Driver Assistance System
ADSs	Autonomous Driving Systems
AI	Artificial Intelligence
FDS	Function Definition Specification
HMR	Hypothesized Metamorphic Relation
LLM	Large Language Model
ME	Metamorphic Exploration
MG	Metamorphic Group
ML	Machine Learning
MR	Metamorphic Relation
MRIP	Metamorphic Relation Input Pattern
MRP	Metamorphic Relation Pattern
MT	Metamorphic Testing
OER	Open Educational Resources
RO	Research Objective
RQ	Research Question
SUT	System Under Test

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The development of autonomous driving systems (ADSs) represents one of the most transformative technological advancements of the 21st century. By integrating complex software and hardware, these systems improve the safety, efficiency, and convenience of transportation [61]. The Society of Automotive Engineers (SAE) classifies ADSs into six levels [19], from Level 0 (no automation) to Level 5 (full automation), each representing a different degree of driving task automation. Level 0 relies entirely on human control, while Level 5 refers to vehicles capable of managing all driving tasks independently in any situation.

Various industry leaders and car manufacturers, including Waymo [193], Tesla [14], and NIO [146], are developing ADS technologies. These systems are built on key components such as sensors, perception systems, decision-making algorithms, and control mechanisms [19]. Sensors like LiDAR, radar, cameras, and GPS gather data about the vehicle's environment [131]. The perception system processes this data to identify objects, understand the surroundings, and determine the vehicle's position [147]. Decision-making algorithms use this information to make real-time driving decisions, while control mechanisms execute these decisions through acceleration, braking, and steering [131]. Additionally, Vehicle-to-Everything (V2X) communication enhances safety and efficiency by enabling vehicles to interact with each other and the surrounding infrastructure [147]. ADSs offer various advanced driving functions that enhance safety, convenience, and efficiency [19]. Features like adaptive cruise control (ACC) [225], lane-keeping assistance (LKA) [207], and automated parking (AP) [210] are becoming standard. More sophisticated systems, such as traffic jam assist [252] and highway pilot [120], allow for autonomous driving (AD) under specific conditions.

Among the open-source ADS platforms, Autoware is popular for its modular architecture and adherence to safety standards. However, it requires significant calibration and expertise to achieve optimal performance [105]. OpenPilot by Comma.ai equips the existing vehicles with self-driving capabilities by using AI-based vision and sensors, although it has limitations like the lack of pedestrian detection and complex installation requirements [43]. Baidu's Apollo platform, targeting Level 4 automation, offers robust tools and resources and is widely used in ADS development due to its open-source nature and comprehensive documentation [71].

Ensuring the reliability and correctness of ADSs is a significant challenge due to their complexity and the critical nature of their application [19], which often faces the oracle problem [254]. In software testing, an oracle is a mechanism used to verify the correctness of the system's outputs for any given input (test case) [21]. The oracle problem occurs when finding or using such an oracle is challenging or impossible [21]. Most conventional software testing techniques are based on the assumption that an applicable oracle exists, which does not always hold true when testing complex applications [254]. Metamorphic testing (MT) has emerged as an effective solution to alleviate the oracle problem [47]. It has been widely adopted in various complex software systems, such as web services [192], embedded systems [39], compilers [113], databases [125], machine-learning classifiers [103], and search engines [84]. MT offers several advantages, including simplicity, cost-effectiveness, and the ability to be automated [47, 127].

MT uses metamorphic relations (MRs), relations among multiple inputs and their corresponding outputs, to verify software correctness [47]. MRs usually include source inputs, follow-up inputs, source outputs, and follow-up outputs [175]. In MT, the source input is passed to the system under tests (SUTs) to produce a source output. Then by applying a specific transformation (specified by MRs) to this input, a set of follow-up inputs is generated, and their outputs are compared with the source output with the expected relations defined by MRs [45]. If the MR is violated, it indicates the potential existence of system defects that requires further investigation [255]. A metamorphic group (MG) is a sequence of source inputs and follow-up inputs related to MR [47]. Different MGs can be constructed under the same MR [133]. The effectiveness of MT relies on the quality of MRs and MGs [47]. Metamorphic Exploration (ME), as an extension of MT, is a testing approach to improve system understanding and use [255]. In this context, MRs are not necessarily required to be accurate; instead, they are hypothesized by users (referred to as hypothesized MRs (HMRs)); these users then use the HMRs to explore the software system, thereby enhancing their understanding of the system and their ability to use it effectively [255].

The effectiveness of MT is significantly dependent on the quality of MRs, while generating high-quality MRs continues to be a challenge and an area of active research [123]. Various methods have been developed to facilitate the generation of MRs [123]. Traditional approaches often rely on human expertise and are applicable to specific types of SUTs, which can limit the scalability and generalizability of MRs [107, 242]. One approach to overcoming these limitations is to compose new MRs from existing ones, with empirical studies showing that composite MRs can be more effective at fault detection than their individual component MRs [123, 159]. Additionally, structured methods such as the *category-choice* framework offer a systematic way to generate and organize test scenarios based on system specifications, improving the relevance of MRs produced [51, 152].

MRs were primarily generated from scratch in early studies, while metamorphic relation patterns (MRPs) provide a way to the systematic generation of MRs [123]. MRPs are abstractions that describe collections of MRs, often applicable across various domains [255]. By using MRPs, the process of generating MRs can become more structured and streamlined [249]. The concept of MRPs can be further extended to two subclasses of patterns: Metamorphic relation input patterns (MRIPs) [255] and metamorphic relation output patterns (MROPs) [176]. MRIPs are abstractions that define the relations between the source and follow-up inputs within a set of MRs [174]. On the other hand, MROPs describe relations between the source and follow-up outputs [176]. This approach has proven effective in generating MRs for testing various systems, such as search engine [255], ADSs [249], and deep learning libraries [129].

More advanced methods for MR generation include the use of AI techniques [103]. For instance, Machine learning (ML) approaches such as decision trees and support vector machines [103] have been employed to predict MRs. Furthermore, neural-network-based approaches and natural language processing (NLP) techniques have also proven effective in automating the MR generation process [25, 245]. As a significant advancement in NLP [253], Large Language Models (LLMs) are designed to understand and generate human language, making them a useful tool for a wide range of applications, including MR generation [247, 253]. Some of the most recognized LLMs include OpenAI's GPT (Generative Pre-trained Transformer) series [213], Google's BERT (Bidirectional Encoder Representations from Transformers) [63], and Meta's LLaMA (Large Language Model Meta AI) [93]. Through training on vast amounts of text from various sources, LLMs could learn basic language concepts such as grammar, facts, and the contextual use of words [253]. After the pre-training phase, these models are fine-tuned on targeted datasets for specific tasks, which enhances their performance in applications such as question answering, text summarization, and language translation [213].

In addition to its general applications, LLMs have shown potential in simplifying the MR generation process. Shin et al. [181] proposed an approach using few-shot prompting, where the model was provided with only a few examples, to instruct LLMs to derive executable MRs (EMRs) from requirements and API specifications. This method has demonstrated the capability to produce understandable and relevant MRs and EMRs, highlighting the potential for LLMs to enhance the MR generation process. This PhD project adopted ChatGPT [151], an LLM developed by OpenAI, which has demonstrated significant capabilities across various tasks, including generating source code, formulating test cases, and aiding in debugging programs [153]. ChatGPT includes two commercial versions, GPT-3.5 and GPT-4, both making significant advancements in AI language processing [92]. In Chapter 4, experiments were conducted involving using ChatGPT (GPT-3.5 [247] and GPT-4[1]) to automatically generate MRs for the parking module of an ADS, finding that the quality of generated MRs varied based on system complexity, test scenario specificity, and the number of MRs requested, necessitating human intervention for validation and quality improvement.

MT has been widely applied in identifying and resolving critical issues within ADSs [17, 59, 83, 94, 126, 133, 197, 205, 233]. For example, researchers have used MT to test the control software of unmanned aerial vehicles (UAVs), identifying scenarios where the system would fail under certain rotations [126]. Similarly, a framework called DeepTest was developed to generate synthetic images and test self-driving car decisions, uncovering dangerous behaviour in corner cases (i.e., rare and unusual situations) [197]. Further applications of MT in ADSs include the development of a declarative Rule-based MT (RMT) framework [59],

[1]This work was in the review process for the Information and Software Technology (IST) Journal.

which uses natural language to specify traffic scenarios and identify safety issues, and a sequential MT framework [133] that uses sequences of MGs to understand ADS behaviours without needing ground-truth datasets, revealing undesirable behaviours in top-ranked ADS models.

Due to the open-source nature of the Baidu Apollo ADS, several studies have applied MT to this platform. For instance, Zhou and Sun [254] used MT to find bugs in the perception module, revealing a high likelihood of obstacle detection failures under certain conditions. Han and Zhou [83] introduced a novel approach combining MT with fuzz testing to identify fatal errors in the Apollo ADS, enhancing its robustness and safety. In Chapter 3, ME was applied [246] to the perception and localization modules, enhancing the system understanding of the testers by detecting anomalies violating HMRs. In the subsequent MT experiments [250], the brightness of camera-captured input images was adjusted, and inconsistent obstacle detection results were identified across sequential frames in the perception-camera module, highlighting the shortcomings of the algorithm [250].

In ADS development, simulators provide a safe and controlled environment for testing complex algorithms and dangerous driving scenarios (Figure 1.1) [240]. Notable simulators include CARLA [66] and SVL [168]. CARLA offers a flexible platform that can simulate various real-world scenarios, enabling comprehensive testing of algorithms and sensor integrations [66]. While SVL is no longer officially supported by its company [194], it provides realistic 3D animations and detailed data for ADS testing [168].

Simulators facilitate rapid and cost-effective testing, allowing for the early detection and resolution of potential issues [240]. They can recreate dangerous or complex scenarios that would be risky to test on real roads, significantly simplifying and accelerating the development process [124]. The data generated from simulators also aids in training ML models, enhancing the adaptability and effectiveness of ADSs [33].

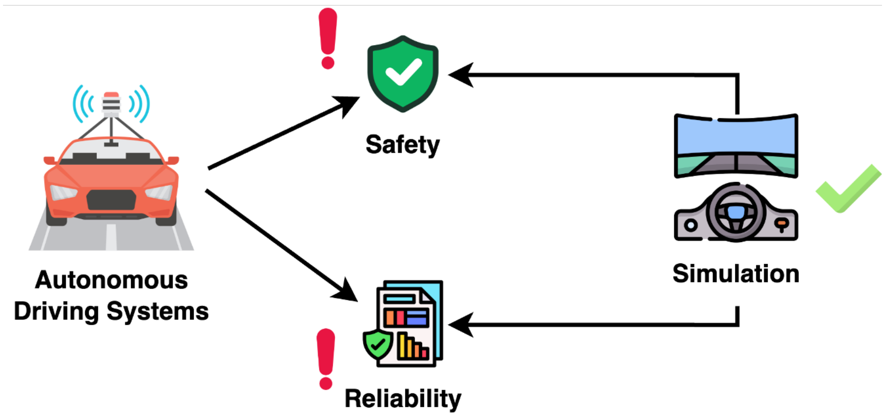


Figure 1.1: Simulation plays an important role in ensuring the safety and reliability of the ADSs

However, the use of any simulator, including CARLA, necessitates a critical evaluation and validation process. For a simulator to be useful in testing and development, it must accurately represent real-world conditions [240]. The validity of a simulator is a crucial factor in assessing its capability to accurately represent real-life driving situations [186]. This is typically assessed by conducting a direct comparison between simulated and real-world driving, and employing statistical tests such as t-tests [108] to determine whether or not there is any significant difference between the data values. However, testers continue encountering the oracle problem, such as validating large amounts of data and replicating real-world scenarios [223].

MT has been applied to simulation models to ensure their reliability and accuracy. For instance, Ahlgren et al. [4] discussed the deployment of the Metamorphic Interaction Automaton (MIA) to test Facebook's large-scale, web-based simulation infrastructure, which resolved inconsistent test results and improved overall simulation reliability. Olsen and Raunak [149] proposed using MRs to establish pseudo-oracles for simulation validation, improving the trustworthiness of simulation results by ensuring accurate system representation. Adigun, Eisele, and Felderer [3] applied MT to simulate ADS scenarios, which reveals flaws in vehicle perception and control algorithms under realistic traffic conditions and corner cases. Iqbal et al. [95] focused on lane-keeping assistance systems, using MT to create test cases that simulate a range of driving conditions, effectively identifying faults related to lane detection, steering control, and system responsiveness. In Chapter 5, MT was applied to multiple AD simulators (NIO [249] and CARLA[2]), uncovering various issues on the simulator's built-in functions such as the compilation of scenario elements and underlying logic of the simulation system.

As MT continues to gain popularity, finding effective methods to educate students, professional software engineers, testers, and end-users about it has emerged as a critical issue [47]. One of the challenges in educating testers to generate MRs, especially for complex systems such as ADSs, arises from the constraints of limited time and resources [127]. This difficulty is particularly acute in scenarios such as guest lectures or workshops where participants, often novice testers, are introduced to the complexities of MR generation in a condensed timeframe [76]. For example, in a university setting, a guest introduction lecture might offer only a single session to cover the principles of MRs, leaving little room for in-depth exploration or hands-on practice. Similarly, project teams facing urgent deadlines may need to quickly upskill members in MR generation to meet testing requirements, operating under the dual pressures of time scarcity and resource limitations [218]. These situations highlight the need for efficient and effective training methods that can rapidly equip participants with the necessary skills to generate meaningful MRs, despite the challenge of navigating these constraints.

Although MT shows effectiveness in alleviating the oracle problem, its widespread implementation encounters significant first-order barriers. Ertmer et al. [69] refer to these as external factors, which include resource availability and institutional support, particularly noting the limited availability of educational resources and training in this specialized area. In the context of ADS testing, the concept of MT poses a substantial hurdle due to a lack of comprehensive and accessible educational materials. This creates a knowledge gap, especially for beginners and

[2]This work was in the review process for the Information and Software Technology (IST) Journal.

professionals seeking to advance their understanding of MT and its application in ADSs.

To address these first-order barriers, in Chapter 6, an Open Educational Resource (OER) for MT was presented specifically in the context of testing ADSs. OERs are freely accessible, openly-licensed text, media, and other digital assets useful for teaching, learning, and assessing, as well as for research purposes [206]. Originating in the early 2000s [206], OERs reflect the principles of the open-source movement [116], with the goal of removing barriers to education. They advocate for the free and open sharing of information and resources, primarily through the Internet [37]. OERs include diverse materials like textbooks and curricula, all of which can be adapted under open licenses like Creative Commons [117]. This flexibility aids educators in customizing content, and can foster innovative teaching. OERs enhance educational access, particularly for underprivileged learners, aligning with global educational equity goals [184].

In summary, this research examined the use of MT in ADS testing, and proposed various frameworks and guidelines aimed at improving MT efficiency and streamlining the generation of MRs. Furthermore, the research explored the integration of current methodologies (MRPs) with AI technologies (LLMs) to address existing challenges in MT. It also introduced an OER to further promote the application of MT in the area of ADS testing.

1.2 Research Objectives and Questions

This thesis aims to enhance ADS testing by advancing the field of MT. The research involves proposing a comprehensive suite of methodologies, frameworks, and experiments designed to address critical challenges, including the efficiency of MT processes, the difficulty of MR generation, and the overall applicability of MT in ADS testing. The three main objectives of this thesis—examining the effectiveness of MT in ADS testing, enhancing MT efficiency, and tackling the MR generation challenge—are closely interrelated. The challenges encountered during MT applications, such as difficulties in MR generation and test automation, which in turn motivates the creation of methodologies and frameworks designed to boost its efficiency. By addressing the MR generation challenge, this thesis aims to lower the barriers for beginners and testers in new systems when using MT. Furthermore, through automating and regulating the process of MT in ADS testing, it seeks to make MT more accessible and applicable. Together, these efforts form a cohesive approach to advancing MT in the ADS testing domain.

Specifically, the research objectives (ROs) were:

RO1. Examine the effectiveness of MT for ADS testing, involving ADS and AD simulators.

- This objective aimed to explore how MT could be applied to test ADSs like Baidu Apollo, which is popular and open-source. The goal was to improve MT efficiency and automate the testing process to make it more effective and manageable. Furthermore, given the significance of simulations in ADS testing [124], the validity of AD simulators was also important to ensure the effectiveness of ADS testing. This involved applying MT to these simulators to validate their performance and reliability, ensuring that they accurately simulate real-world scenarios.

RO2. Tackle the challenge of MR generation, especially for beginners and testers in new systems.

- Generating MRs is a complex task that typically requires domain expertise and testing experience. This can be especially challenging for those new to MT or testing new systems [47]. This objective aimed to simplify the MR generation process by proposing methods that enhance existing knowledge (e.g., MRPs) and integrate advanced technologies (e.g., LLMs) to facilitate and streamline the generation of MRs.

RO3. Enhance MT efficiency by automating and regulating the process of ADS testing, while also lowering the entry barriers for beginners using MT.

- This objective focused on improving the efficiency of MT for ADS testing. It involves developing tools and frameworks that can handle repetitive and complex testing tasks without extensive manual intervention. By automating these processes, the goal was to make MT more accessible and lower the barriers for beginners who are new to using MT in ADS testing. Meanwhile, as MT continues to gain popularity, finding effective methods to educate testers about it has emerged as a critical challenge [47]. Developing platforms that can educate users about MT would promote its use in ADS testing and encourage further research in this area.

Based on these ROs, the following research questions (RQs) were proposed:

RQ1. How can MT be effectively applied in ADS testing to uncover anomalies and enhance system understanding?

RQ2. How can the difficulty of generating MRs be reduced to assist beginners and testers in understanding and testing new systems?

RQ3. How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?

1.3 Contributions

This thesis presents several significant contributions to the field of MT and ADS testing by alleviating the oracle problem in these systems, enhancing testing efficiency, and promoting educational practices. Through a series of experiments, it demonstrates how MT can be effectively applied to tackle challenges in ADS testing. The main contributions can be separated into four parts, where each part corresponds to the contents from Chapter 3 to Chapter 6, respectively, and are closely interconnected with ROs:

Contribution 1 Improving ADS through MT

Contribution 2 Enhancing MR Generation with ChatGPT

Contribution 3 Advancing AD Simulator Testing with MT

Contribution 4 Promoting MT through Education

All the works in the project began with an ME on the Baidu Apollo ADS (**Contribution 1**), as shown in Figure 1.2, identifying an apparent problem that was later confirmed to be a misunderstanding, which enhanced the author's system understanding and laid a solid foundation for subsequent experiments (Section 3.2). The later MT experiments on the object-detection algorithm of the perception-camera module revealed conflicting obstacle-detection results due to brightness adjustments in both individual and sequential driving scenarios, addressing the oracle problem in the module (Section 3.3). During this process, the author designed and implemented an ADS-based test harness, focusing on automating MT in ADS testing. The development of this harness increased testing efficiency and helped testers organize the testing procedure, which was validated through an industry case study (Section 3.3.2). In summary, the successful applications of ME and MT help examine the MT effectiveness in ADS

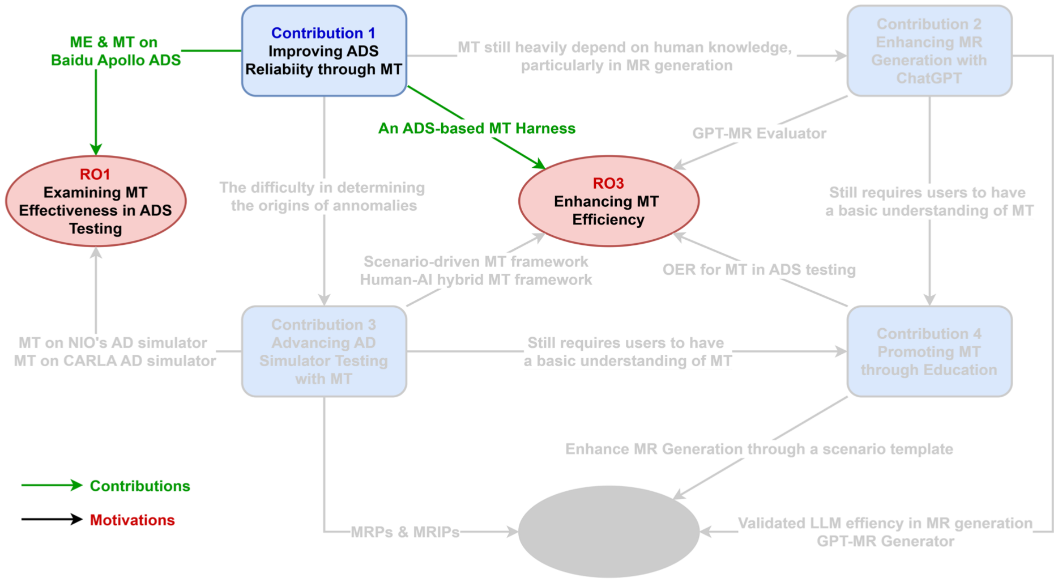


Figure 1.2: Details of Contribution 1

testing, achieving **RO1** of the thesis, while the proposed ADS-based MT harness enhanced testing efficiency that achieves **RO3** (Figure 1.2).

However, as Figure 1.3 shows, although MT was effective for testing ADSs, it still heavily depended on human knowledge, particularly in the MR-generation process (red texts in the image). This motivated the work presented in Chapter 4, which involves a new approach by introducing AI into MR generation, specifically LLMs, and assesses the quality of the MRs to demonstrate its feasibility (**Contribution 2**). As one of the most popular LLMs, the capabilities of ChatGPT (GPT-3.5 and GPT-4) were assessed in generating MRs for SUTs across diverse domains (Sections 4.2 and 4.3). This was supported by proposing a series of MR-evaluation criteria, including an initial version and a refined version, to offer a more comprehensive, effective, and objective assessment of MRs (Sections 4.2.1 and 4.3.2). A customized GPT-MR evaluator was also created, and performed comparably to human evaluators, indicating that AI can enhance MT practices and assist beginners in generating MRs (Section 4.3.3.2), as part of the efforts to fulfill **RO3** of the project. The evaluation results demonstrated the advanced capabilities of ChatGPT, especially GPT-4, in software testing and MR

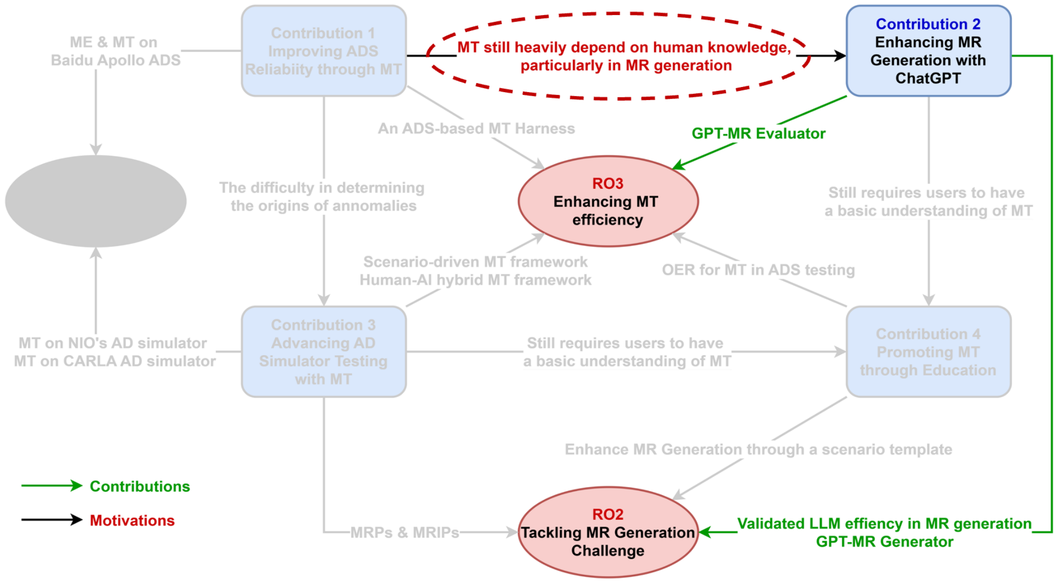


Figure 1.3: Details of Contribution 2

generation across a wide array of applications, successfully achieving **RO2**. The quality of the GPT-generated MRs was further compared with human-generated ones in Chapter 6. The two studies demonstrate the potential of beginners with appropriate guidance to quickly generate useful MRs that possess similar qualities to those generated by powerful LLMs: The first was a pilot study with a small number of participants, comparing the MRs with GPT-3.5 (Section 6.3.1); the second was a subsequent study that involved recruiting more participants and comparing the MRs with GPT-4 (Section 6.3). These studies also underscore areas for improvement in the teaching and training of both humans and LLMs.

Additionally, as shown in Figure 1.4, the challenge of determining whether anomalies arise from the ADS, the simulator, or the simulated data in the testing of Baidu Apollo ADS underscores the essential role that simulation plays in this process (red texts in the image). Consequently, this thesis involves the application of MT to AD simulators as a means of enhancing the overall effectiveness of ADS testing (**Contribution 3**). In Chapter 5, MT was conducted on two AD simulators, the NIO AD simulator and the CARLA simulator. In the case of the NIO AD simulator, MT was implemented alongside the NIO ADS, which

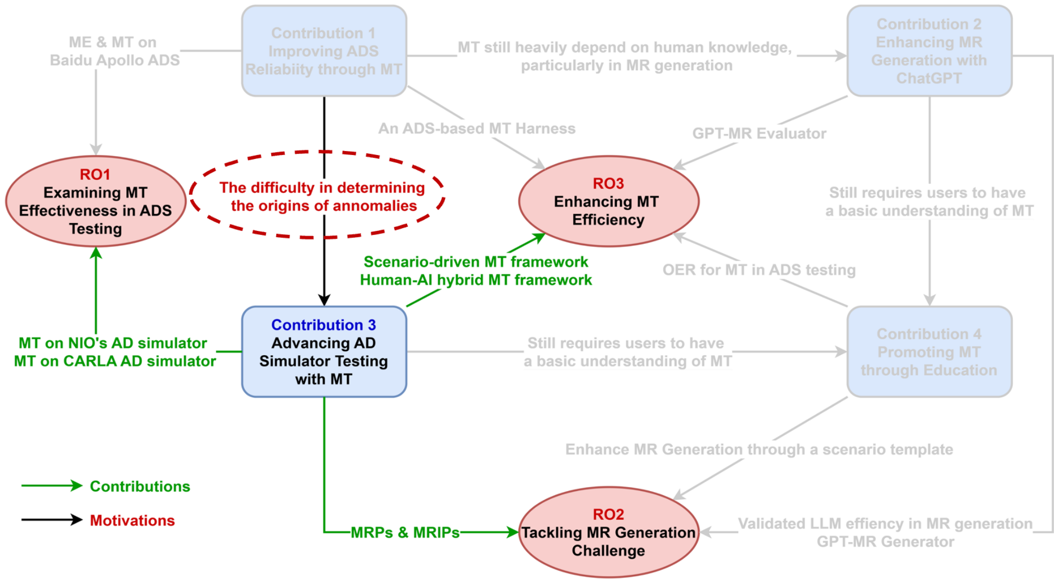


Figure 1.4: Details of Contribution 3

had not been previously tested using MT (Section 5.2). Instead of conventional unit testing, scenario-based testing was implemented to detect defects in various components. The author developed a scenario-driven MT framework integrating ME and MT to enhance defect discovery and reporting (and as a work to achieve **RO3**): An industry case study illustrated the framework's strengths and limitations (Section 5.2.2). Furthermore, during the testing, a set of MRPs and MRIPs were proposed to facilitate the generation of MRs due to the lack of MRPs that specifically target AD-related systems in the literature (Section 5.2.1). The MRs generated from these MRPs and MRIPs have been empirically demonstrated to effectively reveal defects within the AD simulator, achieving **RO2** of the thesis.

However, one challenge of testing the simulator in conjunction with the ADS was to identify the source of the anomalies. Therefore, to further explore the performance of MT in AD simulators, the author applied MT on the CARLA simulator (Section 5.3), discovering four significant bugs related to simulator behaviours that conventional testing methods might have missed. Meanwhile, given the effectiveness of ChatGPT in facilitating MR generation and test automation (Chapter 4), the author designed a human-AI hybrid MT framework that com-

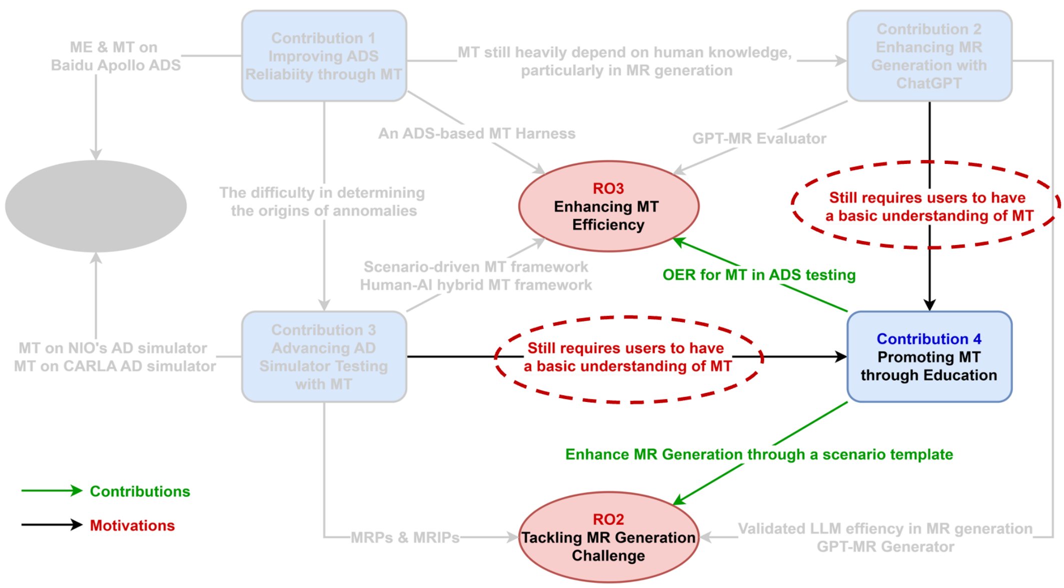


Figure 1.5: Details of Contribution 4

bines human inputs with AI-driven automation to generate and refine MRs for evaluating various aspects of the CARLA simulator (Section 5.3.1). This framework also includes a test harness to automate the generation and execution of test cases, thereby improving the efficiency and effectiveness of the testing process in support of achieving **RO3**. Both MT experiments on AD simulators complement the previous work on Baidu Apollo ADS (Chapter 3), successfully achieving **RO1** of the thesis.

While the experiments on ADSs and AD simulators showcase the effectiveness of MT in facilitating ADS testing, these methods still require that users have a basic understanding of MT, as Figure 1.5 shows. Additionally, finding effective methods to educate students, professional software engineers, testers, and end-users about MT has become a critical challenge [47]. To promote MT in ADS testing (**Contribution 4**), the author developed an OER, including instructions for scenario and MR generation, aimed at improving educational approaches and supporting beginners in the field, which also achieves **RO3**. The MR-generation guideline and test-case generation template simplify the initial steps in scenario creation, making MR generation more efficient and understandable for beginners

(Section 6.2), and achieving **RO2** of the thesis. These materials were later used to guide students in generating MRs targeting specific ADS functions (Sections 6.3.1 and 6.3), highlighting how beginners, with proper guidance, can rapidly produce valuable MRs.

In conclusion, this research makes significant contributions to MT and ADS testing through experiments and methodological advancements. It demonstrates the practical utility of MT in ADS testing by addressing the oracle problem, revealing critical defects, and improving system understanding. The introduction of the templates and guidelines for test case and MR generation alleviates the challenges of MT and lowers the barriers to ADS testing. Moreover, integrating AI into MR generation and evaluation reduces reliance on human expertise and improves MT practices. The development of educational resources like the OER addresses critical challenges in teaching MT, supporting beginners and promoting wider adoption. The proposed frameworks and tools, including test harnesses and hybrid MT frameworks, significantly enhance testing efficiency, reliability, and educational practices in the field of ADSs.

1.4 Thesis Outline

The rest of the thesis is structured into the following chapters:

Chapter 2 (Background) provides the background to the research, beginning with a discussion of traditional testing methods and the oracle problem. It then introduces MT and ME, including key research in MR generation. The chapter also covers ADSs, reviewing existing open-source ADSs and the application of MT in these systems. Further, it explores AD simulators and existing MT in simulation environments. Lastly, the chapter examines LLMs, specifically ChatGPT, and existing research on the integration of MT with LLMs.

Chapter 3 (Improving Autonomous Driving System Reliability with Metamorphic Testing) focuses on the application of ME and MT for testing ADSs, with a specific focus on the Baidu Apollo ADS. It presents the experiments conducted on this platform, specifically targeting the perception and localization modules. An ADS-based test harness is also introduced to enhance testing efficiency, validated through an industry case study. The chapter highlights the effectiveness of ME and MT in the identification of defects and the enhancement of system understanding.

In Chapter 4 (Enhancing Metamorphic Relation Generation with ChatGPT), the thesis examines how AI technologies can enhance MR generation and automate testing processes. It focuses on the application of LLMs, particularly ChatGPT (GPT-3.5 and GPT-4), in generating MRs. Additionally, the chapter presents a set of criteria for evaluating MRs, along with the customization of a GPT-MR evaluator, and assesses its effectiveness in improving the evaluation process for MRs.

Chapter 5 (Advancing Autonomous Driving Simulator Testing through Metamorphic Testing) extends the application of MT to AD simulators in the ADS testing domain, presenting experiments conducted on the NIO AD simulator and CARLA simulator. It then introduces MR generation techniques, involving three MRPs and MRIPs, along with guidelines for generating MRs for common ADS modules. The chapter also discusses frameworks designed to enhance MT efficiency and effectiveness, including a scenario-driven MT framework and a human-AI hybrid MT framework. A significant number of defects were revealed through the MT approach, demonstrating the effectiveness of MT in enhancing the validity of AD simulators.

In Chapter 6 (Promoting Metamorphic Testing through Education), the thesis presents an OER designed to promote MT among students and beginners in MT and ADSs. This OER includes a scenario template for improving ADS testing, along with guidelines for MR generation. The chapter also compares the performance and quality of MRs created by ChatGPT and novice students, highlighting how AI-driven methods can enhance MT practices and support beginners in producing high-quality MRs.

Chapter 7 (Discussion and Conclusion) provides a comprehensive summary of the thesis. It reflects on the RQs and discusses the overall impact of the work on the field of ADS testing. Additionally, the chapter highlights the importance of this research in furthering the use of MT in ADSs and suggests possible directions for future investigation.

Chapter 2

Background

2.1 Challenges in Traditional Testing: The Oracle Problem

Ensuring the correctness and reliability of complex systems is significant in software testing [47]. A fundamental aspect of software testing is the use of an oracle, which is a mechanism used to determine the correctness of the system behaviour or output for given inputs [21]. The majority of conventional software testing methodologies rely upon the existence of an oracle. However, in complex systems, it may be infeasible or impractical to find an oracle (i.e., the oracle problem) [47]. There are four types of test oracles [21]: specified test oracles, derived test oracles, implicit test oracles, and no test oracle. Specified test oracles work by judging whether or not the behaviour of the program satisfies the formal specifications, such as model-based specification languages [112] and assertions [85]. Derived test oracles assess the correctness of the system depending on the information derived from certain conditions or models (e.g., MRs from MT), instead of depending on explicitly defined outputs [21]. Implicit test oracles highlight the issues that can be easily noticed, such as program crashes, and no test oracle applies to situations where a test oracle is absent [21].

This thesis focuses on MT, which falls under the category of derived test oracles. Other testing strategies in this category include pseudo-oracles and regression testing [21]. A pseudo-oracle is one of the earliest versions of the derived test oracle [21, 57]. It involves producing a different version of the same program from a separate development team. If the original program is considered to be validated, the outputs of these two programs with the same set of inputs should be similar or identical. Otherwise, defects may exist in one or both versions, prompting the need for further debugging to identify the source of the inconsistency [57]. The main weakness of this method is that it could only prove the existence of the defect without being able to find it. In addition, this method is not suitable for large and complex programs due to the expense of developing another version. Furthermore, a complete and precise specification must be available to both programming teams [21]. This can be challenging if the other team is from a different department or company, due to trade secrets and intellectual property concerns.

Regression testing is another testing strategy that falls under the category of derived test oracles [21]. It is designed to assess whether or not the modifications are made to the new version of a SUT have disrupted existing functionality [239]. In particular, regression testing can be categorized into several testing techniques: Retest All, Regression Test Selection, Test Case Prioritization, and the Hybrid Approach [68]. Retest All is a conventional strategy that reruns all the test cases for the modified program, which can be very expensive and less cost-effective [118]. Regression Test Selection aims to reduce costs by selecting a subset of test cases to run, based on the changes made to the program [68]. Test Case Prioritization arranges test cases to increase the likelihood of early fault detection, based on factors such as recent changes, fault history, or code coverage, to improve the efficiency of the testing process [170]. The Hybrid Approach combines Regression Test Selection and Test Case Prioritization to leverage the benefits of both techniques, optimizing both the selection and execution order of test cases.

2.2 Metamorphic Testing

MT has emerged as a powerful technique to alleviate the oracle problem, using MRs to test software through the relations among multiple sets of inputs and their corresponding outputs [250]. MT has effectively identified faults in a wide variety of complex software systems, such as web services [192], embedded systems [39, 100], compilers [113], databases [125], machine-learning classifiers [103], online search functions and search engines [84], cybersecurity [48], and ADSs [83, 250, 254].

MT can be used both as a test case generation strategy and a test result verification method [47]. It offers several significant advantages. Firstly, it is conceptually simple and straightforward to implement, enabling testers with limited experience to learn and apply it effectively within a few hours [127]. The major steps in MT, including test case generation, execution, and result verification, are relatively easy to automate, while the MR generation process often requires manual effort and domain expertise [47]. Additionally, MT is cost-effective compared to traditional testing techniques, as it avoids the scalability issues [47], which refers to the number of test cases needed or the testing efforts required will grow exponentially when the size of the SUT increases [87]. MT can be applied with a test suite of any size, making it flexible and efficient even for large and complex programs [47].

2.2.1 Metamorphic Relations

MRs are the core part of the MT [45]. An *MR* is a property that specifies how the output of the program should change if the input changes correspondingly [242]. For instance, in an algorithm that computes the shortest path between two vertices in an undirected graph, an example MR would involve swapping the source and destination vertices, with the relation that the length of the shortest path remains the same [254].

The majority of MRs consist of two components: the input relation, which describes how inputs change among test cases; and the output relation, which describes how outputs change among test cases [148]. Additionally, there is another category of MRs where the construction of follow-up test cases requires the outputs of the source test case [123]. For instance, in MRs designed to test a search engine [256], the source test case must first be executed to obtain a list of search results based on certain keywords. The follow-up test case, using the same set of keywords but restricted to a specific domain, should return a sub-list of results from that domain.

A metamorphic group (MG) is a sequence of source inputs and follow-up inputs related to MRs [47]. Different MGs can be constructed under the same MR [133]. The effectiveness of MT relies on the quality of MRs and MGs [47]. Tian et al. [197] developed individual MGs that include an initial driving image and a follow-up image altered by weather and lighting changes. This method revealed thousands of erroneous behaviours in three deep-learning models used for outputting AD steering values based on static images. Luu et al. [133] introduced the Sequential MetAmoRphic Testing (SMART) framework, which used sequences of MGs of test cases to test the correctness of AV decisions. The study identified numerous undesirable behaviours in the AD models and highlighted key factors influencing their decisions. The results showed the effectiveness of using sequential MGs in revealing undesirable behaviours of SUTs [133].

2.2.1.1 MR Generation

The generation of MRs is one of the main challenges in MT and has thus become a significant research focus [123], which was also what *RQ2* of this thesis tried to address (*How can the difficulty of generating MRs be reduced to assist beginners and testers in understanding and testing new systems?*). Traditional approaches to MR generation are typically constrained to specific application domains and heavily reliant on human expertise [107, 242], and producing a sufficient number of high-quality MRs remains a challenging task [80]. Research has shown that the MRs can be produced either by deriving from existing ones or generating from scratch [123].

Composing new MRs from existing ones was one of the earliest research directions for MR generation [123]. Dong et al. [65] showed that the effectiveness of composite MRs in fault detection depends on the individual component MRs and the composition sequence. Liu et al. [128] developed a method to define the composability of MRs and showed that composite MRs could be more cost-effective than each component MR since they need fewer test cases. More recently, Qiu et al. [159] conducted a comprehensive theoretical and empirical exploration of the fault-detection effectiveness of composite MRs. They identified sufficient conditions under which composite MRs are more effective than component MRs. They also provided explanations and guidelines to help testers determine when to use composite MRs over individual component MRs.

When generating MRs from scratch, the category-choice framework [152] provides a structured method to create and organize test scenarios by grouping inputs into categories and selecting different options within each category, based on system specifications [157]. The category-choice framework, based on Category-Partition Method (CPM) [152] and CHOiCe-reLATion framework (CHOC'LATE) [157], organizes categories and choices derived from software specifications. A category represents a property of an input or environment condition, while choices are distinct subsets of the category's values. Test cases are generated by selecting values from complete test frames (CTFs), which are formed by combining choices from different categories [123]. Chen et al. [51] used the category-choice framework [50] to propose a *METRIC* technique to identify MRs based on the program specifications. *METRIC* involves selecting two distinct test scenarios, and evaluating if there is a definite relationship between their outputs. If so, the tester can generate MRs based on this relationship. This process is repeated until a set number of MRs have been created, or all relevant scenarios have been exhausted [51]. One advantage of this approach is that it does not require testers to access the internal code of the SUT, as it focuses on testing based on the program's functional specifications [144]. Sun et al. [189] developed *METRIC*$^{+}$, an extension of the original *METRIC* technique, by incorporating both input and output domains of the SUT. *METRIC*$^{+}$ presents new definitions for input categories and choices as I-categories and I-choices, and for output categories and choices as O-categories and O-choices, along with combined input-output test frames (IO-CTFs). This approach facilitates the analysis of output relationships, and reduces the need for manual evaluation [189]. Since the category-choice framework can be applied to any type of software where categories and choices can be defined, *METRIC* and *METRIC*$^{+}$ have broad applicability, and are not limited to specific types of software. However, this flexibility also presents a potential limitation: some companies may not release detailed specifications of products for external use, thereby restricting the practical application of these methods [88].

Typically, MRs are specified at a high level of abstraction, to the point where they may not even represent a single relation but rather a collection of them [176]. An MRP is an abstraction describing such a collection of MRs (which could be unlimited in size [255]). The concept of MRPs can be further extended to two subclasses of patterns: MRIPs [255] and MROPs [176]. MRIPs are abstractions that define the relationships between the source and follow-up inputs within a set of MRs [174]. On the other hand, MROPs describe relations between the source and follow-up outputs. Segura et al. [176] first introduced the concept of MROP for RESTful Web APIs, defining them as abstract output relations typically identified in Web APIs. They identified six MROPs based on set operations: equivalence, equality, subset, disjoint, complete, and difference. They also worked on the inference of likely MRs for model transformations by identifying domain-independent trace patterns and defining MRs that could be adapted to specific contexts [202].

Formal definitions of MRP and MRIP were produced by Zhou et al. [255]. They first showed symmetry as a universal property with examples from multiple areas, and then defined the symmetry *MRP* and "change direction" *MRIP*, with a series of case studies for popular applications in multiple areas. The results showed that the MRPs and MRIPs proposed in the study were simple and straightforward to use [255]. MRPs play a crucial role in facilitating the systematic generation of MRs, and by utilizing MRPs, the process of generating MRs can become more structured and streamlined [123]. Due to the diverse applications and complexities of SUTs, finding universally applicable MRPs is challenging. Recent studies have focused on domain-specific MRPs, such as Liu et al. [129] defining a symmetry MRP for fuzz testing deep learning libraries, and Sun et al. [187] proposed 14 interleaving patterns for concurrent programs. Recently, Zhang et al. [249] proposed three MRPs and MRIPs for testing AD simulators, which were proven effective in generating MRs and finding issues (Section 5.2.1).

In 2014, Zhang et al. [242] proposed a search-based approach that can automatically infer "polynomial MRs" for SUTs, where the input relations follow simple linear expressions between the source and follow-up inputs, and the output relations are either linear or quadratic based on the source and follow-up outputs. In the study, the inference of MRs was viewed as a search problem so a set of parameters was used to represent a particular class of MRs, and Particle Swarm Optimization (PSO) [106] was adopted to optimize these parameters [242]. They evaluated the inferred MRs on four scientific libraries, and showed the effectiveness of the MRs in detecting faults [242]. One of the strengths of this method is that it does not require the tester to understand the code, i.e., it is a black-box testing approach [145]. Although the original study used the C programming language, it can also be extended to multiple languages. However, one limitation of its usage is that it currently only supports numerical systems. Zhang et al. [241] further introduced AutoMR, an extension of their PSO-based approach [242], which generates diverse MRs involving more than two inputs and includes both equality and inequality relations. AutoMR also supports eliminating redundant MRs and employs a more fine-grained fitness function to improve the accuracy and simplicity of MRs [241].

MRs can also be generated using AI approaches. Kanewala and Bieman [103] first used ML techniques for MR generation with an approach involving the construction of control flow graphs (CFGs), extraction of features for creating predictive models, and use of these models to predict likely MRs. They applied decision trees (DTs) and support vector machines (SVMs) to predict MRs for 48 scientific numeric functions, finding SVM-predicted MRs superior in killing mutants of functions (modified versions of the SUTs with artificial defects [99]). Additionally, Kanewala [101] proposed using program dependency graphs (PDGs) and graph kernels (random walk and graphlet kernels) to improve MR prediction. Further studies by Kanewala et al. [102] demonstrated that graph kernels enhanced the accuracy of MR prediction, with control flow information proving more effective than data dependency information. The research by Rahman and Kanewala [102], and Nair et al. [143] also supported the applicability and effectiveness of these methods. Zhang et al. [245] introduced a neural network-based approach using CFGs to create a multi-label dataset and employed radial basis function (RBF) neural networks [244] for MR prediction. Other AI-based approaches include program-documentation-based methods [160] and semi-supervised SVM-bagging KNN algorithms [235]. A recent study [247] used ChatGPT (GPT-3.5) to automatically generate MRs for the parking module of an ADS, finding that the quality of generated MRs varied based on system complexity, test scenario specificity, and the number of MRs requested, necessitating human intervention for validation and quality improvement (Section [247]).

Other techniques for generating MRs systematically include using genetic programming [86, 224], tabular expression [122], behaviour driven [58], guideline-based [163], data mutation [190, 191], and template-based [177].

2.2.1.2 MR Selection

Selecting effective MRs could help reduce the number of test cases and enhance defect detection [47]. Research has been conducted to develop guidelines for the selection of MRs [12, 35, 46, 127, 137]. Cao et al. [35] established a strong correlation between the fault-detection effectiveness of MRs and the dissimilarity in the execution profiles generated by source and follow-up test cases. They showed that MRs that lead to greater differences in execution paths tend to be more effective at uncovering faults. In another research, Asrafi et al. [12] conducted an empirical study showing that MRs with low code coverage are less effective at detecting software faults, while high code coverage, indicating diverse execution behaviour, correlates well with fault detection effectiveness, even though it is not a perfect indicator of this effectiveness. Mayer and Guderlei [137] proposed four general rules for quickly assessing MRs based on their suitability for fault detection. The study involved identifying various MRs applied to several common Java programs of determinant computation. The study suggests that systematic criteria can help identify and select effective MRs, thereby improving the efficiency of MT [137].

Although multiple guidelines have been proposed for selecting effective MRs, they are not quantitative and the implementation is rather subjective, according to

Chen et al. [47]. More research is needed to establish formal, objective criteria [47, 175]. To alleviate this challenge, this PhD project involves developing a formal and objective evaluation framework to assist testers in selecting and evaluating MRs (Section 4.3.2).

2.2.2 Test Case Generation for MT

Effective test case generation is essential in MT [127]. Notably, MT was proposed both as a test case generation and test result verification approach [47]. Lv et al. [134] proposed a novel approach for multiple-path test case generation by combining the PSO algorithm with MRs. In this approach, the first test case is generated by the PSO algorithm and then MRs are used iteratively to generate other test cases. This method improves efficiency in generating multiple path-wise test cases and reduces computational costs [134]. Chen et al. [42] proposed the Equivalence-Class Coverage for Every MR (ECCEM) criterion, which can be applied to generate fewer test cases while achieving a high fault-detection rate. ECCEM uses equivalence classes to generate efficient test cases with a strong coverage of MRs.

Traditionally, the source test cases (STCs) in MRs are usually generated with the Random Testing (RT) method (a methodology that systematically generates random inputs for a program [82]) [90, 221]. However, one limitation of this approach is that it does not use any existing information to guide testing [49]. Adaptive Random Testing (ART) [38] was proposed to enhance the testing efficiency of RT by maximizing the distribution of test cases across the input domain [90]. Experiments showed that ART has better performance than RT on test case selection, further improving the performance of MT [22]. Hui and Huang [89] introduced the metamorphic-distance ART (MD-ART) algorithm to guide the selection of the STCs, which is based on both the input diversity (as in ART) and their relationship to previous test cases (using MRs). This approach improves the performance of MT and ART in terms of test effectiveness, efficiency, and test coverage [89]. However, in this study, only the distance between the STCs and the selected test cases was considered, while the effect of the follow-up test cases (FTCs) on the next test case selection was not used [90]. To take advantage of the FTCs, Hui et al. [90] proposed MT-based ART (MT-ART) to optimize test case selection by considering three types of distances during test case selection: the distance between FTCs and executed test cases, between STCs and executed test cases, and between FTCs and STCs. The experiments showed that MT-ART generated more effective test cases compared to MD-ART and RT [90]. Recently, Ying et al. [237] introduced the MT-based ART through Partitioning (MT-PART) algorithms to improve both the effectiveness and the efficiency of MT by dynamically partitioning the input domain and generating new STCs and FTCs in a way to ensure an even distribution across their corresponding input domains. This approach has significantly enhanced the efficiency of testing compared to other MG generation algorithms (e.g., MT-ART [90]) while maintaining good effectiveness [237]. In another research, Ying et al. [236] introduced MT-based ART applied to Source and Follow-up Input Domains (SFIDMT-ART) algorithms for the generation of high-quality MGs that improve the diversity of MGs.

Other approaches for test case generation include using genetic algorithms [24], feedback-directed approach [188], iterative MT (IMT) [220], and equivalance-class coverage method [74]. This PhD project introduces a novel test scenario generation template for MT in ADS testing [251], which improves the accessibility and user-friendliness of the test case generation process for a broader audience (Section 6.2.1), also as a solution to address *RQ3* of the thesis (*How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?*).

2.2.3 Metamorphic Exploration

ME [255] is an extension of MT. While MT focuses on testing the correctness of software with MRs, ME uses the concept of HMRs. These HMRs are believed to hold true based on the tester's understanding of the system [257]. Unlike MRs in traditional MT, where a violation typically indicates a defect in the software, a violation of an HMR does not necessarily point to a software problem. Instead, it may reveal gaps or inaccuracies in the user's understanding of the system [255].

When an HMR is violated, it prompts further investigation. This process often leads users to a deeper exploration of the system, allowing them to enhance their understanding of the system [231]. ME not only helps identify potential software defects, but also enhances the user's comprehension of the system, potentially leading to more effective and accurate use of the software [232, 255]. This exploratory approach is particularly valuable in complex systems where the behaviour may not be entirely predictable or understood, or the specifications are missing or unavailable to users [255]. By hypothesizing and testing various HMRs, users can incrementally build a more comprehensive understanding of the system's functionality and limitations.

ME has been applied in multiple areas, including ML [231, 232], education [200], and object-detection [257]. Yang, Towey, and Zhou [231] used ME for a clustering program, where they identified seven HMRs and detected two HMR violations. These findings provided insights into the system that were not documented in the software's user manual, but were revealed through ME. Zhou et al. [257] conducted an empirical study on the Camera Obstacle Detector (COD) of the Baidu Apollo ADS, proposing an *in-place* approach with three HMRs for programs that process data streams. The empirical results demonstrated that this approach is practical and effective in detecting a large number of real-time obstacle-perception failures, with a relatively low false-alarm rate. Zhang et al. [246] applied ME to the perception and localization modules of the Apollo ADS, which uncovered an apparent anomaly that was later identified as a misunderstanding rather than a fault in the system (Section 3.2). The experience enhanced the researcher's understanding and testing capability of the ADS, which laid a foundation for their future testing work [250]. Additionally, the approach used in the study can also benefit others testing ADSs and have broader implications through integrating ME into SQA training to help learners better understand complex systems [246].

2.3 Autonomous Driving Systems

ADSs, or self-driving, have gone through fast development over the past decade. The levels of self-driving, defined by the Society of Automotive Engineers (SAE), range from Level 0 to Level 5 [19]. Level 0 means no automation, with the driver responsible for all tasks. Level 1 offers basic driver assistance. Level 2, or partial automation, allows the car to control both steering and speed, but the driver must stay attentive. Level 3, or conditional automation, enables the vehicle to handle most of the driving tasks, but the driver must still be present and ready to take over. Level 4, or high automation, performs all driving tasks without driver input, in most environments. Level 5 represents full automation, with the vehicle capable of driving entirely on its own in any situation.

Numerous companies are working on ADSs, such as Waymo [193], NIO [146], Tesla [14], and XPENG [227]. Their interests range from hardware manufacturing and computing to software development for driving assistance, entertainment, and in-car advertisement [19]. The primary components of an ADS include sensors, perception systems, decision-making algorithms, and control mechanisms [240]. Sensors, including LiDAR, radar, cameras, ultrasonic sensors, and GPS, collect large amounts of data about the vehicle's surroundings [131]. These sensors provide essential information on the distance, speed, and characteristics of nearby objects and the vehicle's precise location.

The perception system processes the raw data from the sensors to create a comprehensive understanding of the environment [147]. This involves object detection and classification, which identifies other vehicles, pedestrians, and road signs, as well as localization, which determines the vehicle's position. Additionally, environmental mapping constructs a dynamic map of the surroundings to facilitate safe navigation and obstacle avoidance. Decision-making algorithms then interpret this processed data to make real-time driving decisions [147]. Path planning algorithms generate a safe and efficient route, while behaviour planning decides on specific manoeuvres, such as lane changes and stops, based on current traffic conditions [61]. Predictive algorithms anticipate the actions of other road users to avoid potential collisions [147].

Control mechanisms execute the decisions made by the planning algorithms [131]. These include motion control systems that manage acceleration, braking, and steering to follow the planned route precisely. Feedback control systems continuously monitor the vehicle's performance and make necessary adjustments to maintain stability and accuracy. To ensure seamless operation, ADSs often incorporate Vehicle-to-Everything (V2X) communication technologies, allowing vehicles to exchange information with each other and other infrastructures, enhancing safety and efficiency [147].

The current ADSs offer a variety of functions designed to enhance safety, convenience, and efficiency for drivers and passengers. These functions typically include adaptive cruise control (ACC) [225], which maintains a set speed and safe distance from the vehicle ahead; lane-keeping assistance (LKA) [207], which helps the vehicle stay centred in its lane; and automated parking systems that assist in navigating into parking spaces [210]. More advanced features, such as traffic jam assist [252] and highway pilot [120], enable semi-autonomous driving in certain environments by combining multiple sensors and algorithms to navigate through traffic and along highways.

The complex nature of ADSs often makes it difficult to create fully automated test oracles for testing driverless vehicles, while manual monitoring can be both expensive and imprecise [254]. Furthermore, generating comprehensive system specifications to evaluate system behaviours faces challenges in addressing the oracle problem, which involves replicating the decision-making processes of human drivers [197]. For instance, a human tester may struggle to assess whether the route selected by an ADS in a complex road network is optimal or not [30]. This kind of oracle problem also extends to ensuring the validity of simulation data, which is crucial for the accuracy of ADS development [167]. MT has proven to be an effective approach for addressing the oracle problem by using MRs [47]. Consequently, to address *RQ1* of this thesis (*How can MT be effectively applied in ADS testing to uncover anomalies and enhance system understanding?*), this PhD research has implemented MT across various ADSs and AD simulators, demonstrating its effectiveness with multiple defects revealed (Chapters 3 and 5).

2.3.1 Existing Open-Source ADS Platforms

2.3.1.1 Autoware

Autoware is an open-source software stack designed specifically for AD by Autoware Foundation [105]. Autoware is based on the Robot Operating System (ROS) [169] and designed for vehicles operating in various environments [105]. It includes modules for sensing, localization, perception, planning, and control. Autoware supports multiple applications such as autonomous valet parking, cargo delivery, and robo-taxis [36]. It focuses on modular architecture, allowing for easy integration with different hardware and sensor configurations. It also follows strict safety standards, targeting dependable real-world implementation and widespread usage in academic research and industrial projects [16]. Autoware's development has led to a two-part structure: Autoware Core and

Autoware Universe [15]. Autoware Core is a stable version of the system, offering essential functionalities for real-world AD applications with high safety standards. It includes thoroughly tested and well-maintained modules [15]. In contrast, Autoware Universe promotes innovation and experimentation, integrating advanced, community-contributed features for rapid development and diverse applications [15].

Autoware has several limitations [53], including the need for extensive calibration and tuning to achieve optimal performance across different hardware setups. Integrating multiple sensors and ensuring seamless communication between modules can also pose significant technical challenges. Additionally, Autoware's complexity requires expertise to deploy and maintain, which can be a barrier for some users [53].

2.3.1.2 OpenPilot by Comma.ai

OpenPilot, developed by Comma.ai, is an open-source advanced driver-assistance system (ADAS) designed to enhance the self-driving capabilities of existing vehicles [43]. OpenPilot integrates with supported car models through a certain device, using AI-based vision models and car sensors to provide functionalities such as ACC, LKA, and automatic lane changing (ALC) [20]. It has provided support for more than 150 types of vehicles. Despite its capabilities, it has some limitations, such as the absence of features like pedestrian detection and automatic parking [6]. Additionally, installation and maintenance can be challenging for regular users [6]. The system performance can also be affected in severe weather conditions due to reliance on image recognition [6].

2.3.1.3 The Baidu Apollo ADS

Baidu is developing a series of AD programs for its open-source self-driving car project Apollo [71]. These programs involve manufactured vehicles that aimed to reach the L4 Autopilot level, for instance, *Robotaxi* [104] and *Minibus* [11]; and an in-car operating system that enhances the human interaction with cars called *DuerOS* [67]. The commercial version of Apollo has been successfully deployed in China, providing an open platform that collaborates with many automotive and technology partners. Baidu has launched autonomous taxi services in cities like Beijing and Wuhan, allowing residents to book rides through a mobile app [104].

The Apollo ADS has evolved into the 9th generation since 2017 with powerful modules and functions [71]. The platform offers a range of tools and resources essential for building ADSs, including HD maps, simulation environments, and an array of software development kits (SDKs) [8]. Apollo's modular architecture enables developers to integrate its components into their own systems or use them as standalone solutions. An important aspect of Apollo's impact is its commitment to open-source principles [8]. By promoting an open and collaborative development environment on GitHub, Apollo facilitates the sharing of insights and technological advancements among developers and researchers [8].

This PhD project chose Apollo as the testing target because Apollo is a complex and powerful open-source ADS with comprehensive documentation support. Furthermore, Apollo offers a variety of programs and tools that enable developers and testers to experiment with the system on their own devices, eliminating the need to deploy on an actual vehicle. Nonetheless, a potential drawback is the slow feedback loop between developers and users on the Internet (GitHub [8]).

2.3.2 Applications of MT in the Domain of ADSs

MT has been applied to test multiple kinds of automated systems and obtained valuable results [7, 17, 58–60, 81, 83, 94, 95, 126, 133, 197, 205, 226, 233, 250, 254]. For instance, researchers from the Fraunhofer Center created a simulation environment to test the control software of UAVs (unmanned aerial vehicles) using MT [126]. Their project began by developing a model based on flying scenarios and studying the routes selected by the drone. Their MRs required the drone to behave consistently regardless of the angle at which the scene was rotated. The testing identified corner cases where the drone would fly directly towards obstacles, resulting in crashes [126].

Researchers from the University of Virginia and Columbia University tested the decisions of ADSs with the tool developed by deep neural networks (DNNs) [197]. The tool, *DeepTest*, can automatically generate synthetic images by applying transformations such as scaling, shearing, rotation, and various filters to create test cases. They used MT to test the steering angle outputs of the ADS models. The MRs require that under different conditions of the same input, the steering angle values should be similar [197]. A large number of MR violation cases were found when the SUT performed dangerous erroneous behaviours, exposing the design defects.

However, as was pointed out by Zhang et al. [243], the test cases generated by *DeepTest* might not accurately represent real-world scenarios. For instance, the sidelines of roads were missing in the synthetic images, and the blurring effects were unrealistic, reducing the reliability of the result. To solve the issues, the team developed an unsupervised learning framework, *DeepRoad*, that contains an MT module $DeepRoad_{MT}$ [243]. This module can generate driving scenes with different kinds of weather conditions, and perform MT for DNN-based ADSs. The MRs require the behaviours of cars to be consistent under the synthesized driving scenes with different weather conditions. Three popular ADS models in

Udacity self-driving car challenge [204] were tested and thousands of inconsistent behaviours were detected [243].

Deng et al. [60] developed the declarative Rule-based MT (RMT) framework, using natural language syntax to specify real-world traffic scenarios, effectively identifying previously undetected safety issues in ADS models. They further enhanced RMT to include higher-order and compositional MRs, detecting a significant number of abnormal ADS model decisions [60]. In another study, Deng et al. [58] created the Behaviour Driven Development-based MT (BMT) framework, using human-written behaviour definitions to generate test inputs, which detected numerous erroneous predictions in three driving models (related to speed predictions) that were confirmed as traffic violations. Ao and Pan [7] introduced Scenario-based MT (SMET), demonstrating that complex scenarios are more effective at detecting defects. Xiong et al. [226] proposed Inequality-Based MT (IEMT), efficiently detecting inconsistent behaviours in AD neural network models.

Underwood et al. [205] developed an MT framework for modular ADSs, incorporating hypothesis testing to handle non-deterministic behaviours. This framework identified numerous consistencies and reliability issues in Autoware ADS. Ayerdi et al. [17] proposed MarMot, which uses domain-specific MRs to monitor ADS behaviours at runtime, outperforming other monitoring approaches by identifying a large number of external anomalies (e.g., environmental changes like fog) and internal anomalies (e.g., issues in deep neural networks caused by mislabeled training data). Luu et al. [133] developed a sequential MT framework that uses sequences of MGs to understand ADS behaviours without needing ground-truth datasets, revealing undesirable behaviours in top-ranked ADS models. Yang et al. [233] introduced MetaLiDAR, an automated testing methodology for LiDAR-based systems, which detected inconsistent behaviours in object-detection models, enhancing their robustness. Iqbal [94] applied a simulation-based MT approach using Euro NCAP standards to test the emergency brake function of the ADAS system. The method identified 5.8% of the test cases in which the system did not respond correctly and led to a collision. The findings highlight the effectiveness of MT in revealing hidden errors in ADAS systems and combining simulation-based testing with MRs to design more reliable systems.

Due to the open-source nature of the Baidu Apollo ADS, several studies have applied MT on this platform. For instance, Zhou and Sun [254] used MT to find defects in the LiDAR obstacle-perception (LOP) module. Specifically, they tested the ability of the LOP to detect the obstacles by using the MRs that the number of obstacles identified by the LOP should be a subset of the obstacles identified when random LiDAR data points were scattered outside the area of detection (i.e., Region of Interest (ROI)). The results showed that there was a 2.7% probability for the car to fail to detect the obstacles on the roadway if the number of random points scattered outside the ROI was 10, and 33.5% probability when the number increased to 1000. Eight days after the team reported the issue to the Baidu Apollo self-driving car team, a real-life traffic accident occurred in which a vehicle, using the same type of sensor unit in its perception module as the SUT in the study, collided with a pedestrian [119].

In more recent studies, Han and Zhou [83] introduced a novel approach combining MT with fuzz testing to identify fatal errors in the Apollo ADS. Their research focused on creating varied test cases through MRs and integrating these with fuzz testing to simulate unpredictable real-world scenarios. This combined methodology detected critical failures, particularly in the software's decision-making processes, that might not be revealed by the traditional testing techniques. This approach demonstrated that by generating diverse and unexpected inputs for testing, the robustness and safety of ADSs could be significantly improved. Zhou et al. [257] proposed an in-place MT method for testing video streams in the Baidu Apollo's camera perception module. This method identified previously unknown perception failures, including both undetected and incorrectly detected objects.

Complementing these works, Guo, Feng, and Chen [81] focused on the LiDAR obstacle perception system in Baidu Apollo. They employed MT alongside fuzzing techniques to enhance the testing of LiDAR point clouds, which are critical for the vehicle's environment perception. Their study introduced *LiRTest*, a framework designed to automatically generate and modify LiDAR point clouds to simulate various real-world conditions. This allowed for the identification of unknown fatal errors in the LiDAR-based perception algorithms. The innovative approach of integrating fuzzing with MT provided a more comprehensive evaluation of the system's robustness, particularly in detecting and handling corner cases that might not be covered by traditional testing methods. More recently, Zhang et al. [250] performed MT on the obstacle identification algorithms of Apollo's perception-camera module (Section 3.3), an area previously untested by MT, contributing to the field of MT in ADS testing. The experiments revealed conflicting obstacle-detection results both in individual and sequential frames when raising the brightness of a specific part of the input images. This study also developed an MT harness with an industry case study to facilitate the testing of ADSs, which would increase efficiency and help testers better organize the testing procedure [250].

2.3.3 The Role of Autonomous Driving Simulators in ADS Testing

The development of ADSs involves many challenges, including the integration of complex sensors and advanced decision-making algorithms [41]. In this environment, simulators such as CARLA [66] and SVL [168] have become important for the designing and testing of these advanced technologies.

CARLA is a popular tool among AD simulators [66]. Its flexible and customizable platform allows researchers, engineers, and developers to experiment with and refine various aspects of autonomous vehicle technology in a virtual setting. The simulator can simulate a wide range of real-world situations, including different weather conditions, detailed urban environments, and complex traffic patterns [66]. This broad range of features provides a controlled but dynamic setting for testing the effectiveness of algorithms, the integration of sensors, and the robustness of decision-making processes. Importantly, this testing can take place without the risks associated with real-world trials [240].

The SVL simulator is a Unity-based multi-robot simulator [168]. It provides the ability to work with Apollo [71] and Autoware [105] platforms by having a communication bridge to transfer messages between the simulator and these ADSs [168]. In addition, the software supports simulation of the environment, sensor, control and vehicle dynamics with realistic 3D animations and detailed data. Users can switch maps and vehicle models to test the outputs of the ADSs under multiple scenarios. However, the SVL simulator was officially unsupported by LG in 2021, resulting in a lack of compatibility with newer ADS updates and increased usability challenges [194].

AD simulators are significant and necessary in the ADS development [124]. They enable testers to perform comprehensive testing more quickly and economically than physical road tests [33]. For example, simulators can safely rebuild dangerous scenarios including severe weather or complex interactions between multiple vehicles, allowing these scenarios to be tested repeatedly. This capability greatly accelerates the development process by facilitating early identification and resolution of potential issues, thus reducing the likelihood of these problems arising in actual road situations [124]. Moreover, the data generated from these simulators is a valuable resource for training ML models [124]. By integrating synthetic data from simulators with real-world data, the adaptability and effectiveness of autonomous systems can be significantly improved.

Ensuring the validity of the simulation data is critical in the accuracy of ADS developments [223]. To be effective in testing and development, a simulator must be accurate in replicating real-world conditions, making validity a critical criterion for assessment [223]. This refers to the capability of a driving simulator to accurately depict real-world driving conditions. Two forms of validity that most studies assess are absolute validity and relative validity [141]. The concept of absolute validity refers to how closely values obtained from a simulator, such as speed or lateral position, match those from an actual vehicle, without the need for any relative or proportional adjustments [141]. Relative validity serves as an alternative criterion when the outcomes or impacts observed in simulated driving closely mirror those experienced in real-world driving. Although achieving absolute validity is the ultimate goal, it is often unattainable due to challenges like the oracle problem [141]. Consequently, in specific contexts, pursuing relative validity can be considered an acceptable approach [223].

2.3.4 MT for Validating Simulation Software

MT has emerged as an effective approach for ensuring the reliability and accuracy of simulation models. Ahlgren et al. [4] discussed the deployment of the *Metamorphic Interaction Automaton (MIA)*, an MT system for Facebook, to test Facebook's large-scale, web-based simulation infrastructure. MIA addresses issues of test flakiness (inconsistent test results) and the oracle problem by automating continuous integration and regression testing, enhancing the reliability of simulations in large-scale web systems.

Olsen and Raunak [149] extended the application of MT to simulation validation. They proposed a framework for validating simulation models using MRs between parameters and behaviours, which enhanced the validity of simulation models by detecting errors and inconsistencies in behaviours that might be missed with traditional testing methods. This approach addresses the challenges of the oracle problem in simulation validation, thereby improving the trustworthiness of simulation results.

Adigun, Eisele, and Felderer [3] explored specific methodologies for applying MT to simulate AD scenarios. Their research outlined detailed procedures for creating MRs tailored to the unique behaviours and decision-making processes of autonomous drones. By focusing on realistic traffic conditions and edge cases, they demonstrated how MT could uncover flaws in the vehicle's perception and control algorithms. Additionally, Iqbal et al. [95] applied MT to AD simulation platforms, specifically targeting the LKA system. Their study involved creating a set of MRs that accurately reflect the expected behaviours of the LKA system under various driving conditions. By systematically varying parameters such as road curvature, vehicle speed, and environmental factors, they were able to generate test cases that simulate a wide range of scenarios. Their results showed that MT could effectively identify faults related to lane detection, steering control, and system responsiveness.

In a recent study, Zhang et al. [249] applied MT to the NIO AD simulator, revealing various issues with its algorithms and performance (Section 5.2). MT revealed a significant portion of issues, most of which were not detected by conventional methods. The majority of these issues were related to the compilation of scenario elements and the underlying logic of the simulation system, and were classified as high priority, highlighting the effectiveness of MT in uncovering critical AD simulator defects. The team later applied MT to test the CARLA simulator[1] (Section 5.3), revealing critical defects related to its built-in functions and performance. Both studies enhance the application of MT in ADS testing, addressing *RQ1* of the thesis.

[1]This work was in the review process for the Information and Software Technology (IST) Journal.

2.4 Integrating MT with Artificial Intelligence

Integrating AI into software testing has become a trend with the rapid technology developments [208]. Manual testing is often time-consuming, expensive, prone to human error, and may not cover all test cases and scenarios [96]. Integrating AI technologies, such as ML, in software testing can enhance efficiency, accuracy, and effectiveness, addressing existing drawbacks of manual testing. Among the approaches, LLMs have demonstrated strong potential in advancing MT due to their reasoning abilities and capability to generate human-like contexts, as MT requires reasoning about the essential properties of a SUT [203].

2.4.1 Overview of Large Language Models

LLMs are a major advancement in the field of natural language processing (NLP). These models are designed to understand and generate human language, making them useful for a wide range of applications [253]. Examples of well-known LLMs include OpenAI's GPT (Generative Pre-trained Transformer) series [213], Google's BERT (Bidirectional Encoder Representations from Transformers) [63], and Meta's LLaMA (Large Language Model Meta AI) [93].

To create an LLM, researchers start by training the model on a massive amount of text from books, articles, websites, and other sources [253]. This process, called pre-training, helps the model to learn the basics of the language, such as grammar, factual information, and contextual word usage [253]. During pre-training, the model learns to predict missing words in sentences, which helps it to understand how words fit together.

After pre-training, LLMs are fine-tuned for specific tasks using smaller, focused datasets [253]. This fine-tuning process allows the models to perform well on particular applications like answering questions, summarizing text, or translating languages [213]. Because of their versatility, LLMs are used in many fields, including healthcare, finance, education, and customer service [253].

2.4.2 ChatGPT

This PhD project focused on ChatGPT [91]. ChatGPT, an LLM developed by OpenAI, has gathered significant attention for its wide-ranging capabilities [247]. Empirical studies have documented its proficiency in a variety of tasks, such as mathematical and logical problems [73] and bioinformatics research [183]. In software engineering, evidence suggests that ChatGPT can generate source code, formulate test cases, and aid in the debugging of programs [121]. However, critical assessments in recent literature have highlighted limitations and failures in ChatGPT's performance [213]. The model's effectiveness in executing complex software engineering tasks, which often require advanced cognitive capabilities, remains a subject of ongoing evaluation and debate.

GPT-3.5 [153] and GPT-4 [150, 151] belong to the current commercial versions of OpenAI's GPT series, each marking significant advancements in AI language processing [92]. GPT-3.5, as the free version of ChatGPT [153], has equipped with powerful language understanding and generation capabilities. GPT-4, a more sophisticated and larger model, further refined these capabilities, leading to a more detailed and contextually aware conversational experience in the latest iterations of ChatGPT [151]. The evolution from GPT-3.5 to GPT-4 represents a leap in the ability to create AI that can interact in human-like ways, with ChatGPT showcasing the practical application of these advancements in fields ranging from customer service to education, offering refined, accurate, and context-sensitive interactions [151].

This project investigated ChatGPT's capability to generate MRs for typical ADS functions, and compared them with MRs created by students using various evaluation criteria. The detailed findings are outlined in Chapter 4.

2.4.3 Applications of MT in LLMs

While LLMs are good at handling natural language tasks, there is limited research on their application in MR generation. Shin et al. [181] proposed an approach that uses a few-shot prompting strategy, where the model was provided with only a few examples, to instruct LLMs to derive executable MRs (EMRs) from provided requirements and API specifications. The approach involves setting the context, identifying relevant sentences in the requirements, rewriting them as MRs, and then converting these MRs into EMRs using a domain-specific language (DSL) called SMRL [181]. The feasibility of the approach was evaluated through a questionnaire-based survey in collaboration with an industry partner, focusing on four of their software applications, and the accuracy of EMRs generated for a web application was also assessed. The findings indicate that this method can produce understandable and relevant MRs and EMRs, highlighting its potential to automate the MT process. Tsigkanos et al [203] explored the use of autoregressive transformer-based LLMs to automate the extraction of variables from software documentation, a critical step in MT that is usually labour-intensive. They proposed a workflow incorporating LLMs that achieved 0.87 accuracy in variable identification, with many derived as exact or partial matches[203]. This suggests that LLMs can effectively streamline and scale the variable extraction process, thus improving the efficiency of MT.

In Chapter 4, experiments were conducted involving using ChatGPT (GPT-3.5 [247]) to automatically generate MRs for the parking module of an ADS, which were then evaluated by domain experts. The results showed that ChatGPT was effective in generating MRs, but the quality of the MRs varied based on system complexity, test scenario specificity, and the number of MRs requested, necessitating human intervention for validation and quality improvement [247].

In the subsequent experiments[2], the performance of GPT-3.5 and GPT-4 were compared, and the evaluations were extended to a broader scope of SUTs, ranging from basic computational programs to large and complex systems with and without AI integrations. Additionally, another GPT was configured to perform the evaluations along with the human evaluators. The findings revealed GPT-4 generated improved MRs, although weaknesses such as novelty remained. The configured GPT evaluator also showed promise in improving efficiency in MR selection and refinements. These findings further led to the development of a GPT-MR generator in Section 5.3.1, which the MRs derived were directly used in testing, and revaling several critical defects of an AD simulator. Additionally, Section 6.3 compared MRs produced by GPT and students, illustrating that beginners, with appropriate guidance, could generate MRs of quality comparable to those created by LLMs. The findings also underscored areas for improvement in the teaching and training of both humans and LLMs. In summary, these explorations of LLM applications have demonstrated their potential to advance MT practices and lower barriers for beginners, addressing *RQ2* and *RQ3* of the thesis.

[2]The works below were in the review process for the Information and Software Technology (IST) Journal.

Chapter 3

Improving Autonomous Driving System Reliability with Metamorphic Testing

Publications related to this chapter

1. **Y. Zhang**, M. Pike, D. Towey, J. C. Han and Z. Q. Zhou, "Preparing future SQA professionals: An experience report of metamorphic exploration of an autonomous driving system," in *2022 IEEE Global Engineering Education Conference (EDUCON)*, Tunis, Tunisia, 2022, pp. 2121-2126, doi: 10.1109/EDUCON52537.2022.9766791.

2. **Y. Zhang**, D. Towey, M. Pike, J. C. Han, G. Zhou, C. Yin, Q. Wang, and C. Xie, "Metamorphic testing harness for the Baidu Apollo perception-camera module," in *2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)*, Melbourne, Australia, 2023, pp. 9-16, doi: 10.1109/MET59151.2023.00009.

3.1 Introduction

To address *RQ1* (*How can MT be effectively applied in ADS testing to uncover anomalies and enhance system understanding?*) outlined in Section 1.2, this chapter presents a detailed analysis of the application of MT for the testing of ADSs, with a specific focus on experiments conducted on the Baidu Apollo ADS. The experiments demonstrated the efficacy of MT in uncovering anomalies, enhancing system understanding, and addressing specific testing challenges.

While conducting MT/ME on Apollo [71], an anomaly was detected through the violation of the identified HMR [246]. This anomaly was further investigated, with a project stakeholder eventually identifying it to be a misunderstanding of the system, and the related testing implementation. The insights gained from ME not only contribute to a deeper understanding of the ADS but also suggest the potential use of ME experiences in teaching and training Software Quality Assurance (SQA) professionals. More importantly, the findings from ME serve as a foundation for the later MT experiments.

To further explore the impact of MT on testing ADSs [250], the author conducted experiments on the object-detection algorithm of Apollo's perception-camera module [161]. The perception-camera module [10] is one of the core components of the Apollo ADS. As one of the most important sensors in the ADS, the camera has played an essential role in obstacle and traffic light detection [147]. It is an important supplement to the LiDAR perception results [10]. However, the camera has shortcomings, such as being easily affected by the environment and lacking depth information, which brings great challenges to the visual perception algorithm in the unmanned driving system [10]. Therefore, how to establish a set of high-precision and high-stability visual perception algorithms is the core issue of the unmanned vehicle perception module. The visual perception algorithm has three main application scenarios on the Apollo platform, namely: traffic light detection; lane line detection; and camera-based obstacle detection [71].

The MT experiments were performed on Apollo's perception-camera module that used the YOLO (You Only Look Once) model[1] [164]. Compared to other object-detection models, the YOLO model has the advantages of high speed and an easy-to-implement feature. The experiments revealed conflicting obstacle-detection results while raising the brightness of a specific part of the driving scenarios, both in individual and sequential frames, demonstrating the ability of MT to address the oracle problem when validating the perception module of ADSs [250]. Furthermore, the MT practice involved developing an MT harness to facilitate the testing of ADSs, which would increase efficiency and help testers to better organize the testing procedure, with an industry case study showcasing the harness's application during production phases.

Through these experiments, the chapter underscores the critical role of MT/ME in improving the reliability and robustness of ADSs, demonstrating how MT can be effectively applied to uncover hidden issues and enhance overall system performance and addressing *RQ3* of the thesis: *How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?*

3.2 Exploring Baidu Apollo ADS through ME

3.2.1 Experiments

The experiments in this section [246] report the ME experience that used data from the Apollo ADS perception-obstacle and localization modules: The perception-obstacle module provides NPC vehicle (the one not controlled by the ADS) information (including speed and position); while the localization module provides position data for the ego vehicle (the vehicle controlled by the ADS) [147].

[1]This model was used before Apollo version 7.0

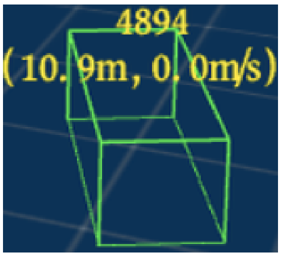


Figure 3.1: Dreamview data: Texts include the obstacle tracking ID; distance from ego vehicle; and obstacle's speed

The experiments were conducted on the Dreamview Platform of the Apollo ADS [228]. The Dreamview Platform, an integral part of Baidu's Apollo system, offers real-time visualization, simulation, and data management for autonomous vehicles [228]. It includes an intuitive interface for monitoring vehicle processes, robust simulation capabilities for safe testing, and seamless integration with the Apollo ecosystem. This makes Dreamview essential for the development and deployment of the Apollo system.

3.2.1.1 Scenario Description

In the initialization phase, the NPC vehicle was created in the rightmost lane. It proceeded at a constant speed of 10 km/h in the source test case but increased to 15 km/h in the follow-up test case. The ego vehicle (the one controlled by the ADS) was created stationary behind it in the same lane. The initial distance between the ego and NPC vehicles ensured that the ego vehicle could accelerate to a velocity greater than that of the NPC vehicle. To avoid a collision, therefore, the ego vehicle had to reduce its speed to maintain its distance from the NPC vehicle. The changing distance between the two vehicles during this entire process was the focus of the experiments.

3.2.1.2 The HMR in the Experiments

In the experiments, an HMR, $\text{HMR}_{\text{DistanceConsistency}}$, was constructed around the distance between the ego and NPC vehicles during the simulations. It was related to the distance values obtained from manually calculating data recorded during the Apollo simulations, which could also be directly obtained from Apollo Dreamview (shown in Figure 3.1) calculated in real-time during the simulation. The HMR stated that the distance values should be the same since both ways of calculation focused on the same property of the system.

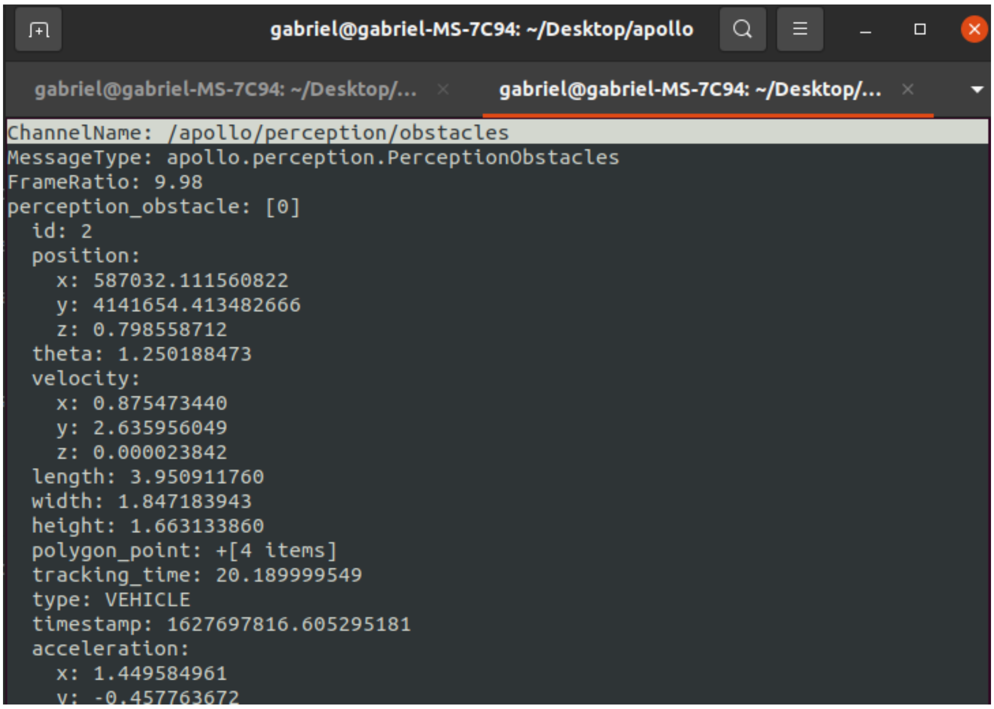
The HMR could be formally presented as:

> **$\mathbf{HMR}_{\mathbf{DistanceConsistency}}$:** In the initial stage, the ego vehicle approaches the NPC vehicle, which is moving at a slower but constant speed. Let d_D denote the distance, according to Dreamview, between the ego and NPC vehicles when both are travelling at the same speed. Similarly, let d_s be the distance calculated from simulation recordings. Since d_D and d_s refer to the same concept, the values should be equal.
>
> **Rationale:** d_D and d_s should be equal because they share the same data source (through the *perception-obstacle* and *localization* modules), and they both use Euclidean Distance [64] in the calculation.

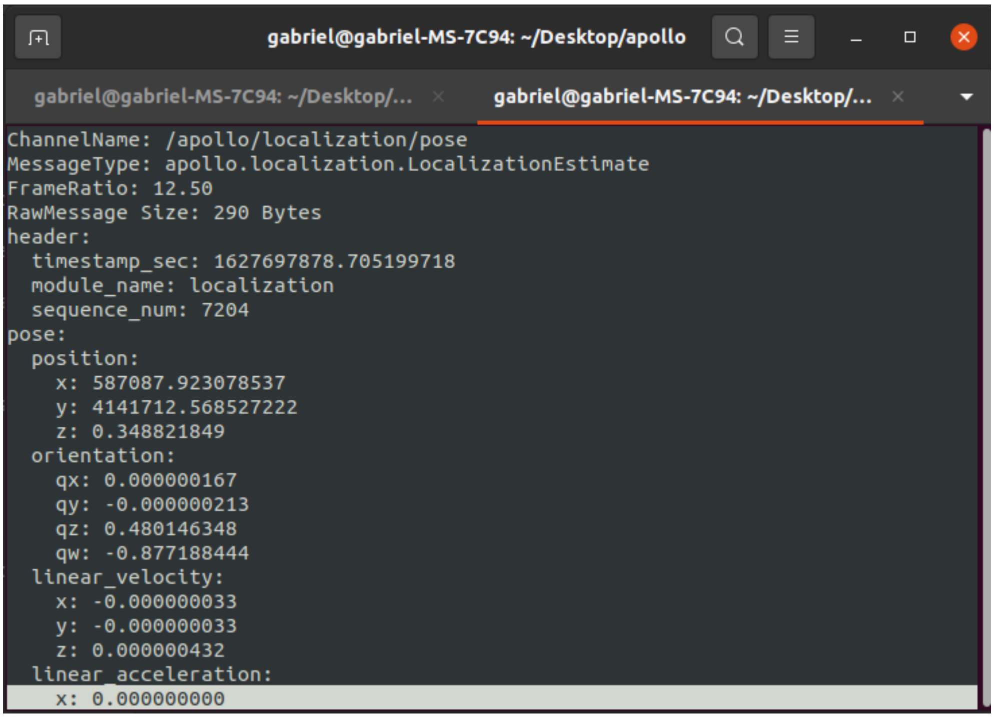
3.2.1.3 Calculating the Distance between the Vehicles

The perception-obstacle and localization modules record information about the NPC and ego vehicles, which is crucial for calculating the distance value between two vehicles, as shown in Figure 3.2[2]. The Euclidean Distance [64] between the two vehicles were then calculated using their position data from the outputs of these modules.

[2]The values displayed were monitored using the Cyber_monitor, a tool provided by Apollo to track inter-module data [9]

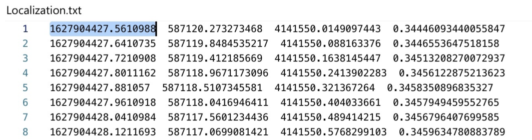


(a) NPC vehicle information (perception-obstacle module)



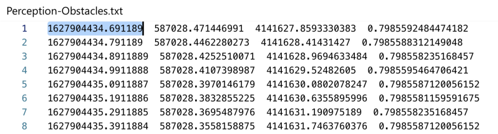
(b) Ego vehicle information (localization module)

Figure 3.2: The NPC and ego vehicle information from two modules



```
Localization.txt
1  1627904427.5610988  587120.273273468  4141550.0149097443  0.34446093440055847
2  1627904427.6410735  587119.8484535217  4141550.088163376  0.3446553647518158
3  1627904427.7210908  587119.412185669  4141550.1638145447  0.34513208270072937
4  1627904427.8011162  587118.9671173096  4141550.2413902283  0.3456122875213623
5  1627904427.881057  587118.5107345581  4141550.321367264  0.345835089683532 7
6  1627904427.9610918  587118.0416946411  4141550.404033661  0.3457949459552765
7  1627904428.0410984  587117.5601234436  4141550.489414215  0.3456796407699585
8  1627904428.1211693  587117.0699081421  4141550.576829910 3  0.3459634780883789
```

(a) Localization module data



```
Perception-Obstacles.txt
1  1627904434.691189  587028.471446991  4141627.8593330383  0.7985592484474182
2  1627904434.791189  587028.4462280273  4141628.41431427  0.7985588312149048
3  1627904434.8911889  587028.4252510071  4141628.9694633484  0.798558235168457
4  1627904434.9911888  587028.4107398987  4141629.52482605  0.7985595464706421
5  1627904435.0911887  587028.3970146179  4141630.0802078247  0.7985587120056152
6  1627904435.1911886  587028.3832855225  4141630.6355895996  0.7985581159591675
7  1627904435.2911885  587028.3695487976  4141631.190975189  0.798558235168457
8  1627904435.3911884  587028.3558158875  4141631.7463760376  0.7985587120056152
```

(b) Perception-obstacle module data

Figure 3.3: Apollo ADS start timestamps in (a) localization and (b) perception-obstacle modules

During this process, a challenge arose when computing values from both sensors due to the misalignment of their recorded timestamps. The localization module began recording at the start of the simulation, while the perception-obstacle module only captured obstacle data when the NPC vehicle was within the ego vehicle's Region of Interest (ROI). As a result, there could be significant discrepancies in the timestamps from these sensors, complicating the alignment needed to calculate the distance between the NPC and ego vehicles.

The ME approach involved extracting sensor values from the data bag[3] after the simulation had ended. Since the starting timestamps of the position data for both the NPC and ego vehicles were usually different (Figure 3.3), it was necessary to find the timestamp in the localization data that was closest to the start time of the perception-obstacle module.

In the original algorithm (presented below), after extracting the position information from the data bag, the alignment of the timestamps was performed. Once the timestamps were aligned, the distance could be calculated using position information from both the perception-obstacle and localization modules.

Algorithm Original algorithm for calculating the distance value

Require: The data bag generated after the simulation has ended.
1: Extract the position data of both the NPC and ego vehicles into two arrays.
2: Align the timestamps between the two arrays.
3: Calculate the Euclidean Distance between the corresponding positions in the two arrays.
4: **return** An array containing the distance information.

[3]A data file containing all the messages sent by sensors that can be replayed or extracted later for analysis.

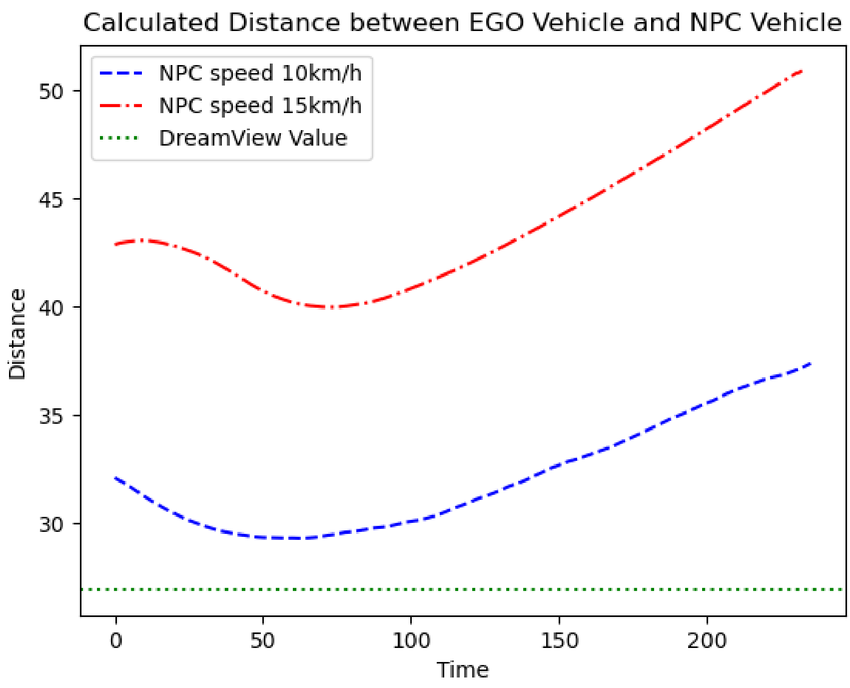


Figure 3.4: Distance values when the ego vehicle reaches the same speed as the NPC vehicle

3.2.2 Insights Gained from ME

3.2.2.1 ME-Identified Anomaly

Figure 3.4 shows the calculated distance values after implementing the original algorithm.

By comparing the blue (dash) and red (dash-dot) values in Figure 3.4, it can be seen that the calculated distance values showed a different trend from those calculated by the Dreamview, indicating a violation of $\text{HMR}_{\text{DistanceConsistency}}$: After the ego vehicle slowed down to the same speed as the NPC vehicle, the calculated distance first decreased, then continued to increase (Figure 3.4), which was different to the Dreamview data: Regardless of the NPC vehicle speeds in different test cases, the Dreamview distances for when the two vehicles were travelling at the same speed were almost the same.

To better understand the apparent anomaly, and explore whether the increasing

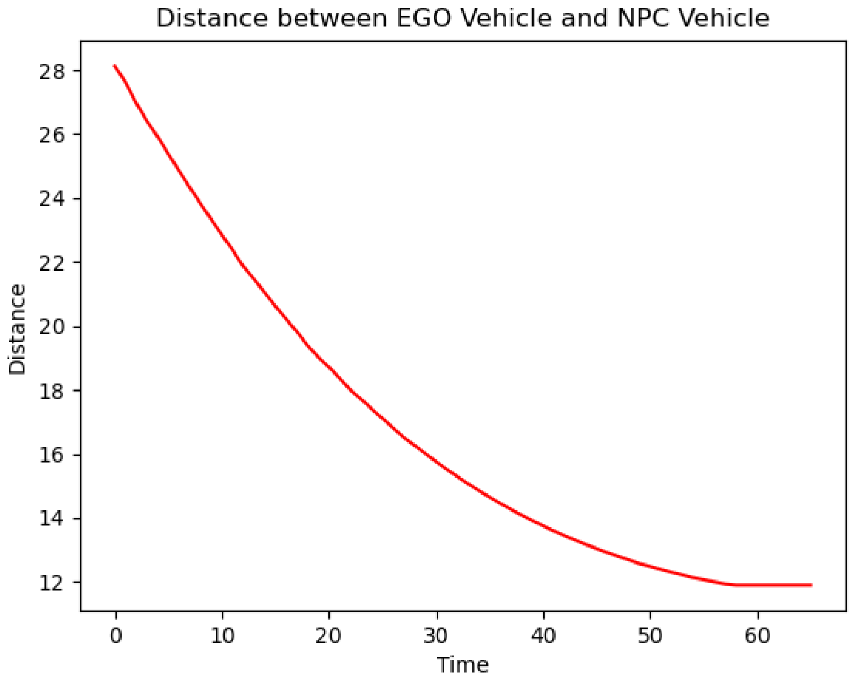


Figure 3.5: Calculated distances when the NPC vehicle is stationary

distance was caused by the movements of the NPC vehicle, a similar scenario was constructed, while this time the NPC vehicle remained stationary. The calculated distance values are shown in Figure 3.5. As can be seen in the figure, when the NPC vehicle did not move, the calculated distance showed a clear decreasing trend until the ego vehicle stopped behind the NPC vehicle. This was also aligned with the distance values obtained from the DreamView. This situation suggested that the anomaly described before would only occur when the NPC vehicle was moving.

3.2.2.2 Analysis of the Original Algorithm and Improvements

As described before, one challenge of the distance calculations was the alignment of timestamps for the position data of the NPC and the ego vehicles. This step could be erroneous, as the sampling frequency of the two modules was different [147]: Even when the recording start times were the same, the amount of data in the two modules was different. Since calculating the Euclidean Distance required data from the two modules aligned at one time, as more data was processed, differences between the timestamps of the position information would increase as well. A detailed example of this potential erroneous step is shown below:

Assuming that the data from the two modules has been stored in two arrays, with the sampling frequency for the localization module being f_L, and for the perception-obstacle module being f_{po}: If the counter of the data compared has been set to t, and the initial timestamps have been aligned, then the corresponding indexes of the module position data for two modules at the same time are $I_L = f_L * t$ and $I_{po} = f_{po} * t$.

In order to get accurate results, the distance should be calculated under the same timestamp, which means that the difference between the indexes should be zero. However, since the lengths of the two arrays are not the same (because of the sampling frequency), the difference between the indexes is:

$$I_{\text{Diff}} = |f_L - f_{po}| * t \tag{3.1}$$

As t increases (when more comparisons are performed), the difference I_{Diff} also increases, resulting in the increasing trend of the distance values (the abnormal data) in Figure 3.4. For example, at the start of the comparison, $t = 0$, so I_{Diff} is also 0, and the position information of the NPC and ego vehicles is recorded at the same time (because the start timestamps of two arrays have been aligned). However, when calculating the distance after 10 units of time, I_{Diff} becomes $|f_L - f_{po}| * 10$, and the corresponding position information of the two vehicles may no longer be recorded at the same time. Figure 3.6 illustrates this kind of index differences situation.

After discussing the anomaly with experienced Apollo developers, an updated algorithm was developed as follows, where the sensor data was no longer processed after the simulation ended: Instead, two listener programs [70] that can capture

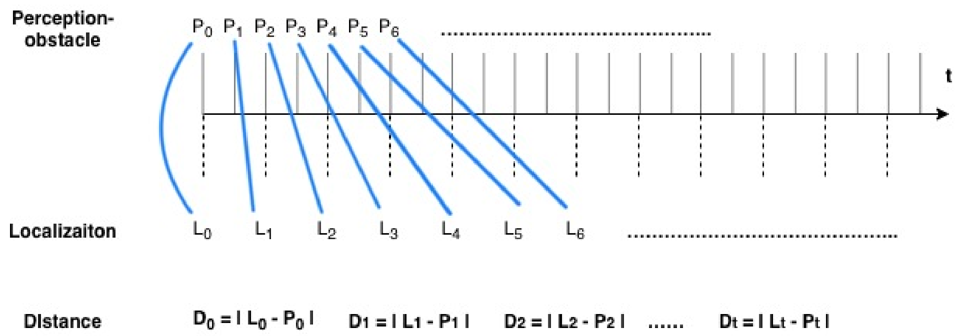


Figure 3.6: Index differences

the sensor data while the simulation was running were created—one for each module. The algorithm called on the two listeners simultaneously, thus gathering the data at the same time (e.g., every 0.002s). This revised approach enabled the distance to be calculated directly and removed the influence of the potential misalignment of the timestamps.

Algorithm Updated algorithm for dynamic data processing

Require: Real-time sensor data from sensors.

1: Create listeners for the perception-obstacle module and the localization module.
2: **while** Simulation is running **do**
3: Get the position data of the NPC vehicle from the listener node of the perception-obstacle module.
4: Simultaneously, get the position data of the ego vehicle from the listener node of the localization module.
5: Calculate the Euclidean Distance between the NPC and ego vehicles.
6: Store the distance values in an array.
7: **end while**
8: **return** An array containing the distance information.

Figure 3.7 shows a comparison of the distance values calculated by the original (dashed line) and the updated algorithm (unbroken line). It can be found from the figure that after the ego vehicle approached the NPC vehicle, it maintained the distance when travelling at the same speed as the NPC vehicle (the updated algorithm), which was also almost the same as the distance values shown on Dreamview (where the distance values were obtained from DreamView).

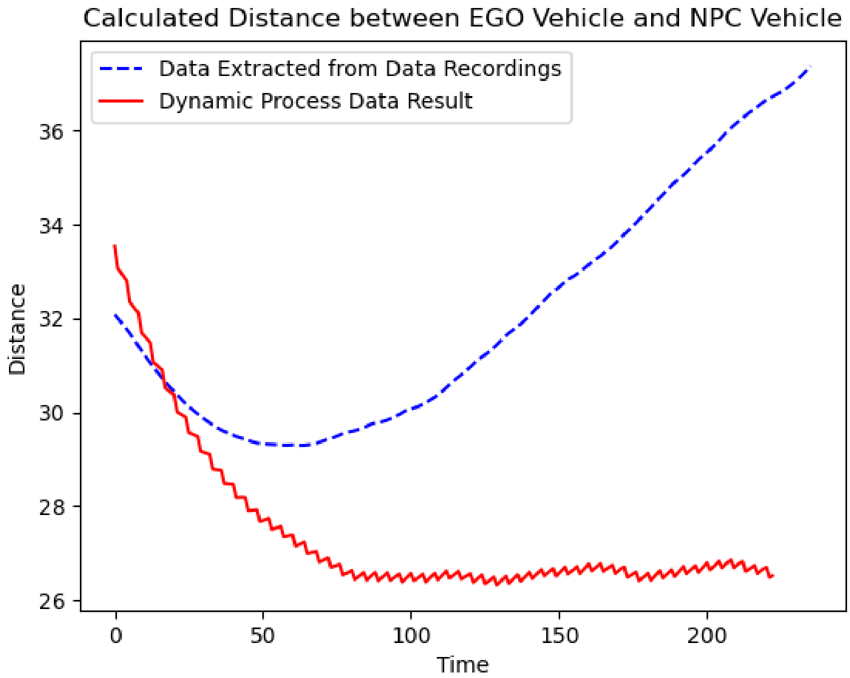


Figure 3.7: Comparison of distance values calculated by the original (dashed line) and the updated algorithm (unbroken line)

3.2.3 Discussion

This experience highlights the potential for ME to scaffold the testing process of such complex systems as Baidu Apollo ADS. Although the identified anomaly was not an Apollo ADS flaw, its discovery led to a deeper exploration and understanding of the SUT. This in itself can be a valuable learning experience and pathway for testers, especially those embarking on their earliest testing experiences, such as students or trainees [62]. For instance, this experience provided the author with a solid foundation in generating MRs and performing MT on ADSs. The author subsequently conducted MT experiments on the perception-camera module of the Apollo ADS, successfully detecting real defects.

In keeping with the SE tradition of retrospectives and reflective practice [27], this experience has prompted the following insights for ADS designers, testers, and educators:

1. **Potential solution for similar problems:** Other ADS developers and testers might also encounter similar problems described in this experience since analyzing the information in the data bag is a common approach in testing. They might find the solution presented here applicable to their own situation.

2. **ME supporting system understanding:** The identified problem in this experience was mainly due to the lack of sufficient documentation detailing the ADS system, including the explanations of different sampling rates for the different modules. In the absence of such documentation, this ME experience not only effectively led to the discovery of this information, but also resulted in a better awareness of the possibility of such system discrepancies.

3. **Potential use of ME to teach/train SQA:** The potential use of MT to teach and train SQA workers is promising. This experience enhanced the tester's understanding of ADSs and highlighted potential misunderstandings that can arise without comprehensive documentation. Integrating MT into a broader SQA training program has proven effective and engaging [198, 199]. In practice, learners generate assumptions based on their current understanding of ADSs while producing HMRs in ME. If HMRs are satisfied, the initial assumptions are validated; If they are violated, as shown with this practice, learners would gain valuable insights into ADS mechanisms and behaviours. This approach, although based on the Apollo ADS, has practical applicability to other complex or traditionally "untestable" systems [217].

The ME practice in this section established a strong base for creating MRs and conducting MT on the ADSs. This led to the MT experiments on the perception-camera module of the Apollo ADS in the following section, which revealed a critical defect in the obstacle identification model through adjusting the brightness of the input images.

3.3 Implementing MT for Baidu Apollo ADS

3.3.1 Defining MRs for the Study

As verifying the outputs of the perception module can be very difficult—under specific scenarios, it is unrealistic for testers to continue tracking whether the perception model correctly identifies all the obstacles in the scene—it is an instance of the oracle problem [21]. In this section, a series of MT experiments were conducted to test the obstacles identified by Apollo's perception module against predefined MRs [250]. The generation of MRs was based on the mechanisms of the YOLO model [164] (the detailed SUT in this study), and previously-generated driving scenarios. The YOLO model works by dividing an input image into a grid and simultaneously predicting bounding boxes and probabilities for each grid cell to locate different objects [164]. Similar to a study that separated the images into the background region and target-object region [196], this study separated the input images[4] into two regions, a *Modification Zone* and an *Identification Zone*. However, the MRs in this study modified the entire region, while the other study [196] targeted certain objects from the region. The *Identification Zone* contained the obstacles the YOLO model had identified. It was generated based on the assumption that changing pixel values outside the bounding boxes of obstacles on the image should not affect the identification result. Due to the nature of the YOLO model [164], the probability results of whether the obstacles within the grids of the image should not change if the pixel values outside the grids were changed (the altered pixels were not close to the obstacles).

Figure 3.8 validates the assumption by altering the horizontal and vertical layouts of the *Modification Zone*: If changing the pixel brightness around the obstacle's bounding box affected the YOLO model's ability to detect the obstacle, as seen in

[4]The images were parsed with the official APIs from the data-recording bags provided by Apollo [40]

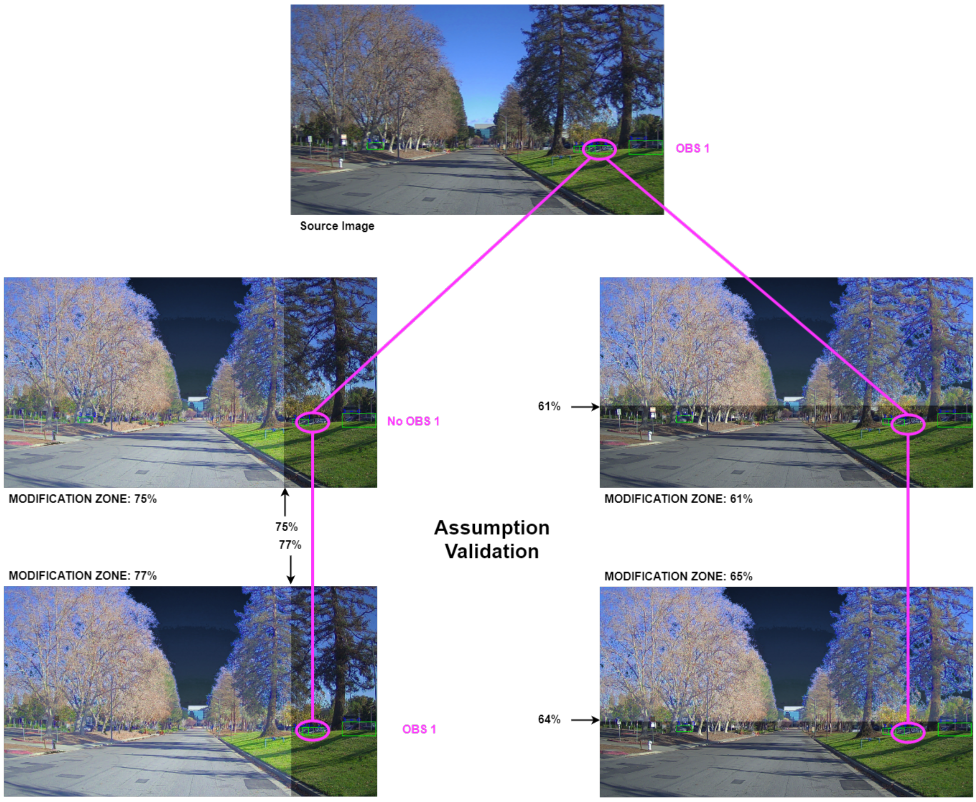


Figure 3.8: Validation of the assumption motivating the MRs. If changing the pixel brightness around the obstacle's bounding box would affect the possibility of the YOLO model detecting the obstacle (e.g., decreases it), then increasing the boundary of the modification zone to make it closer to the original bounding box should result in the obstacle remaining unidentified.

the middle images where the obstacle was missed, then expanding the boundary of the *Modification Zone* to make it closer to the original bounding box should still leave the obstacle unidentified. However, the bottom images show that the obstacle was identified again with the same bounding box as the original image. The two examples of different layouts between the two regions prove the correctness of the assumption.

Based on this assumption, two MRs were generated:

> $\mathbf{MR_{ChangeBrightness}}$: The source test case is the raw input image with the identified obstacles (in the *Identification Zone*). Then, in the follow-up cases, changing the brightness of the pixels in the *Modification Zone*, the obstacles identified should not change (Figure 3.9

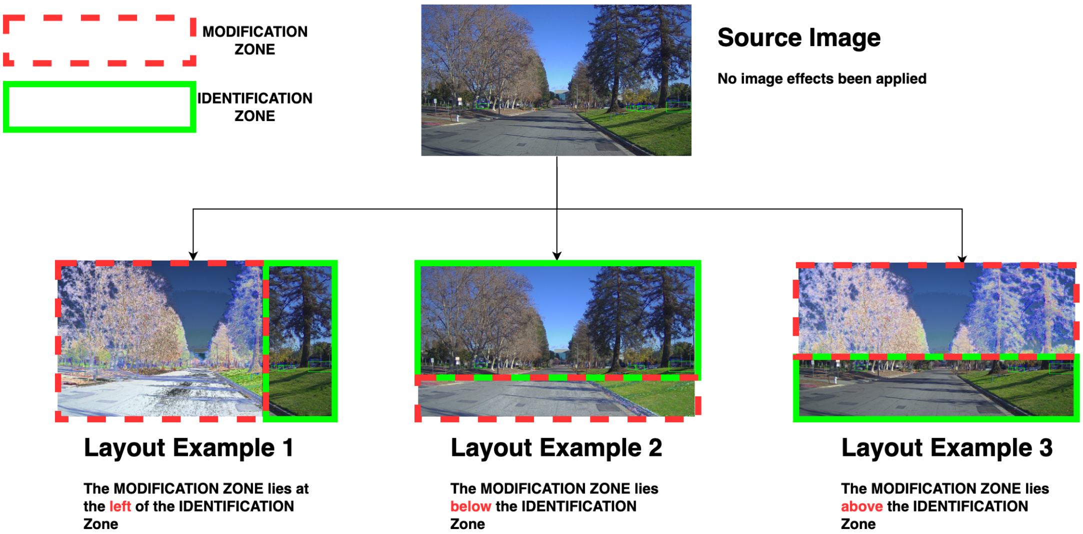


Figure 3.9: Example layouts of two zones

shows an example layouts of two zones, which are stitched in both horizontal and vertical approaches, with area ratios ranging from 10% to 90%).

MR$_{\mathbf{ChangeArea}}$: Changing the area of the *Modification Zone* (the boundary does not overlap with the bounding box of obstacles identified in the source test case in the *Identification Zone*) while maintaining the same pixel brightness value, the obstacles identified should not change.

3.3.2 Developing an ADS-Based Test Harness Framework

The author proposed a test harness framework that aimed to automate the process of test case generation and execution [250]. The framework is structured into several key components: the Test Case Generator, the ADS Test Harness Coordinator, and the Result Processor. The Test Harness Coordinator is the core component that manages all other scripts and ensures that they are executed in the correct order. The framework takes source test cases (i.e., source raw images) as inputs and sends them to the MT Test Case Generator, which generates follow-up test cases according to the MRs. The Coordinator shifts test cases from a cases pool produced in the preparation phase after all cases have been processed and testing has been completed. Prior to the conclusion of testing, the Result Processor is triggered to analyze the data and provide a report that includes information on test time, test cases, and MR-violation details.

Figure 3.10 illustrates the key phases of the proposed framework. The operation starts with the intake of source test cases—represented as images since the framework was designed for testing the perception module that uses images as input—which are then sent to the MT Test Case Generator. This generator then produces follow-up test cases based on MRs. After all test cases are processed and testing is complete, the Test Harness Coordinator passes the results to the Results Processor, which is responsible for analyzing the data and generating a report that includes information on test duration, case contents, and instances of MR violations.

The implementation methodologies, including inter-docker communication [23], are adaptable across various ADS configurations. This framework serves as a procedural guide for conducting MT on common ADS modules to organize the testing process and enhance efficiency. The detailed implementation of the framework is described as follows:

1. **The Test Harness Coordinator:** The Test Harness Coordinator is the framework's central component. It first calls the Test Case Generator (① in Figure 3.10), which produces multiple input images during each run. After generating the images, the coordinator starts the Apollo perception-camera module (②) to take raw images as inputs. It then outputs images containing the identified obstacles with bounding boxes (as shown in Figure 3.11). Once all the images within the same MG have been processed, the Result Processor program is started (③), which judges whether or not the obstacles identified in the source and follow-up images are different. It will output a

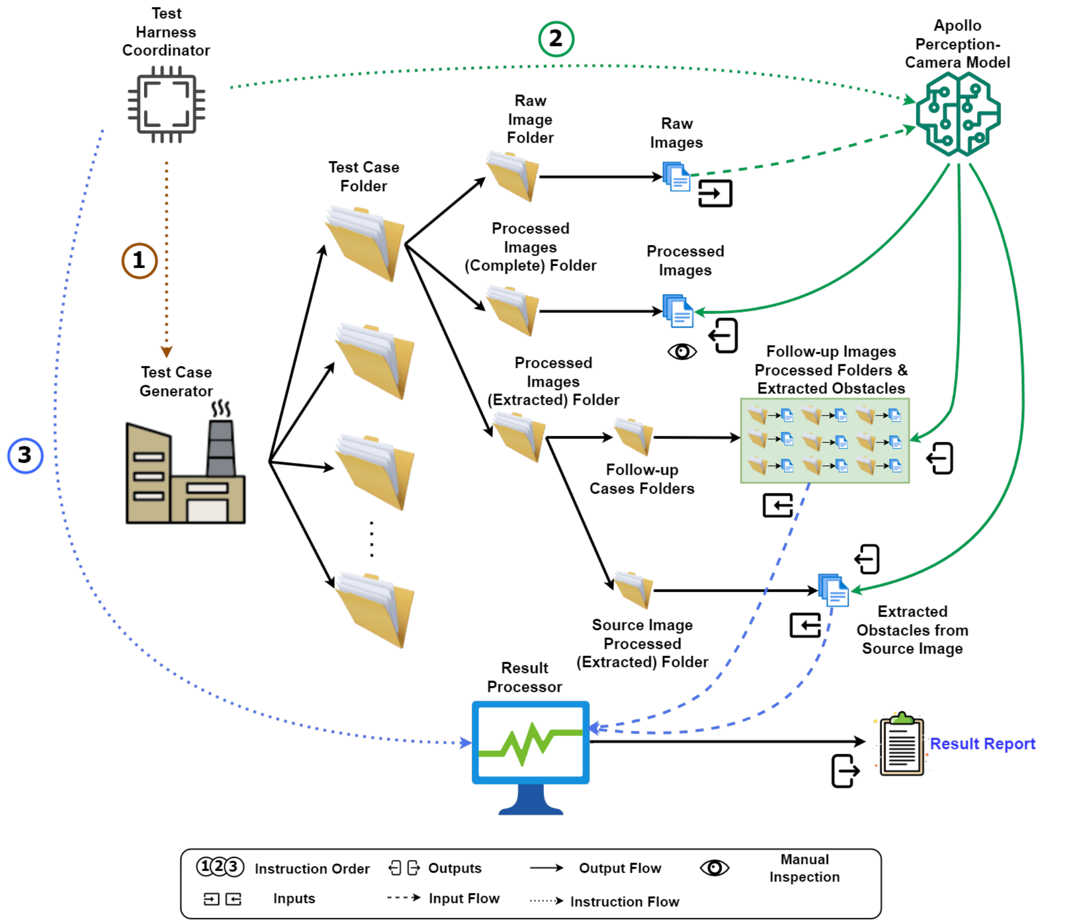


Figure 3.10: Key phases of the ADS-based test harness framework, including input and output flows (as labelled in the legend)

final report that summarizes whether or not any MRs have been violated.

2. **Test Case Generator:** The Test Case Generator is a program that takes a source image and generates follow-up images based on specified parameters such as separation direction, enumeration, quantity, and brightness levels. When changing the pixel brightness, the input images are transformed from an RGB (Red, Green, Blue) to HSV (Hue, Saturation, Value) colour space so that the pixel brightness can be increased by code. The images are then transformed back to RGB for the Apollo perception-camera module to process.

   The images are placed following the hierarchical structure of the folder layout presented in Figure 3.10. The raw input images folder stores unprocessed images within the same MG, while the processed images (complete) folder contains both source and follow-up test case images. Identified obstacles are highlighted with green bounding boxes, as illustrated in Figure 3.11. The Result Processor extracts these obstacles and saves them as separate images in the processed images (extracted) folder for later comparison. The numbers beneath each image in Figure 3.11 denote the identified object types (e.g., "3" means identified as vehicle[5]).

3. **The Result Processor:** The Result Processor uses the Structural Similarity Index Measure (SSIM) matrix [215] to measure the similarity among obstacles identified in the source and follow-up images. The python package *image_similarity_measures* [142] was used to output the SSIM value of two images: The higher the value, the more similar the two images are. If the highest SSIM value between the obstacle identified in the source image and the obstacles identified in the follow-up image is lower than a predefined threshold (e.g., 0.8 based on the observations obtained from manual inspections of experiments results), the program gives the result that the source and follow-up test cases identified different obstacles, and the MRs are violated. In order to boost the testing efficiency, the SSIM matrix is only calculated if the number of obstacles identified in both folders is the same. According to $\mathrm{MR}_{\mathrm{ChangeBrightness}}$, if the model identified a different number of obstacles in the *Identification Zone* of the same input image, the MR is violated.

   The Result Processor generates a report detailing the time, test cases, and information about obstacles. It records the follow-up image names where the number of obstacles differs from the source image and those that may have potential MR violations. These images are then compared to the source image using the SSIM matrix to assess their similarity. The report concludes with a summary of images that have MR violations. As indicated

[5] `https://github.com/ApolloAuto/apollo/blob/266afbf68d83fa6fac7a812ff8a950223f5ab2c0/modules/perception/base/object_types.h`

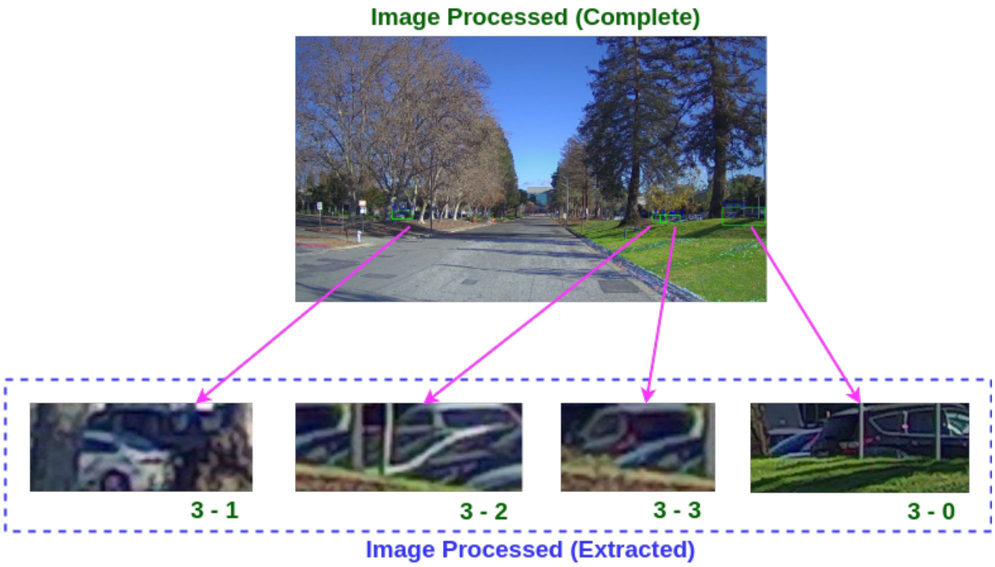


Figure 3.11: Image processed results

by the eye symbol in Figure 3.10, if no MR violations are found, a manual inspection is generally required to quickly validate the results across the images in the Processed Images (Complete) folder.

The MT experiment results can be found in Section 3.3.3.

3.3.2.1 Industrial Extension

In addition to being used in this MT study, the author has adapted the proposed framework and implemented it as an industrial extension when conducting MT on another SUT. Notably, the author observed a situation in which testers could only manually execute one scenario at a time. This was inefficient, as a significant amount of time was spent preparing to test. If the tests could be performed and the outputs could be organized automatically, testers could concentrate more on the testing itself.

In order to resolve the issue, the author customized the proposed test harness framework, and a tool tailored to the needs of testing was constructed. Similar to the Test Harness Coordinator in the proposed framework, the customized tool contained a coordinator that directs all the work. The same file management method as the proposed framework was used to organize the test cases. In addition, the customized tool could locate specific pieces of code in the input scenarios and change them to generate follow-up scenarios. The tool could execute multiple scenarios simultaneously by employing multiprocessing to coordinate the scripts within and outside of the docker container, which is a lightweight, standalone, executable package that includes everything needed to run the SUT [23]. The outputs of the system were automatically saved when a test case was finished, and organized into files and folders to enhance the testing efficiency.

By adapting the harness framework and implementing a clear architecture, this tool ensured easy maintenance and offered a complete chain of automated testing, allowing users to focus on the final results.

3.3.3 Analyzing Experimental Results

3.3.3.1 Layouts and Brightness Degrees that Cause MR Violations

This study tested over 2,000 images, involving both source (raw) images and follow-up images generated from MRs, across 20 layouts (both vertical and horizontal) and 10 brightness levels. The testing of $\mathrm{MR}_{\mathrm{ChangeBrightness}}$ and $\mathrm{MR}_{\mathrm{ChangeArea}}$ revealed that MR violations primarily occurred when the boundary of the *Modification Zone* was near obstacles in the *Identification Zone*. Increasing the brightness in the *Modification Zone* would cause the model to not identify objects, such as vehicles or pedestrians. For instance, the model missed detecting a pedestrian in the follow-up image after brightness increased, as illustrated in Figure 3.12. Further analysis identified ten specific layouts more prone to MR violations, validated with various source images from random video frames in the same data bag, which included a total of 1,000 follow-up images, as shown in Figure 3.13. It was discovered that when the *Modification Zone* was above the *Identification*

First Tier	{ Brightness 3, Vertical 63% }
	{ Brightness 5, Vertical 63% }
Second Tier	{ Brightness 3, Vertical 61% }
	{ Brightness 6, Vertical 63% }
	{ Brightness 7, Vertical 61% }
	{ Brightness 3, Vertical 64% }
	{ Brightness 2, Vertical 65% }
	{ Brightness 8, Vertical 65% }
	{ Brightness 9, Vertical 65% }

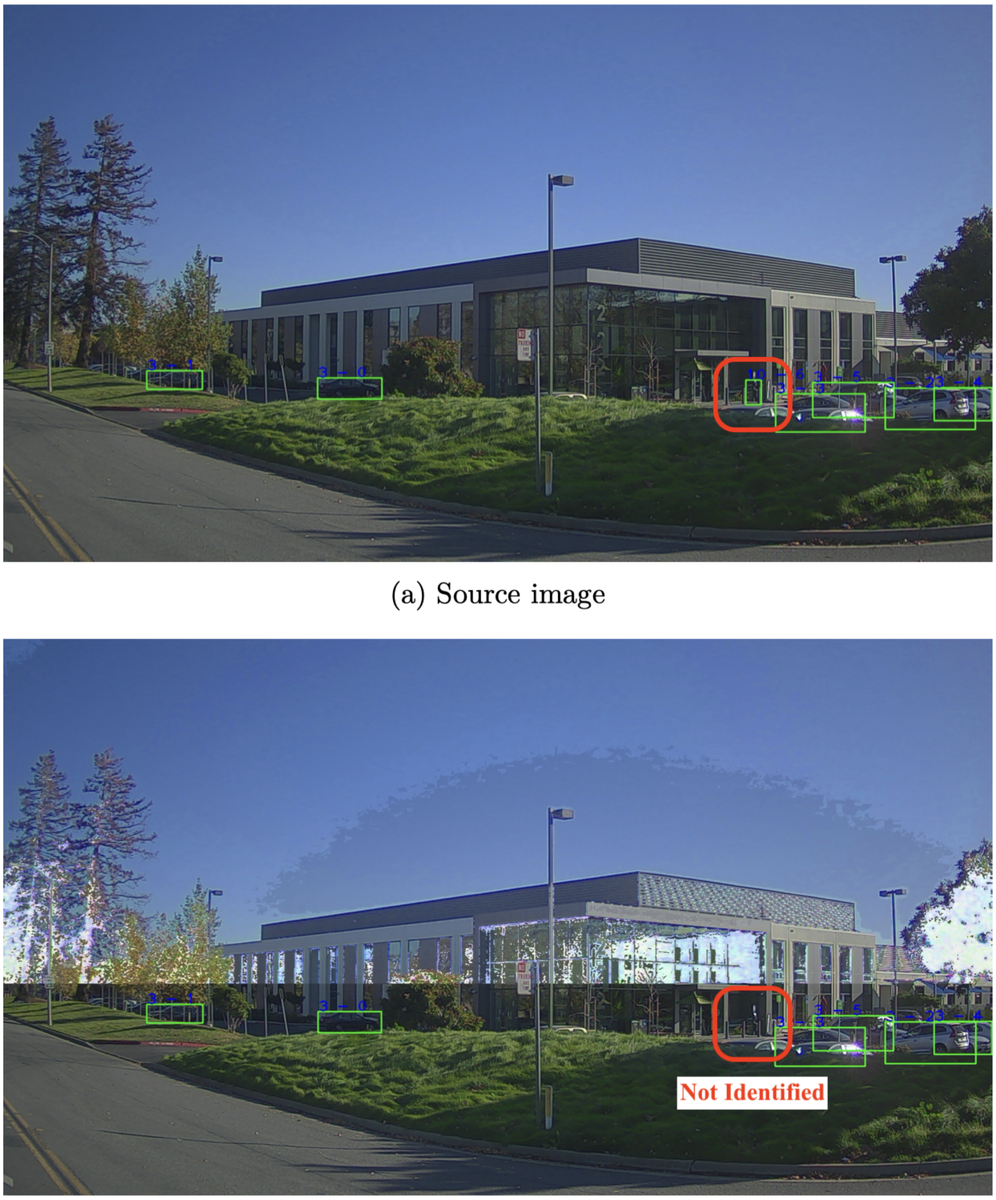
Table 3.1: Combination of brightness and layouts that cause the majority number of MR violations

Zone (as shown in Layout Example 3 in Figure 3.9), and took from 61% to 65% of the overall image, the MR violation rates were higher than other layouts, with most above 40%, indicating a higher likelihood of the model making inconsistent judgments.

The pie chart in Figure 3.14 shows the proportion of the ten brightness levels with the same *Modification Zone* among the MR-violated images when testing $\mathrm{MR}_{\mathrm{ChangeBrightness}}$. The proportions of each brightness level were similar, suggesting an equal potential for causing MR violations. Figure 3.15 shows the number of MR-violated images across various layout and brightness combinations, summarized in Table 3.1. This table highlights the layout and brightness combinations that resulted in the highest MR violations among all input images, which are also represented in Figure 3.15 by the bar heights, indicating a greater likelihood for those combinations to cause MR violations.

3.3.3.2 MR Violations on Sequences of Consecutive Frames

The MR violation situation also occurred in sequences of consecutive frames. If the object was not detected in several sequences of consecutive frames, then it would not be tracked, and would be marked as lost [72]. The Apollo perception-camera model used Kalman filtering [72] to track objects across multiple frames. Kalman filtering is a mathematical algorithm that predicts an object's future



(a) Source image

(b) Follow-up image

Figure 3.12: Example of MR violation cases

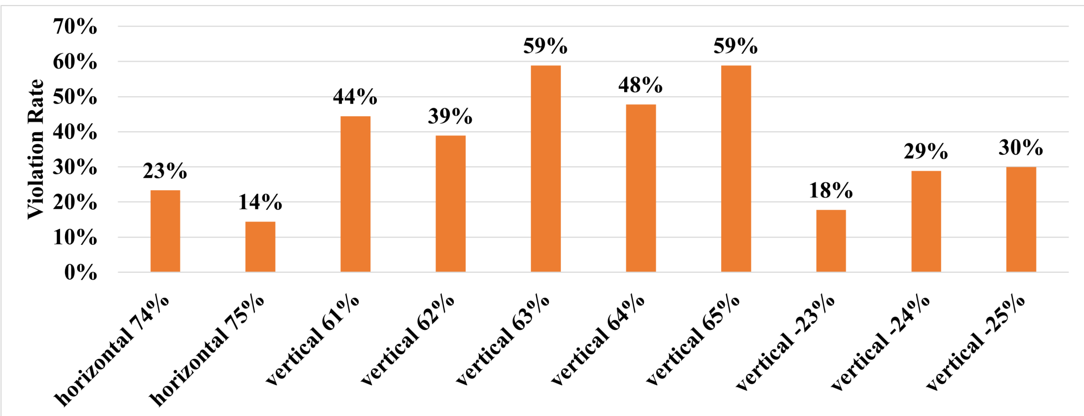


Figure 3.13: MR violation rates for the ten alignments among all the images

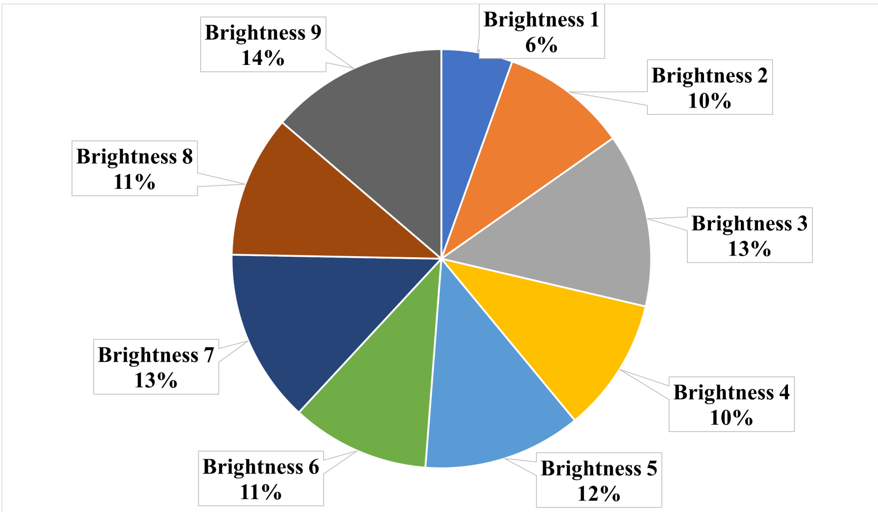


Figure 3.14: Percentage of brightness degrees among all the MR violated images

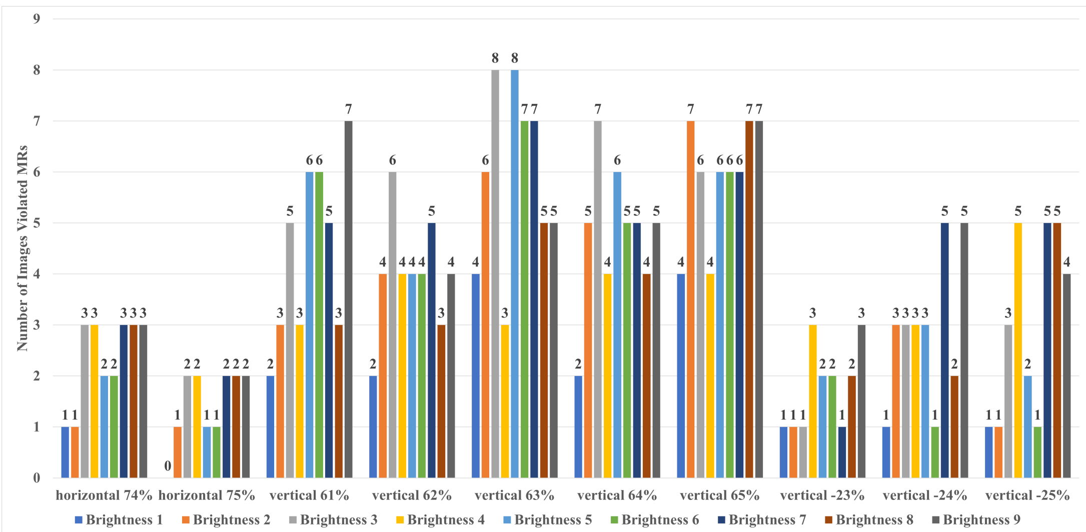


Figure 3.15: Number of MR violated images for all the combinations

position based on its previous states, helping the model maintain accurate tracking [72]. If an object has not been identified in three consecutive frames, then it is lost by the model[6].

Another study has found that the same Apollo model might lose track of obstacles in sequences of consecutive frames [257]. Figuring out whether the model had found all of the obstacles in each frame is an example of the oracle problem, especially considering that a video lasting several minutes would contain thousands of frames. The following experiments tested $\mathrm{MR}_{\mathrm{ChangeBrightness}}$ and $\mathrm{MR}_{\mathrm{ChangeArea}}$ by comparing the number of obstacles found in each frame between the source and follow-up images. The precondition was that the sequential raw images should have the same obstacles identified in consecutive frames. Then by adjusting the brightness and layouts on the frames, if a different number of obstacles was detected in the follow-up images compared to the source images, and this discrepancy was observed across consecutive frames, it indicated MR violations in those frames.

Table 3.2 shows an example of such a violation across a sequence of consecutive frames, which were extracted from a random scenario (duration one second) from the official recorded data bag (the MR violation scenarios were similar to the example shown in Figure 3.12). The MR violation occurred mainly in Frames 1 to 4, where the number of obstacles detected in the follow-up images (with increased brightness) was different from in the source driving scene. The reason Frames 5 to 7 were not regarded as an MR violation was that the number of obstacles in Frames 1 to 4 of the source test case had not changed. Since the perception module itself had a problem where it could lose track of obstacles in consecutive frames [257], the frames in which the model functions normally were chosen to highlight the MR violation.

[6] This is defined in `https://github.com/ApolloAuto/apollo/blob/93f69712269da572206e021cc7419b21c6feb595/modules/perception/fusion/lib/data_fusion/motion_fusion/kalman_motion_fusion/kalman_motion_fusion.cc`.

Table 3.2: Number of obstacles detected on a sequence of consecutive frames

	1st Frame	2nd Frame	3rd Frame	4th Frame	5th Frame	6th Frame	7th Frame
Number of Obstacles Detected (Source)	4	4	4	4	2	3	5
Number of Obstacles Detected (Follow-up)	**3**	**3**	**5**	**3**	2	2	5

3.3.4 Conclusion

The ME/MT experiments presented in this chapter demonstrate its effectiveness in addressing the oracle problem during ADS testing. The approach can serve as a reference for others testing ADSs, while the proposed ADS-based test harness framework offers a procedural guide for organizing and streamlining MT on common ADS modules. Future extensions of this research may include applying additional image-transformation effects to raw images and updated models, as well as refining the harness to accommodate a broader range of input cases, thereby improving its usability and efficiency.

While MT has proven effective in testing ADSs, it still heavily depends on human knowledge, particularly in the MR generation process. This motivates the work presented in the next chapter, which involves a new approach by introducing AI into MR generation, specifically LLMs, and assesses the quality of the MRs to demonstrate its feasibility. Meanwhile, due to the indispensable role of simulation in facilitating ADS testing, verifying the validity of AD simulators is equally important, as it can be challenging to determine whether anomalies arise from the ADS, the simulator, or the simulated data. Chapter 5 explores the applications of MT in AD simulators, highlighting methods that improve MR generation and testing efficiency.

Chapter 4

Enhancing Metamorphic Relation Generation with ChatGPT

Publications related to this chapter

1. **Y. Zhang**, D. Towey and M. Pike, "Automated metamorphic-relation generation with ChatGPT: An experience report," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, Torino, Italy, 2023, pp. 1780-1785, doi: 10.1109/COMPSAC57700.2023.00275.

2. **Y. Zhang**, D. Towey, M. Pike, Q.-H. Luu, H. Liu, and T. Y. Chen, "Integrating Artificial Intelligence with Human Expertise: An In-depth Analysis of ChatGPT's Capabilities in Generating Metamorphic Relations," *arXiv preprint arXiv:2503.22141*, 2025. [Online]. Available: `https://arxiv.org/abs/2503.22141`.

4.1 Introduction

In the previous chapter, the author discussed the applications of MT for ADSs and proposed methods to enhance MT efficiency. Although the experimental results demonstrated the effectiveness of these techniques, they still heavily depend on domain knowledge and manual inputs. In other words, users still face barriers to quickly and effectively adopting these methods in testing, especially for beginners in MT.

Integrating AI into software testing is becoming increasingly widespread [208]. Traditional manual testing methods often face challenges related to time and accuracy, while employing AI technologies can help overcome these limitations [96]. Among the AI approaches, LLMs offer significant potential in advancing MT due to their reasoning abilities and capacity to generate human-like contents [203]. Recent advancements in LLMs have largely enhanced their capabilities, gaining global popularity [253]. Compared to humans, LLMs have the advantage of vast training data, providing them with extensive knowledge and access to numerous existing test cases [253]. Given the challenges of MR generation, particularly for beginners, and the fact that many MRs can be expressed in natural language, this chapter delves into the advancement of MT through the integration of LLMs, with a specific focus on using ChatGPT. This helps address both *RQ1* and *RQ2* of the thesis in reducing the MR generation difficulties and lowering MT barriers for beginners. ChatGPT has two versions: GPT-3.5 and GPT-4 [151]. GPT-3.5 improved language understanding and generation as the free version [153], while GPT-4, a more sophisticated model, further refined these abilities, enabling more nuanced, context-aware conversations [151].

The chapter begins with a preliminary study that evaluated the MR generation capability of GPT-3.5 [247]. It assessed its performance in generating MRs tailored to ADS functions, comparing different quantities of MRs produced. Given the complexity and unpredictability of ADS functions, evaluating the effectiveness of MR generation by GPT models is critical in determining their potential to enhance MT practices. Evaluation of MRs traditionally relies on criteria developed from empirical research and expert insights [47]. However, such criteria often lack the depth and breadth needed to effectively assess MRs in the context of increasingly complex software environments [102]. This gap in comprehensive evaluation methodologies highlights the need for ongoing research and development in this area.

This chapter introduces two sets of evaluation metrics, $Criteria_{V0}$ and $Criteria_{V1}$. $Criteria_{V0}$ was used to evaluate the quality of MRs generated by GPT-3.5, as a preliminary study [247]. The MRs were evaluated by several MT researchers, who have several years of experience in MT and software testing. The results showed that ChatGPT can be a cost-effective and time-saving approach for generating diverse and relevant MRs, though the quality and relevance of the generated relations may vary, and additional customization and refinement by the testing team may be necessary [247].

However, $Criteria_{V0}$ has several limitations, including a lack of objectivity and clarity. In addition, with the launch of GPT-4, the capability of ChatGPT has a large advancement [151]. Therefore, a follow-up study to GPT-4 was conducted, with an updated set of criteria, $Criteria_{V1}$, to provide a more comprehensive, effective, and objective assessment of MRs. The study first involves a comparison of the capabilities between GPT-3.5 and GPT-4. The MRs generated by GPT-4 and the ones gathered in the previous study were compared, both targeting the parking function of an ADS [247]. To directly benchmark the results, the comparison was based on $Criteria_{V0}$. The results showed that GPT-4 has had improvements in areas like correctness, novelty, and clarity while maintaining high standards in other aspects such as applicability and computational feasibility.

To provide a more comprehensive evaluation of GPT-4's ability to generate MRs, the author extended the assessments from one SUT to a larger group of nine SUTs, encompassing both simple and complex AI/ML systems. The revised evaluation employs $Criteria_{V1}$, which includes seven criteria: completeness, correctness, generalizability, novelty, clarity, computational feasibility, and applicability. Each criterion has been redefined to reduce ambiguity and enhance objectivity, aiming for a standardized evaluation process. Additionally, to eliminate the bias from humans and provide more objective evaluations, a custom GPT-MR evaluator was created, which, in conjunction with human evaluators, offered a detailed comparison of AI and human capabilities in MR evaluation. The findings demonstrate the advanced capabilities of ChatGPT, especially GPT-4, in software testing and MR generation across a wide array of applications. Evaluations of GPTs' performance underscore the increasing proficiency of AI in generating and evaluating MRs, while still recognizing the indispensable role of human expertise in conducting critical and detail-oriented evaluation processes. This collaboration between AI and human skills is significant in simplifying the generation of MRs and advancing the methodologies of MT.

This chapter presents the formal evaluation results of the GPT-generated MRs, which serve as the basis for the subsequent applications of ChatGPT in MR generation throughout the thesis. Section 5.3.6.3 assesses the performance of a customized GPT-MR generator, while Section 6.3 compares the MRs produced by ChatGPT with those created by students who received a short training. Together, these contents highlight both the strengths and weaknesses of ChatGPT in advancing and simplifying complex tasks in MT, effectively addressing RO2 and RO3 of the thesis.

4.2 Evaluating MR Generation with ChatGPT (GPT-3.5)

4.2.1 Experimental Design

4.2.1.1 Generating MRs Using ChatGPT

In this experiment, ChatGPT was asked to generate MRs for the parking module of an ADS [135]. The author varied the number of MRs generated by ChatGPT, and asked several evaluators to evaluate the quality of the generated MRs based on their experiences using and understanding the parking module. Specifically, the usefulness and relevance of the MRs were assessed in testing the ADSs' parking module and their potential to reveal faults or weaknesses in the system [247].

4.2.1.2 Evaluating ChatGPT

To evaluate the effectiveness of the MRs produced by ChatGPT, the author designed the following marking scheme ($Criteria_{V0}$) shown in Table 4.1. This scheme includes a set of evaluation criteria, including correctness, applicability, novelty, clarity, relevance to safety, overall usefulness, and computational feasibility. Specifically, correctness refers to the degree to which an MR is free of errors and accurately captures the intended bahaviour of the system. Applicability evaluates how widely an MR can be applied to different inputs and situations. Novelty assesses the originality and creativity of the MR. Clarity measures the ease of understanding and interpretation of the MR. Relevance to safety assesses how well the MR addresses safety-critical aspects of the system. Overall usefulness evaluates the practical utility of the MR for testing and verifying the system. Computational feasibility measures the ease and efficiency of applying the MR in practice.

Table 4.1: Contents of $Criteria_{V0}$

Criteria	**Description**	**Score**
Correctness	Does the MR accurately capture the intended behaviour?	0-5
Applicability	Is the MR applicable to a wide range of inputs?	0-5
Novelty	Is the MR a new and original idea?	0-5
Clarity	Is the MR easy to understand and unambiguous?	0-5
Relevance to safety	Does the MR address a safety-critical aspect of the module?	0-5
Overall usefulness	How useful is the MR for testing the module?	0-5
Computational feasibility	Is it computationally feasible to apply the MR?	0-5

Using these criteria, testers and researchers could gain insights into the strengths and weaknesses of the MRs generated by ChatGPT, and make informed decisions about how to optimize their testing strategies.

4.2.2 Results and Key Observations

4.2.2.1 Impact of Request Size on the Quality of MRs

To assess the effectiveness of ChatGPT in generating MRs, the author conducted an experiment wherein the model was requested to produce a diverse set of MRs, ranging from a small to a large number (e.g., 5 vs 20), for the SUT. This was inspired by the difficulties MT practitioners face in creating an adequate number of MRs [47]. The aim was to compare the quality of MRs generated by the model under different request sizes.

To ensure direct comparisons and minimize the influence of other responses in the same session, multiple requests were made to ChatGPT in separate conversational sessions, since it can use the contexts from previous conversations to generate more personalized and contextually-relevant outputs [172]. The original outcomes produced by ChatGPT are presented in Tables 4.2 and 4.3.

It was observed that asking ChatGPT to generate a smaller number of MRs produced better and more accurate results than asking it to generate a larger number for the same function. For example, the MRs in Table 4.2 contain the name, scenario description, and the relations among the inputs and outputs; while the results by requesting ChatGPT to generate a large number of MRs (Table 4.3) are not always correct, and lack such relations—they are more like function specifications.

The hypothesis of the reason behind this was that ChatGPT used a probabilistic approach to generate responses [91], and when asked to generate a large number of MRs, there was a higher chance of some MRs being irrelevant or less useful for testing. However, when generating a smaller number of MRs, ChatGPT might have more resources to focus on each individual MR, leading to a higher likelihood of generating effective and relevant MRs. Therefore, when using ChatGPT to generate MRs for software testing, it is recommended to ask for a smaller number of MRs to improve the quality and accuracy of the generated MRs.

While automated techniques for generating MRs offer various benefits, validating their correctness and relevance with domain experts before using them for software testing is necessary [18]. The lack of such validation increases the likelihood of generating irrelevant or erroneous MRs, which may result in the failure to identify defects, or the production of false positives: Testers may spend significant time and effort resolving MR violations that are actually caused by incorrect MR definitions, leading to inefficiencies in the testing process [159]. Hence, it is essential to integrate expert feedback and judgment into the MT process to ensure the efficacy and dependability of the generated MRs [47].

The quality of MRs generated by ChatGPT can vary depending on various factors, such as the complexity of the SUT and the specificity of the testing scenario. Generally, the efficacy of MRs generated by ChatGPT is lower when compared to existing MR-generation tools and strategies, such as MRPs or even human-

Table 4.2: The result of using ChatGPT for generating a small number of MRs

MRs	**Description**
$\mathrm{MR}_{\mathrm{Reverse\_into\_parking\_space}}$	Given a scenario where the autonomous vehicle is in a parking lot and is required to park in a spot, the system should be able to reverse into the parking spot without hitting any other vehicle or obstacle in its path. The metamorphic relation is to test if the system is able to park in the same spot but with a different orientation, such as turning the vehicle 90 degrees or reversing into the spot from a different angle.
$\mathrm{MR}_{\mathrm{Parallel\_parking}}$	Given a scenario where the autonomous vehicle is required to parallel park between two other parked cars, the system should be able to navigate into the parking spot and position itself correctly without hitting any other vehicle. The metamorphic relation is to test if the system is able to parallel park in a different spot, with different-sized vehicles on either side.
$\mathrm{MR}_{\mathrm{Obstacle\_detection}}$	Given a scenario where the autonomous vehicle is required to park in a parking lot, the system should be able to detect obstacles such as pedestrians, other vehicles, or shopping carts, and avoid hitting them. The metamorphic relation is to test if the system is able to detect and avoid different types of obstacles, such as bicycles or large delivery trucks.
$\mathrm{MR}_{\mathrm{Parking\_spot\_occupancy}}$	Given a scenario where the autonomous vehicle is required to park in a parking lot, the system should be able to identify empty parking spots and park in them. The metamorphic relation is to test if the system is able to identify a parking spot as occupied and adjust its path accordingly to find another spot.
$\mathrm{MR}_{\mathrm{Changing\_parking\_conditions}}$	Given a scenario where the autonomous vehicle is required to park in a parking lot, the system should be able to adapt to changing parking conditions, such as an obstruction in the path, a spot becoming unavailable, or a vehicle leaving its spot. The metamorphic relation is to test if the system is able to adjust its path and behaviour in real-time based on changes in the parking environment.

Table 4.3: The result of using ChatGPT for generating a large number of MRs

MR	Description
MR1	If the initial parking spot is occupied, the system should identify an alternative parking spot.
MR2	If the car is facing in the opposite direction of the initial parking spot, the system should perform a U-turn and approach the spot from the correct direction.
MR3	If there are obstacles in the parking spot, the system should not attempt to park and display a warning message.
MR4	If the car is moving too fast, the system should not attempt to park and display a warning message.
MR5	If the car is too close to the car in front or behind, the system should not attempt to park and display a warning message.
MR6	If the car is too close to the curb or another obstacle, the system should adjust the parking position accordingly.
MR7	If the car is not centred in the parking spot, the system should adjust the position accordingly.
MR8	If the parking spot is too small for the car, the system should not attempt to park and display a warning message.
MR9	If the car is parked too far away from the curb, the system should adjust the position accordingly.
MR10	If the car is parked too close to the curb, the system should adjust the position accordingly.
MR11	If the car is parked on a hill, the system should engage the parking brake.
MR12	If the car is parked on a slope, the system should adjust the position to prevent the car from rolling.
MR13	If the car is parked on a rough surface, the system should adjust the position to prevent damage to the car.
MR14	If the car is parked in a tight space, the system should adjust the steering wheel accordingly.
MR15	If the car is parked on a narrow street, the system should adjust the position accordingly.
MR16	If the car is parked on a busy street, the system should display a warning message to the driver.
MR17	If the car is parked in a parallel parking spot, the system should adjust the position accordingly.
MR18	If the car is parked in a diagonal parking spot, the system should adjust the position accordingly.
MR19	If the car is parked in a handicapped parking spot, the system should display a message to the driver.
MR20	If the car is parked in a spot reserved for electric vehicles, the system should display a message to the driver.

based approaches, which allow for customization based on different scenarios. This may be due to a lack of diversity and creativity in the generated MRs. One illustrative instance of human-generated MRs is the process of amalgamating two distinct scenarios wherein the SUT produces identical outputs [249]. The resultant composite scenario ought to preserve the property of invariant behaviour such that the SUT yields the same output.

While the MRs generated by ChatGPT are useful for initial testing phases, due to their closeness to the functional requirements, their efficacy may be limited. Therefore, for complex systems requiring comprehensive testing and the generation of diverse and creative MRs, incorporating manual intervention can facilitate a more in-depth exploration of the input-output relationships, thereby improving the overall testing effectiveness [47].

4.2.2.2 Evaluating the Quality of MRs

The evaluation results are presented in Tables 4.4 to 4.8, based on the assessment of the MRs from Table 4.2. The analysis showed that all MRs exhibited high computational feasibility scores, indicating their potential for practical implementation. However, the novelty scores for these MRs were relatively low, suggesting that they did not significantly differ from existing MRs. Notably, $MR_{\text{Obstacle\_detection}}$ received the highest total score (25) due to its high scores in correctness, applicability, relevance to safety, and computational feasibility, but scored low in novelty. $MR_{\text{Changing\_parking\_conditions}}$ received the second-highest total score (24) with high scores across all criteria except for novelty. $MR_{\text{Reverse\_into\_parking\_space}}$ and $MR_{\text{Parallel\_parking}}$ received the lowest total scores (19 and 20, respectively) due to lower scores in several criteria. These results provided insights into the strengths and weaknesses of the evaluated MRs, highlighting potential areas for further improvement.

Table 4.4: Evaluation of $MR_{\text{Reverse\_into\_parking\_space}}$

Correctness	Applicability	Novelty	Clarity	Relevance to Safety	Overall Usefulness	Computational Feasibility
2	4	1	2	2	3	5

Table 4.5: Evaluation of $MR_{\text{Parallel\_parking}}$

Correctness	Applicability	Novelty	Clarity	Relevance to Safety	Overall Usefulness	Computational Feasibility
3	4	1	2	2	3	5

Table 4.6: Evaluation of $MR_{\text{Obstacle\_detection}}$

Correctness	Applicability	Novelty	Clarity	Relevance to Safety	Overall Usefulness	Computational Feasibility
4	4	1	4	4	3	5

Table 4.7: Evaluation of $MR_{\text{Parking\_spot\_occupancy}}$

Correctness	Applicability	Novelty	Clarity	Relevance to Safety	Overall Usefulness	Computational Feasibility
3	4	1	3	4	3	5

Table 4.8: Evaluation of $MR_{\text{Changing\_parking\_conditions}}$

Correctness	Applicability	Novelty	Clarity	Relevance to Safety	Overall Usefulness	Computational Feasibility
4	4	1	3	4	3	5

4.2.2.3 Enhancing Result Accuracy Through Prompt-Engineering

Prompt engineering in LLMs is the process of designing effective inputs or instructions to guide the model's output and optimize its performance for specific tasks or applications [77]. The experiments in this section have found that repeated conversations with ChatGPT have the potential to improve the accuracy of MRs generated. These interactions allow ChatGPT to learn from the past, and to refine its ability to generate effective and correct MRs [91]. As ChatGPT receives feedback about its answers, it can recognize patterns and relationships between inputs and outputs more accurately [153], leading to the generation of more comprehensive and diverse MRs. The quality of feedback provided by the user can largely determine how many user prompts are needed, and the quality of

Table 4.9: Improved MRs from Table 4.2 after interacting with ChatGPT

MRs	Input Relation	Output Relation
$\text{MR}_{\text{Reverse\_into\_parking\_space}}$	The autonomous vehicle is in a parking lot and required to park in a spot by reversing into it.	The output should be the same as the source input, but with the vehicle parked in a different orientation.
$\text{MR}_{\text{Parallel\_parking}}$	The autonomous vehicle is in a scenario where it's required to parallel park between two other parked cars, with different-sized vehicles on either side of the parking space.	The output should be the same as the source input, but with the vehicle parked between different-sized vehicles.
$\text{MR}_{\text{Obstacle\_detection}}$	The autonomous vehicle is in a parking lot and required to park in a spot, with obstacles such as pedestrians, other vehicles, or shopping carts, and different types of obstacles, such as bicycles or large delivery trucks.	The output should be the same as the source input, but with the vehicle avoiding different types of obstacles.
$\text{MR}_{\text{Parking\_spot\_occupancy}}$	The autonomous vehicle is in a parking lot and required to park in an empty spot, but with the parking spot occupied by another vehicle.	The output should be the same as the source input, but with the vehicle finding another empty spot to park in.
$\text{MR}_{\text{Changing\_parking\_conditions}}$	The autonomous vehicle is in a parking lot and required to park in a spot, with changing parking conditions such as an obstruction in the path, a spot becoming unavailable, or a vehicle leaving its spot, and different changing parking conditions, such as a different obstruction in the path, a different spot becoming unavailable, or a different vehicle leaving its spot.	The output should be the same as the source input, but with the vehicle adapting to different changing parking conditions.

the generated MRs. According to the empirical findings with ChatGPT, typical MR-enhancing prompts include: Requesting more detailed explanations of relationships; necessitating the inclusion of input-output associations; and demanding the creation of precise and properly structured MR formats.

A detailed example of the interactions with ChatGPT regarding the evaluation of responses can be found in Appendix A, while Table 4.9 shows the refined MRs, evolved from the originals in Table 4.2. It can be found that the updated MRs demonstrate enhanced clarity and accuracy regarding the relationships between inputs and outputs. For instance, for $\mathrm{MR}_{\mathrm{Parallel\_parking}}$, the original version merely described the source scenario, while the follow-up scenarios and the output relation were vague ("The metamorphic relation is to test if the system is able to parallel park in a different spot, with different-sized vehicles on either side."). This description was more like a testing objective rather than a relation. In contrast, the revised version of the MR clearly specified the follow-up scenarios and output relations as: "The output should be the same as the source input, but with the vehicle parked between different-sized vehicles."

In order to provide effective feedback, users should be familiar with the concepts of MT and MRs, although they do not need to be experts in the SUT domain. High-quality feedback enables ChatGPT to quickly learn and produce quality MRs with fewer prompts [31]. Conversely, if the feedback is of low quality or lacks specificity, ChatGPT may require more prompts to generate effective MRs.

Engaging ChatGPT in diverse conversational contexts can expose it to a broader range of scenarios and inputs, thereby enhancing the generation of more effective MRs [153]. Through repeated interactions, ChatGPT can refine its understanding of the domain-specific concepts of the SUTs, which can lead to the generation of more accurate MRs. This is particularly beneficial when testing complex software systems.

It is noteworthy, however, that the number of feedback inputs required by the model is contingent on the preceding responses, and may vary depending on the SUT and user requirements [153]. Additionally, it is important to consider the impact of incorporating user feedback into the MR-generation process, particularly in situations where the user's level of expertise or the complexity of the SUT may affect the quality of the MRs generated.

Overall, the ability of ChatGPT to learn from past interactions and refine its answers to generate effective and correct MRs makes it a valuable tool in the field of software testing [153]. By providing clear communication and guidance to users on providing effective feedback, the quality and accuracy of MRs generated through repeated interactions with ChatGPT can be further improved, leading to more effective and thorough software testing.

4.2.3 Threats to Validity

The use of ChatGPT for generating MRs is a promising approach, but its limitations and potential threats to validity must be considered. Customization and refinement may be necessary to ensure that the generated MRs are relevant and appropriate for the specific SUT under consideration. In addition, the diversity of the MRs generated may be limited, which could reduce the effectiveness of the testing approach, particularly when generating a large number of MRs.

User feedback used to refine MRs may be subjective, and individual biases or errors may negatively impact the quality of the generated MRs. The effectiveness of the testing approach may also vary depending on the complexity and variability of the SUTs, which reduces the generalizability of findings to other domains and SUTs. Moreover, the reproducibility of the approach is a potential limitation as it assumes a high level of expertise on the part of the person interacting with ChatGPT, which may reduce the replicability of the results. Additionally, the refinement process through ongoing interaction may limit the creativity that ChatGPT could offer as it adjusts its responses to the inquirer. Furthermore, the reliability of the feedback mechanism may be affected by cognitive biases, as different people may focus on different aspects of the phenomenon under study.

Therefore, there is a need to provide additional training of the AI models to enhance the quality of the generated MRs. Additionally, providing training to users on how to effectively interact with and leverage these LLMs is also necessary, as discussed in Section 4.2.2.3, where prompt engineering was shown to enhance MR accuracy.

4.3 Enhancing MR Generation with ChatGPT (GPT-4)

4.3.1 Comparative Analysis: GPT-3.5 vs GPT-4

This section compares the MRs generated by GPT-3.5 and GPT-4. Previous analyses have highlighted the quality of MRs from GPT-3.5, particularly for AD functions (Section 4.2). The results demonstrated that ChatGPT can be a cost-effective and time-saving approach for generating diverse and relevant MRs. Furthermore, asking ChatGPT to generate a small number of MRs could give better results than generating a large number of MRs [247]. To compare the quality of MRs generated by the GPT-3.5 and GPT-4 models, the same prompts were given to GPT-4, with the results evaluated using the same evaluation criteria ($Criteria_{V0}$) proposed in the previous section (Table 4.1).

4.3.1.1 Generation of MRs for New Target Systems

To gain fair and useful answers from ChatGPT, the author adopted zero-shot prompting, which was the same method used in the pilot study. Zero-shot prompting involves interacting with the model without providing explicit examples or prior context [109]. The operation was performed in two steps: first clarifying with ChatGPT about the target system, and then asking it to generate a set of unique MR candidates for each target system in each independent session. The previous experiment involved using GPT-3.5 to generate five MRs [247]. In interaction with GPT-4, by limiting the number of MRs to five for the same function, a controlled environment was established, allowing for a direct and fair comparison. It ensured that the changes observed can be attributed more confidently to the GPT versions rather than external variables. The results were evaluated by experienced experts in both MT and the relevant domain, the same individuals who previously assessed the MRs generated by GPT-3.5 [247].

4.3.1.2 Evaluation Results

Table 4.10 presents the MRs formulated by GPT-4 for the ADS Parking function. These MRs explore different aspects such as changes in parking spot order, weather conditions, obstacle proximity, vehicle dimensions, and time of day.

The evaluation results are presented in Table 4.11. In terms of correctness, the MRs generally displayed a high level of accuracy and alignment with expected results. “Applicability” was a strong area for all MRs, indicating their versatile use in different testing scenarios. However, in terms of “novelty”, all the MRs generated showed markedly low scores. This suggests that the MRs show many similarities with the ones already established in the literature, indicating a lack of innovative content or original approaches. “Clarity” was consistently high across all MRs, meaning that their applications and processes were easily

Table 4.10: MRs for ADS's parking function generated by GPT-4

MRs	Description
MR1: Permutation of Parking Spot Order	In the original input, the vehicle is instructed to park in a designated spot among a sequence of available spots. The modified input involves permuting the sequence of available parking spots. The expected output is that the vehicle should still successfully park in a designated spot, with the choice of the spot consistent with the vehicle's parking algorithm, irrespective of the permutation of spots.
MR2: Varying Weather Conditions	The original input involves parking under clear weather conditions. The modified input replicates the parking maneuver under different weather conditions (e.g., rain, snow). The expected output is that the vehicle should successfully complete the parking maneuver, adjusting its approach for safety, and ultimately achieving the same parked position.
MR3: Change in Obstacle Proximity	Originally, parking occurs at a certain distance from obstacles. The modified input adjusts the distance from these obstacles. The expected output is that the vehicle adjusts its parking maneuver but still parks successfully and safely in the designated spot.
MR4: Varying Vehicle Dimensions	The original input executes a parking maneuver with a vehicle of specific dimensions. The modified input uses a vehicle of different dimensions. The expected output is that the autonomous system adapts its parking strategy to accommodate the different vehicle size, resulting in a successful and safe park.
MR5: Time of Day Variation	The original input involves performing a parking maneuver during daytime. The modified input performs the same maneuver at night or during twilight. The expected output is that the vehicle uses its sensors and algorithms to successfully and safely complete the parking maneuver, as it did during the daytime.

understandable. The MRs' "relevance to safety" showed moderate scores, indicating their significance in safety considerations, yet suggesting room for more focused development in this area. Similarly, in terms of "overall usefulness", the MRs achieved moderate scores. This indicated their positive impact on testing processes, but implied potential for further enhancement. Lastly, "computational feasibility" was a standout feature for all MRs, reflecting their ease of integration and practicality in current testing environments.

Table 4.12 presents an evaluation comparison of MRs between GPT-4 and GPT-3.5 generated for the same target system using $Criteria_{V0}$. Compared to MRs

Table 4.11: Evaluation results of the MRs generated by GPT-4

Evaluation Criteria	**MR1**	**MR2**	**MR3**	**MR4**	**MR5**
Correctness	3	3	3	4	4
Applicability	4	4	4	4	4
Novelty	3	1	2	1	1
Clarity	4	4	4	4	4
Relevance to safety	2	2	4	3	3
Overall usefulness	3	3	3	3	3
Computational feasibility	5	5	5	5	5
Total score	24	22	25	24	24

Table 4.12: Evaluation comparison of MRs for the same target function between GPT-4 and GPT-3.5 using $Criteria_{V0}$

Evaluation Criteria	**GPT-3.5**	**GPT-4**
Correctness	3.2	3.4
Applicability	4.0	4.0
Novelty	1.0	1.6
Clarity	2.8	4.0
Relevance to Safety	3.2	3.0
Overall Usefulness	3.0	3.0
Computational Feasibility	5.0	5.0
Total Score	22.2	24.0

from GPT-3.5 [247], GPT-4 showed a slight improvement in “correctness”. Both versions scored equally in “applicability”, showing their similar capability to be applied across different scenarios. GPT-4 scored a higher “novelty” mark than GPT-3.5, suggesting slightly more innovative approaches compared to GPT-3.5, though the “novelty” score itself was not high.

However, “clarity” was significantly better in GPT-4, since the MRs generated by GPT-4 have a clear structure and less vague expressions that made the MRs clearer and more comprehensible. This was a significant improvement in GPT-

4, as one of the main limitations found in the previous study of GPT-3.5 was that the MRs generated were closer to the system specifications instead of the relations among multiple test cases [247]. GPT-4 clearly specified the source test case, with both input and output relations, which eased the process of generating test cases and analyzing results against MRs.

In “relation to safety”, GPT-3.5 scored slightly higher, since its MRs were more aligned with safety considerations. Both versions were equal in terms of “overall usefulness”, suggesting a balanced contribution to the testing process. In “computational feasibility”, both GPT-4 and GPT-3.5 achieved the full score, reflecting ease of implementation and practicality in testing environments.

The total score revealed that GPT-4, with a score of 24.0, outperforms GPT-3.5, which had a score of 22.2. This comparison indicates that the advancements in GPT-4 have led to improvements in areas like correctness, novelty, and clarity while maintaining high standards in other aspects such as applicability and computational feasibility.

4.3.2 Establishing New Evaluation Criteria

The pilot study (Section 4.2.3) identified the refinement of evaluation criteria as a potential area for future research. As indicated in Section 4.2.3, the original descriptions for each criterion exhibit a degree of subjectivity and ambiguity. This potentially leads to varying interpretations and assessments by evaluators, each bringing their own unique experiences and knowledge to the evaluation process. Moreover, the criterion “relevance to safety” was found to be somewhat restrictive, primarily applicable to SUTs that are related to safety areas, thereby limiting its broader applicability.

In response to these limitations, the author has worked on enhancing both the objectivity and the general applicability of these evaluation criteria by proposing an updated evaluation framework, $Criteria_{V1}$. It now includes seven criteria: completeness, correctness, generalizability, novelty, clarity, computational feasibility, and applicability of the SUT. Each criterion has been carefully redefined to minimize ambiguity and enhance objectivity, aiming for a more standardized and uniform evaluation process.

In $Criteria_{V1}$, each main criterion is divided into several specific sub-criteria, with each contributing to a holistic assessment. This approach not only simplifies the evaluation process, but also allows for a more precise and targeted analysis of each component's contribution to the overall functionality and reliability of the MR. The criteria are designed with a scoring system: If an MR completely satisfies all aspects of a criterion, it is awarded full marks, otherwise, it gains marks for the matching sub-criteria. This approach ensures a clear-cut and straightforward evaluation.

Table 4.13 presents the details of $Criteria_{V1}$. The "completeness" criterion is focused on verifying the inclusion of all essential components within an MR: the source test case, input relation, and output relation. The input relation is necessary for the generation of follow-up test cases, serving as the blueprint for how these cases are derived from the original source test case. The output relation contributes to the validation phase. It provides a benchmark against which the outputs of both the source and the follow-up test cases are evaluated. The "correctness" criterion is concerned with how well the input and output relations of the MR adhere to the intended behavioural patterns and the overall correctness of the MR. The updated version enhances evaluation accuracy by breaking down the assessment into distinct evaluations of essential components, resulting in a more detailed and transparent process compared to the general "correctness" criterion in $Criteria_{V0}$ (Table 4.1).

The new "generalizability" criterion now explicitly defines the range of SUTs for which an MR is suitable (originally the "applicability" in $Criteria_{V0}$). Unlike

Table 4.13: Contents of $Criteria_{V1}$

Criterion	Description	Levels
Completeness	Evaluates if the MR includes all key components.	**1 mark**: All components are present (source test case, input relation, output relation). **0 marks**: Missing one or more components.
Correctness	Assesses if the MR reflects the SUT's expected bahaviour.	**3 marks**: Both input and output relations align with the SUT specifications. **0 marks**: Any violation of SUT specifications.
Generalizability	Evaluates the MR's adaptability to different SUTs.	**3 marks**: Applicable to all systems in the same category. **2 marks**: Applicable to a specific group of systems. **1 mark**: Only applicable to the current SUT.
Novelty	Measures the originality of the MR.	**3 marks**: Entire MR is innovative. **2 marks**: Output relation is novel. **1 mark**: Input relation introduces some novelty.
Clarity	Assesses the ease of understanding the MR.	**3 marks**: Clear to a general audience. **2 marks**: Understandable to those with basic field knowledge. **1 mark**: Requires expert-level understanding.
Computational Feasibility	Evaluates ease of automating the MR.	**3 marks**: Fully automatable, both generation and validation. **2 marks**: Test generation automatable, validation requires manual steps. **1 mark**: Test generation is simple, but automation is limited.
Applicability	Assesses relevance to the SUT's unique features.	**3 marks**: Both input and output relations directly reflect the SUT's unique features. **2 marks**: Either input or output relates to the SUT. **1 mark**: The MR is generic and unrelated to specific features of the SUT.

the earlier version, which was vague about the "wide range of inputs", the new definition categorizes generalizability into three distinct levels: only the current SUT; specific groups of SUTs; and all types of SUTs. This structured approach provides a clearer framework for assessing the MR's adaptability, enhancing the precision of the evaluations.

The updated "novelty" criterion focuses on assessing the uniqueness of MRs, particularly in their approach and formulation of input/output relations. While it targets the same fundamental aspect as the older version, the refined criterion simplifies the scoring process and enhances objectivity by distinctly evaluating the innovation in input and output relations.

The new "clarity" criterion categorizes the audience into domain experts, individuals with basic understanding, and the general public. This stratification allows evaluators to more objectively determine the MR's comprehensibility across different knowledge levels.

The "computational feasibility" criterion introduces three specific dimensions:

ease of generating source test cases; feasibility of automating test case generation; and feasibility of automating MR validation. This comprehensive approach, encompassing the entire MT process, provides a more detailed and objective framework compared to the original criterion, which lacked clear definitions for evaluating computational feasibility.

Finally, the "relevance to safety" criterion was replaced with the new "applicability" criterion, expanding its applicability beyond safety-focused SUTs. This criterion now evaluates the MR's relevance in three categories: no relevance; partial relevance; and strong relevance to the SUT's key features. Additionally, the "Overall Usefulness" criterion was removed due to its lack of specificity, which made it challenging to apply effectively in assessing the MRs. This revision ensures that each criterion directly contributes to a more targeted and meaningful evaluation of MRs.

4.3.3 Applying Enhanced Criteria to GPT-4-Generated MRs

To further evaluate the quality of MRs generated by GPT-4, the author extended the range of SUTs to nine target systems, as summarized in Table 4.14. The table presents the primary function of each system, the main inputs they receive, and the outputs that they generate. The systems range from basic computational functions, like the 'sine program' and 'sum program', to more complex AI/ML-driven systems such as 'av-perception' and 'AD systems parking function'. This classification is grounded in the need to systematically differentiate the SUTs based on the levels of computational complexity they exhibit and the extent to which they incorporate AI methodologies. Further, the author established a GPT Evaluator by integrating $Criteria_{V1}$ into GPT-4's configuration. This integration used a new feature introduced by OpenAI [92], enabling the creation of tailored

Table 4.14: Target systems for testing with ChatGPT

ID	System	Description	Main Inputs	Main Outputs	Category
1	SIN	Computing *sin*	One number	One number	Basic computational functions
2	SUM	Computing *sum*	A list of numbers	One number	Basic computational functions
3	SHORTEST-PATH	Finding the shortest path	A graph with vertices, edges	A path	Basic computational functions
4	REGRESSION	Multiple linear regression	Data rows	Coefficients, predicted data	Complex systems without AI
5	FFT	Fast Fourier Transform-based analysis	Time-series data	Frequencies, amplitudes	Complex systems without AI
6	WFS	Weather forecasting system	Multiple sources	Multiple outputs	Complex systems without AI
7	AV-PERCEPTION	Autonomous vehicle perception	Images, point clouds	Object-detection	Complex systems with AI
8	TRAFFICSYS	AI-based traffic light control	Sensor data	Traffic decisions	Complex systems with AI
9	AUTOPARKING	Autonomous vehicle parking	Location, obstacles	Parking trajectory, decisions	Complex systems with AI

ChatGPT versions for specific applications. Such customization in ChatGPT ensures enhanced accuracy and efficiency, requiring fewer prompts to generate desired results [92].

1. Basic Computational Functions (Systems 1-3):

   These SUTs represent fundamental and deterministic algorithms. They are used for assessing the GPT models' ability to generate accurate and logical MRs in straightforward computational contexts.

2. Complex Systems without AI Integration (Systems 4-6):

   This category includes systems that perform complex data processing and analysis, yet do not incorporate AI algorithms. They are used for evaluating the GPT models in scenarios involving advanced numerical methods and data interpretation, which require a deeper understanding of mathematical and statistical concepts.

3. Complex Systems with AI Integration (Systems 7-9):

   The selection of SUTs with AI integration is intended to assess the GPT models' effectiveness in generating MRs for systems that are inherently non-deterministic and driven by data. Generating MRs for such systems is usually challenging for human testers.

4.3.3.1 Prompt Methods

Since the previous experiments showed that the GPT model would give better results when asked to generate a small number of MRs (Section 4.3.1.2), the author asked GPT-4 to generate eight MRs for each SUT in this follow-up study. The prompting method included specifying the main inputs and outputs of the programs (as presented in Table 4.14) to the GPT model. This made it possible to tailor its responses based on the provided information, enabling it to understand the functional scope and the data types it was dealing with [151]. For instance, knowing that the inputs were images and point clouds and the outputs were object-detection results, the model could generate MRs that are more relevant to image-processing and object-detection tasks. This focused approach minimized the possibility of generating irrelevant or generic MRs, thereby improving the overall quality and applicability of the MRs for the given SUT. It is important to note that all prompts used in the experiments followed the same structure: they first specify the number of MRs to be generated, followed by details of the inputs and outputs of the SUT. An example of such a prompt is as follows:

> "Generate eight MRs for an ADS perception program. The main inputs are images and point clouds, and the main outputs are object-detection results."

4.3.3.2 Configuring a GPT-MR Evaluator

To eliminate the bias from humans and provide more objective evaluations, a custom GPT evaluator was created, which, in conjunction with human evaluators, offered a detailed comparison of AI and human capabilities in MR evaluation. The configuration of a GPT Evaluator is a straightforward and structured process. The detailed configurations are outlined as follows:

1. **Define the GPT Evaluator's role:**

   The GPT's role is defined as an MR Evaluator in the context of software testing. This role primarily involves evaluating MRs using $Criteria_{V1}$ proposed in Section 4.3.2. The evaluation process is characterized by a binary scoring system for each criterion, where meeting the criterion results get its corresponding points and failure to meet it results in zero points. An example of how the GPT Evaluator was configured in this study is provided below:

   > *In your role as GPT-MR Evaluator, you focus on evaluating metamorphic relations (MRs) in software testing, using a set of criteria. Each MR is assessed based on input relation and output relation aspects. The scoring for each criterion is binary: if the criterion is met, the MR earns the full points for that criterion; if not, it scores zero.*

2. **Incorporate the updated evaluation criteria:**

   The revised evaluation criteria are to be inputted directly beneath the role definition. This ensures that the criteria are clear, accessible, and directly linked to the evaluator's role.

3. **Set answer formats and components:**

   The format for presenting evaluations needs to be specified. The evaluator is expected to begin with a summary table displaying scores for each criterion, followed by a comprehensive justification for these scores. Clarity and precision in the explanations are emphasized. An example used in this study is provided below:

   > *Your evaluations should start with a summary table of scores for*

each criterion, followed by detailed justifications. Maintain clarity and precision in your explanations, aligned with the latest software testing standards.

Following this configuration approach ensures that the GPT Evaluator is effectively prepared to assess MRs. Figure 4.1 shows an example of the evaluator's outcomes and explanations. The evaluators provided scores for each MR under each criterion, along with detailed justifications for their results. This allows the human to understand the results and make their judgements.

4.3.3.3 Results Analysis Methods

In this second evaluation, six human experts were selected for the evaluation work, based on their software testing expertise, contributions to the MT field, and familiarity with the SUTs (total h-index: 139; 163 publications related to MT). After collecting data from human experts, the average scores for SUTs were calculated in each category, rounding to the nearest whole number when necessary. This method effectively balances the inputs of all evaluators, which would accommodate minor discrepancies. Rounding preserves the integrity of the original evaluation scale, ensuring that final scores are both interpretable and meaningful.

The entire process enabled a comparative analysis between the evaluations conducted by human experts and the GPT Evaluator, highlighting similarities and differences in the assessment of MRs. It would provide a comprehensive and objective perspective on the quality of the MRs. Furthermore, by analyzing the scoring tendencies of both human evaluators and the GPT, the unique preferences and characteristics in each evaluator's approach to assessment could be summarized.

You

Evaluate the following MRs for a ADSs parking program.

MR:

a. Vehicle Size Variation MR: Changing the size of the vehicle (e.g., switching between a compact car and an SUV) should result in appropriate adjustments in the parking strategy, such as choosing a larger parking spot for the SUV.

b. Parking Space Orientation Change MR: Rotating the orientation of the parking space (e.g., from parallel to perpendicular) should lead to a corresponding change in the parking maneuver executed by the ADS.

c. Surrounding Vehicle Adjustment MR: Shifting the position of surrounding vehicles slightly should result in minor adjustments in the parking maneuver to accommodate these changes while still successfully parking.

d. Sensor Noise Introduction MR: Introducing a controlled amount of noise to the parking sensors should degrade the parking performance predictably, without leading to significant errors like collisions or failure to park.

e. Parking Area Scaling MR: Changing the size of the parking area (e.g., making it larger or smaller) should result in the ADS adjusting its parking strategy to fit the available space.

f. Obstacle Introduction MR: Introducing new obstacles (e.g., cones or carts) in or around the parking space should result in the ADS either choosing a different parking spot or adjusting its maneuver to avoid the tacles.

g. Lighting Condition Variation MR: Varying the lighting conditions (e.g., parking

(a) Asking the GPT Evaluator to evaluate MRs

GPT MR Evaluator

Evaluation Summary Table

MR	Completeness	Correctness	Applicability	Novelty	Clarity	Computationa Feasibility
a	1	3	3	2	3	2
b	1	3	3	2	3	2
c	1	3	3	2	3	2
d	1	3	3	2	3	2
e	1	3	3	2	3	2
f	1	3	3	2	3	2
g	1	3	2	2	3	2
h	1	3	2	2	3	2

Detailed Justifications

Completeness

All MRs are complete, encompassing necessary input-relation and output-relation

(b) The GPT Evaluator gave evaluation results and justifications for the MRs

Figure 4.1: Sample evaluation of new MRs using the GPT Evaluator

Table 4.15: MRs for SIN systems under the basic computational functions category generated by GPT-4

MR Type	Description
Additive Angle	If the input is x, the output is $\sin(x)$. For a new input $x+\pi$, the output should be $-\sin(x)$. This tests the periodicity and symmetry of the sine function.
Subtractive Angle	For an input x, the output is $\sin(x)$. For a new input $x-\pi$, the output should be $-\sin(x)$. This tests the sine function's bahaviour under angle subtraction.
Multiplicative Angle	If the input is x, the output is $\sin(x)$. For a new input $2x$, the output should follow the identity $2\sin(x)\cos(x)$, allowing testing of the sine function over angle doubling.
Half-Angle	For an input x, the output is $\sin(x)$. For a new input $x/2$, the output should be the square root of $(1-\cos(x))/2$, testing the sine function's bahaviour under half-angle conditions.
Negative Angle	If the input is x, the output is $\sin(x)$. For a new input $-x$, the output should be $-\sin(x)$, testing the odd function property of sine.
Complementary Angle	For an input x, the output is $\sin(x)$. For a new input $\pi/2-x$, the output should be $\cos(x)$, testing the complementary angle identity.
Angle Invariance	If the input is x, the output is $\sin(x)$. For a new input $x+2\pi$, the output should be the same as $\sin(x)$, testing the periodic nature of the sine function over a full period.
Reflection	For an input x, the output is $\sin(x)$. For a new input $\pi-x$, the output should be $\sin(x)$, testing the reflection symmetry of the sine function about $\pi/2$.

4.3.3.4 Evaluating MRs for Basic Computational Functions

Tables 4.15, 4.16, and 4.17 present the MRs created by GPT-4 for the SUTs within the basic computational function category. Each table is organized into two primary columns: "MR Type" and "Description", which were both generated by GPT. The "MR Type" column classified each MR into specific categories, whereas the "Description" column provided a comprehensive description for each MR.

Table 4.16: MRs for SUM systems under the basic computational functions category generated by GPT-4

MR Type	Description
Additive Constant	If the input is a list of numbers [a, b, c, ...], the output is their sum S. For a new input where each number in the list is increased by a constant k, the output should be $S+n\cdot k$, where n is the number of elements in the list. This tests the program's ability to handle uniform increments.
Subtractive Constant	For an input [a, b, c, ...] with sum S, for a new input where each number is decreased by a constant k, the output should be $S-n\cdot k$. This tests the program's handling of uniform decrements.
Element Duplication	If the input list is [a, b, c, ...] with sum S, duplicating any element (e.g., [a, b, c, ..., a]) should result in a new sum of $S+a$. This tests handling of repeated elements.
List Concatenation	Given two lists with sums S_1 and S_2, concatenating them should result in a sum of S_1+S_2. This tests the program's handling of concatenation.
Reverse Order	If the input list [a, b, c, ...] results in sum S, reversing the order should still result in the same sum S. This tests order-agnostic sum calculation.
Element Removal	For a list [a, b, c, ...] with sum S, removing any element (e.g., removing b) should result in $S-b$. This tests response to element removal.
Zero Element Addition	Adding zero to the list ([a, b, c, ..., 0]) should not change the sum. This tests handling of neutral elements.
Negative Element Addition	If the input list is [a, b, c, ...] with sum S, adding a negative number $-d$ should result in $S-d$. This tests handling of negative numbers.

Table 4.17: MRs for SHORTEST-PATH systems under the basic computational functions category generated by GPT-4

MR Type	Description
Edge Weight Increase	For a given graph, if the shortest path is found between two vertices, increasing the weight of one or more edges not in the shortest path should not change the shortest path. This tests handling of irrelevant edge weight changes.
Edge Weight Decrease	If the shortest path is identified, decreasing the weight of edges not part of this path should not affect the path. This tests handling of non-critical edge weights.
Adding Vertex and Edges	Adding a new vertex and edges should not change the shortest path between two vertices unless the new edges create a shorter path. This tests adaptability to graph expansion.
Removing Non-Critical Edge	Removing an edge not part of the shortest path should not change the shortest path. This tests handling of edge removal in non-critical areas.
Path Invariance with Vertex Duplication	Duplicating a vertex with the same connections and weights should not change the shortest path between two original vertices. This tests robustness against graph restructuring.
Reversing Path Direction	The shortest path from vertex A to vertex B should be the same as from B to A in terms of distance, though the actual path may reverse. This tests handling of undirected graphs.
Edge Subdivision	Subdividing an edge (replacing it with two edges whose weights sum to the original edge's weight) should not change the shortest path. This tests handling of graph granularity changes.
Combining Edges	Combining two consecutive edges in the shortest path into a single edge should not change the shortest path. This tests handling of edge aggregation.

Table 4.18 shows the average values for the evaluation results for the MRs from both human experts and the GPT model. The evaluation of MRs for the SIN, SUM, and SHORTEST-PATH programs by human experts and GPT reveals both agreements and discrepancies across various criteria.

Table 4.18: MR evaluation results of basic computational functions from human evaluators and the GPT model

Evaluation Criteria	SIN Human/GPT	SUM Human/GPT	SHORTESTPATH Human/GPT
Completeness	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0
Correctness	2.4 / 3.0	2.0 / 3.0	2.7 / 3.0
Generalizability	3.0 / 3.0	2.9 / 3.0	2.5 / 2.0
Novelty	1.0 / 3.0	1.0 / 1.0	1.8 / 1.5
Clarity	2.9 / 3.0	3.0 / 3.0	1.9 / 3.0
Computational Feasibility	2.9 / 3.0	3.0 / 3.0	2.0 / 2.0
Applicability	3.0 / 3.0	3.0 / 3.0	2.5 / 3.0
Total	16.3 / 19.0	15.9 / 17.0	14.3 / 15.5

In the category of "completeness", both human experts and GPT uniformly scored '1' for all programs, indicating that all essential components—source test case, input relation, and output relation—are included in the MRs. This suggests that all evaluators (including GPT) agreed that the MRs contain the elements required to be considered complete. However, the “correctness” scores show a notable difference: The GPT consistently awarded a score of ‘3’, implying that the MRs captured the intended behaviours of the SUT without violating specifications and maintained a clear focus on the relations among multiple test cases. In contrast, human experts gave relatively low results for the correctness of MRs. From the perspective of human experts, the majority of MRs closely aligned with the basic arithmetic operations of the functions, which was considered as a drawback because it simply replicates the specifications of the SUTs. Consequently, these MRs did not achieve full marks for the “overall correctness” criterion within the broader “correctness” category.

The “generalizability” criterion scores were similar for both human experts and GPT across all programs. For the SIN and SUM programs, all evaluators gave (almost) full marks, while for the SHORTESTPATH program, both human evaluators and GPT considered some MRs were not universally generalizable. The results indicate a shared perception that most MRs were universally fit for all types of SUTs within the same category, while some could only be applied within certain groups of SUTs.

The evaluation of the “novelty” criterion presented a clear contrast for the SIN program, where GPT’s rating was significantly higher. Both humans and GPT provided the same score for the SUM program, indicating agreement on a low level of novelty. A minor difference in the marks is found in the SHORTEST-PATH program, with human experts assigning a slightly higher score than GPT. The main reason for the “novelty” criterion getting overall low marks from both human experts and GPT is that the MRs were based on fundamental arithmetic operations, which are not novel in the context of mathematical computations.

The “clarity” of the MRs assessed by GPT indicates that they were accessible to a general audience, as reflected by the consistent score of ‘3’ across all programs.

Human experts aligned with this assessment for the SIN and SUM programs, believing that the MRs were easily understood without specialized knowledge. However, for the SHORTEST-PATH program, human experts offered a lower score, which means that they found the MRs require basic knowledge of the domain to understand and apply.

The “computational feasibility” criterion was rated highly by both human experts and GPT for the SIN and SUM programs, reflecting an ease of generating and automating test cases. However, for the SHORTEST-PATH program, both results indicated a reduced feasibility, pointing to a more complex generation or validation process that might require more computational resources, or that could not be fully automated.

Finally, GPT gave all programs full marks on the “applicability” criterion, aligning with the assessments of human experts for the SIN and SUM programs, which also received top marks. However, the SHORTESTPATH program was rated slightly lower. This indicates a consensus that the MRs focused on and highlighted key features of the corresponding SUTs, with input and output relations specifically tailored to showcase the distinctive features and behaviours of the systems.

In summary, the evaluation results revealed that both humans and GPT consistently recognized the MRs’ completeness, generalizability, and applicability of the SUTs. Differences arise in the evaluation of the correctness, novelty, and clarity of the MRs, with GPT generally assigning higher scores for correctness and novelty. Conversely, human evaluators exhibited a more critical attitude, particularly regarding the novelty of the MRs. These differences highlight the areas where human evaluators and GPT may prioritize different aspects of the MRs.

Table 4.19: MRs for REGRESSION systems under complex systems without AI integration generated by GPT-4

MR Type	Description
Data Scaling MR	If the input data rows are scaled by a constant factor, the coefficients should adjust accordingly to produce the same predicted data. Tests uniform data scaling.
Data Shifting MR	Shifting data rows by adding a constant value should result in an adjustment of the intercept coefficient, while other coefficients remain unchanged. Tests response to data shifts.
Feature Addition with Zero Weight MR	Adding a zero-valued feature should not change the coefficients or predicted data. Tests robustness to irrelevant feature addition.
Duplicate Data Row MR	Duplicating data rows should not fundamentally change the coefficients. Tests handling of data redundancy.
Removing Irrelevant Feature MR	Removing a negligible coefficient feature should minimally impact other coefficients and predicted data. Tests adaptability to feature reduction.
Permuting Data Rows MR	Changing the order of data rows should not affect coefficients or predictions. Tests order irrelevance in regression analysis.
Combining Dependent Features MR	Combining linearly dependent features should result in predictable coefficient changes and consistent predicted data. Tests handling of multicollinearity.
Inverse Data Transformation MR	Applying inverse transformation to predicted data should align with original scale predictions. Tests consistency across data transformations.

4.3.3.5 Evaluating MRs for Complex Systems Without AI

Tables 4.19, 4.20, and 4.21 display the MRs generated by GPT-4 for the SUTs categorized under complex systems without AI integration.

Table 4.22 shows the average values for the evaluation results for the MRs from both human experts and the GPT model. The evaluations of MRs for REGRESSION, FFT, and WFS systems by human experts and the GPT model showed a clear consensus on the structural integrity of the MRs to the system's key features, as reflected in the agreement on the MRs' completeness found in both sets of results.

However, when it comes to the "correctness" criterion, human experts assigned lower marks compared to GPT, reflecting a nuanced understanding of the MRs' accuracy. For instance, the 'Data Shifting MR' in the REGRESSION program was noted as partially correct by human experts: While the MR correctly states that shifting predictor values impact the intercept—the constant term in the regression equation representing the expected value of the dependent variable when all predictors are zero—it oversimplifies the influence on other coefficients.

Table 4.20: MRs for FFT systems under complex systems without AI integration generated by GPT-4

MR Type	Description
Time Scaling MR	Expanding or contracting the time scale should inversely scale frequencies while maintaining amplitudes. Tests time scaling in data.
Amplitude Scaling MR	Scaling input amplitude should proportionally scale output amplitudes without affecting frequencies. Tests amplitude sensitivity in FFT analysis.
Data Shifting MR	Shifting time-series data should not affect frequencies and should impact only the zero frequency amplitude. Tests handling of DC shifts.
Time Reversal MR	Reversing time-series data should yield the same frequencies and amplitudes. Tests response to time-reversed data.
Data Concatenation MR	Concatenating time-shifted data should result in the same frequencies with amplitude changes. Tests data concatenation handling.
Zero Padding MR	Zero padding should not change fundamental frequencies but may increase resolution. Tests FFT consistency with zero padding.
Frequency Domain Filtering MR	Applying a filter and inverse FFT should result in predictable time-domain changes, reflecting the filter's characteristics.
Harmonic Addition MR	Adding a harmonic should result in detection of the additional frequency with corresponding amplitude. Tests harmonic detection capability.

Table 4.21: MRs for WFS systems under complex systems without AI integration generated by GPT-4

MR Type	Description
Data Source Consistency MR	Same weather data from different sources should result in consistent forecasts. Tests data source consistency.
Temporal Shift MR	Shifting input data in time should result in a corresponding forecast shift. Tests handling of time-shifted data.
Data Scaling MR	Scaling input data should result in predictable output changes. Tests response to uniformly scaled data.
Data Omission MR	Omitting a data subset should degrade forecast quality predictably but not lead to different patterns. Tests resilience to incomplete data.
Cross-Parameter Consistency MR	Changes in one parameter should result in predictable changes in related forecasts. Tests internal consistency in handling related parameters.
Data Addition MR	Adding new data sources should enhance accuracy without contradicting previous forecasts. Tests integration of additional data.
Historical Data Validation MR	Inputting historical data should align forecasts closely with actual outcomes. Tests accuracy against known events.
Location Shift MR	Shifting input data's geographical location should result in an appropriate forecast for the new location. Tests geographical adaptability.

Table 4.22: MR evaluation results of complex programs without AI integration from human evaluators and the GPT model

Evaluation Criteria	REGRESSION Human/GPT	FFT Human/GPT	WFS Human/GPT
Completeness	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0
Correctness	1.9 / 3.0	2.6 / 3.0	1.6 / 3.0
Generalizability	3.0 / 3.0	2.6 / 2.0	2.9 / 2.9
Novelty	1.9 / 2.0	2.0 / 2.0	2.0 / 2.0
Clarity	2.6 / 3.0	2.1 / 3.0	2.0 / 3.0
Computational Feasibility	1.7 / 3.0	2.0 / 2.0	2.0 / 2.0
Applicability	3.0 / 3.0	2.4 / 3.0	2.9 / 2.9
Total	15.1 / 18.0	14.7 / 16.0	14.3 / 16.8

This is particularly true for complex models where interactions or non-linear transformations are present, which might result in changes to coefficients beyond the intercept. GPT, with a broader but potentially less detailed analysis, awarded full marks.

The “generalizability” scores assigned by both human experts and GPT were comparable across all evaluated programs. In the cases of the REGRESSION and WFS programs, evaluators awarded nearly perfect scores. However, for the FFT program, both humans and GPT noted that some MRs can not be applied universally. This indicates that, although the majority of MRs were adaptable to a wide range of SUTs within the same category, some were suitable only for particular groups of SUTs.

In the “novelty” criterion, both human experts and the GPT model recognized the MRs as having elements of innovation but not as wholly new concepts. GPT rated the novelty slightly higher, suggesting it found more uniqueness in the MRs’ approach to input and output relations, yet not to the extent of reaching the full scores for this criterion. Human experts’ ratings reflected an acknowledgement of some new methods within the MRs, but within the bounds of known MROPs.

In the “clarity” evaluation, the GPT model rated the MRs as being accessible to a general audience, indicating that the MRs were written clearly enough to be understood without specialized knowledge. Human experts assigned lower scores, indicating that while the MRs were clear, they still contained technical terms and complex methodologies that may require a basic understanding of the field. This points to a nuanced difference in interpretation: GPT assumed the MRs were sufficiently self-explanatory for anyone, whereas human experts considered the potential need for a basic level of domain familiarity.

For the “computational feasibility” criterion, both human experts and GPT rated the MRs with medium scores, suggesting that they found the automation of test case generation to be practical, while the validation of the MRs failed to be automated. Neither GPT nor human experts assigned full marks, indicating that they both recognised the MRs' automation potential yet were aware of the limitations that prevent achieving the most efficient level of automation.

Finally, GPT gave all programs (almost) full marks on the "applicability" criterion, aligning with the assessments of human experts for the REGRESSION and FFT programs, which also received full marks. However, human experts rated the FFT program slightly lower, considering some MRs of the FFT program did not highlight the unique features of the SUT. For instance, the transformation of inputs (i.e., reversing time-series data) in the "Time Reversal MR" is a common feature that appears in other MRs as well.

In conclusion, the evaluation of MRs for REGRESSION, FFT, and WFS systems reflected a consensus on their completeness, with both human experts and GPT recognizing the MRs as fully structured and significant to the SUTs' key features. Correctness assessments differ, with human experts raising more concerns about the MRs' details, while GPT was consistently positive. Generalizability was rated highly by both, indicating the MRs' general utility. Novelty was acknowledged by both evaluators, with GPT noting slightly more originality in the MRs' methodologies. For clarity, GPT perceived the MRs as universally understandable, while human experts believed they might require some domain knowledge. Computational feasibility was considered practical for test case generation by both, but not to the highest degree of automation to involve the MR validation. Applicability was overall high for all programs, with human experts being more strict on certain MRs.

Table 4.23: MRs for AV-PERCEPTION systems under complex systems with AI integration generated by GPT-4

MR Type	Description
Image Brightness Adjustment MR	Altering brightness should not significantly change detected objects, testing robustness to lighting variations.
Point Cloud Density Variation MR	Varying point cloud density should not fundamentally change object identification, testing handling of different densities.
Image Scaling MR	Scaling images should result in consistent object detection, testing robustness to image scale changes.
Camera Angle Rotation MR	Rotating camera angle should adjust object orientation in detection without missing or falsely detecting objects, testing camera angle variations.
Partial Occlusion MR	Partially occluded objects should still be detected, testing the ability to handle occlusions.
Synthetic Object Addition MR	Adding synthetic objects should result in their detection, testing the ability to detect new entities.
Background Variation MR	Changing background settings should not affect object detection, testing consistency across environments.
Sensor Noise Introduction MR	Introducing sensor noise should predictably degrade performance without drastic errors, testing resilience to noise.

Table 4.24: MRs for TRAFFICSYS systems under complex systems with AI integration generated by GPT-4

MR Type	Description
Sensor Data Scaling MR	Scaling sensor data should lead to proportional traffic decisions, testing response to traffic density variations.
Time Shift MR	Shifting sensor data time frame should predictably change traffic decisions, reflecting different traffic patterns.
Sensor Data Omission MR	Omitting sensor data should lead to a conservative traffic response, prioritizing safety, testing resilience to incomplete data.
Synthetic Sensor Data Addition MR	Adding synthetic sensor data should appropriately change traffic decisions, reflecting the added data.
Cross-Intersection Data Consistency MR	Consistent traffic patterns at multiple intersections should lead to harmonized traffic decisions, promoting fluidity.
Variable Traffic Pattern MR	Varying traffic patterns should appropriately adjust light durations and sequences, maintaining flow.
Pedestrian Flow Introduction MR	Introducing pedestrian data should influence traffic decisions for pedestrian safety.
Emergency Vehicle Prioritization MR	Detecting emergency vehicles should override regular traffic patterns for prioritization.

4.3.3.6 Evaluating MRs for Complex Systems With AI

Tables 4.23, 4.24, and 4.25 show the MRs generated by GPT-4 for the SUTs categorized under the complex programs with AI integration.

Table 4.26 shows the average values for the evaluation results for the MRs from both human experts and the GPT model.

In terms of the "completeness" criterion, both human experts and the GPT model awarded a score of '1' across all MRs, signifying agreement that the MRs included all necessary components: a well-defined source test scenario, a clear

Table 4.25: MRs for AUTOPARKING systems under complex systems with AI integration generated by GPT-4

MR Type	Description
Vehicle Size Variation MR	Changing vehicle size should appropriately adjust parking strategy, e.g., for larger vehicles.
Parking Space Orientation Change MR	Rotating parking space orientation should lead to a corresponding change in parking maneuver.
Surrounding Vehicle Adjustment MR	Shifting surrounding vehicles should result in minor parking maneuver adjustments.
Sensor Noise Introduction MR	Introducing noise to parking sensors should predictably degrade parking performance without significant errors.
Parking Area Scaling MR	Changing parking area size should adjust the parking strategy to fit the space.
Obstacle Introduction MR	Introducing obstacles should lead to adjusted parking strategies or spot selection.
Lighting Condition Variation MR	Varying lighting conditions should not significantly impair parking ability, assuming visibility.
Surface Texture Variation MR	Changing surface texture should not prevent successful parking, but may adjust approach.

Table 4.26: MR evaluation results of complex programs with AI integration from human evaluators and the GPT model

Evaluation Criteria	AV-PERCEPTION Human/GPT	TRAFFICSYS Human/GPT	AUTOPARKING Human/GPT
Completeness	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0
Correctness	2.1 / 3.0	1.6 / 3.0	2.1 / 3.0
Generalizability	2.6 / 2.9	2.5 / 2.9	2.7 / 2.9
Novelty	1.9 / 2.1	2.0 / 2.1	1.8 / 2.0
Clarity	2.2 / 3.0	1.8 / 3.0	2.3 / 3.0
Computational Feasibility	1.9 / 2.0	1.9 / 2.0	2.0 / 2.0
Applicability	2.6 / 2.9	2.5 / 2.9	2.9 / 2.9
Total	14.4 / 16.9	13.3 / 16.9	14.6 / 16.8

input-relation, and a detailed output-relation.

The "correctness" criterion evaluations showed a contrast: the GPT model assigned a full score of '3' to all MRs, suggesting it found them to accurately reflect the intended behaviours of the systems. On the other hand, human experts were more conservative in ratings, particularly for the TRAFFICSYS and AUTOPARKING MRs, which received significantly lower scores. The primary reason behind this was due to the experts' observations of vague expressions within the output relations and the identification of inaccuracies in some instances. For instance, in the 'Vehicle Size Variation MR' of the AUTOPARKING system, whether the system is changing the parking strategy or not depends on the system specifications. The correctness of the MR depends heavily on the specific features and capabilities of the ADS parking system. If the system is not designed to alter its strategy based on vehicle size, then the MR does not apply. In addition, an effective MR should have clear and specific output relations [178]. The original MR's output, which suggests an adjustment in parking strategy based on vehicle size, lacks specificity. It does not define what constitutes an "appropriate adjustment" or how the strategy changes with different vehicle sizes. Finally, the MR does not provide guidelines on how the parking strategy should change or what degree of change is necessary. For instance, it does not specify what qualifies as a larger parking spot for an SUV or how the system should identify and choose these spots differently compared to spots for compact cars.

In the assessment of the "generalizability" criterion, both humans and the GPT model rated the MRs highly but not perfectly, indicating that the MRs were largely but not universally generalizable. The slightly lower scores from humans suggested a cautious recognition of practical limitations in applying these MRs across all systems. The GPT model gave higher marks, showing a more optimistic view of their broad generalizability, yet also acknowledged some limitations since it did not give full marks. This reflects an understanding from both sides that while MRs were versatile, their application was not without certain restrictions.

For the "novelty" criterion, both human experts and the GPT model provided scores that reflected a recognition of some innovative elements within the MRs, but not to the extent of groundbreaking originality. GPT's marginally higher scores indicated a slightly more favourable view of the MRs' uniqueness.

In assessing the "clarity" criterion, the GPT model rated the MRs with the highest score, implying it found them to be universally understandable. Human experts gave lower scores, suggesting that they found the MRs generally clear but required some specialized knowledge for full comprehension, especially for such SUTs that have AI embedded. The discrepancy may indicate that GPT evaluated the MRs' clarity from a more theoretical standpoint, while human experts might be considering practical nuances that could affect understanding for a broader audience.

For the "computational feasibility" criterion, both human experts and the GPT model gave moderate scores, which implied an agreement on the practicality of applying the MRs, yet with some reservations. GPT's slightly higher scores compared to those of human experts might reflect its extensive training on diverse datasets [151], potentially providing it with a broader knowledge base to evaluate the automation potential of the MRs. GPT could be recognizing more opportunities for efficient automation based on its understanding of existing tools and techniques, which would explain its optimism compared to the more reserved human scores. Human experts, while knowledgeable, might be more aware of practical challenges and constraints that are not as apparent in theoretical data, leading to their more cautious scoring.

Lastly, for the "applicability" criterion, the marks were generally rated highly by both human experts and GPT, although GPT's scores were slightly higher. This indicated a shared view that the MRs were pertinent and focused on key features of the systems, with GPT seeing them as slightly more aligned with the SUT's core functionalities.

In conclusion, the evaluation of MRs for complex systems with AI integration, carried out by both human experts and the GPT model, revealed a shared recognition of their completeness and relevance. However, there were notable divergences in correctness, generalizability, novelty, clarity, and computational feasibility. The GPT model generally showed a more favourable view of the MRs across these criteria, while human experts exhibited a more cautious and critical stance, reflecting a deeper consideration of practical and system-specific aspects. These differences highlighted the varied perspectives and interpretations brought by human expertise and AI analysis [5] in evaluating the effectiveness of MRs.

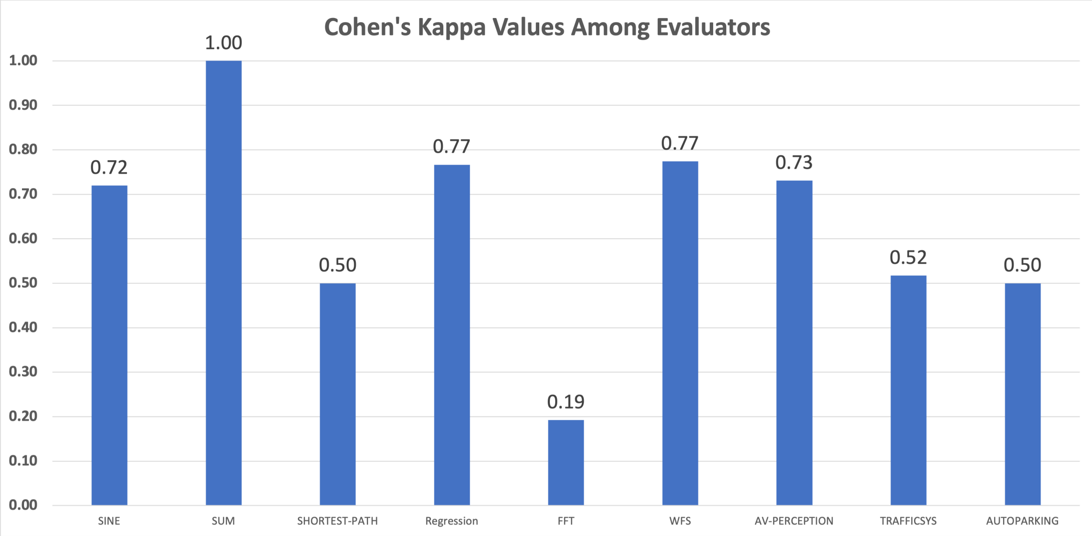


Figure 4.2: Cohen's Kappa agreements between evaluators

4.3.4 Qualities of the MRs Produced by GPT-4

Analyzing the evaluation results of MRs from human experts and GPT across three categories of systems—basic computational functions, complex systems without AI integration, and complex systems with AI integration—reveals some key similarities and differences in their assessment approaches and perspectives on the MRs.

4.3.4.1 Evaluator Agreement Analysis Using Cohen's Kappa

In this section, the levels of human evaluators' agreement on the results of MRs are presented, using a famous statics method, Cohen's Kappa [216]. It is widely adopted in the marking and evaluation works [162, 209, 212]. The calculation adjusts for the possibility that some agreement may occur by chance. The Kappa value ranges from -1 (complete disagreement) to 1 (perfect agreement), with 0 meaning the agreement is purely by chance [216]. For each system, the comparisons were made on how often the evaluators gave the same or different scores across the criteria. Then, the formula adjusts for the expected agreement by chance, resulting in a value that reflects how much the evaluators truly agree.

Table 4.27: Cohen's Kappa evaluation results across the nine SUTs

System	Cohen's Kappa	Interpretation
SINE	0.72	Substantial agreement
SUM	1.00	Perfect agreement
SHORTEST-PATH	0.50	Moderate agreement
REGRESSION	0.77	Substantial agreement
FFT	0.19	Slight agreement
WFS	0.77	Substantial agreement
AV-PERCEPTION	0.73	Substantial agreement
TRAFFICSYS	0.52	Moderate agreement
AUTOPARKING	0.50	Moderate agreement

Cohen's Kappa Interpretation Scale:

- **Kappa >0.80**: Almost perfect agreement
- **Kappa between 0.61 and 0.80**: Substantial agreement
- **Kappa between 0.41 and 0.60**: Moderate agreement
- **Kappa between 0.21 and 0.40**: Fair agreement
- **Kappa between 0.01 and 0.20**: Slight agreement
- **Kappa = 0**: No agreement beyond chance
- **Kappa <0**: Agreement worse than chance

The higher the Kappa value, the stronger the agreement between the two evaluators.

Figure 4.2 shows the results of the Cohen's Kappa agreements across the nine SUTs, where Table 4.27 shows the interpretation of the values. It can be found that the human evaluators were largely in agreement with most systems, except the FFT system. After examination of the cause, the evaluators believed the reasons were due to different familiarity with the systems: One evaluator had greater familiarity with the FFT system, resulting in stricter assessments. The low kappa value of the FFT results highlights real-world variability among evaluators, which supports objectivity in the evaluation. To synthesize these evaluations, the average scores across SUTs were calculated within each category, rounding when necessary. This averaging approach balances differences across evaluators while preserving the integrity of the original scale, making the final scores interpretable and reflective of a comprehensive assessment.

4.3.4.2 Summary of the MR Qualities

Among the MRs generated by GPT-4 for the nine systems across three categories, there are some common aspects based on the evaluation results:

1. **Completeness and applicability of the MRs:**
   The MRs across all system types were structured to include all necessary components, such as source test cases, input relations, and output relations. They effectively focused on and highlighted key features and behaviours of the SUTs.

2. **Broad generalizability of the MRs:**
   The MRs demonstrated wide generalizability across various systems. This trait was shown in both basic and complex systems, including those with AI components.

3. **Correctness of the MRs:**
   The MRs generally provided accurate representations of intended system bahaviours, a critical aspect recognized across various types of systems. However, there was a noted variation in the perceived level of correctness, particularly in complex and AI-embedded systems. To achieve higher levels of correctness, it was recognized that the MRs needed more clearly defined constraints to make the scenarios more accurate. These constraints were essential for ensuring that the MRs were not only correct but also precise and directly applicable to the bahaviours and specifications of the SUTs. This need for enhanced specificity and constraint was especially important for complex systems, where missing details could cause the MRs to violate

the specifications of the SUTs.

4. **Novelty and clarity in the MRs:**

   The novelty of MRs across different system categories was generally perceived as moderate rather than high. In basic math programs, the MRs were often based on fundamental arithmetic operations, which did not present a high level of novelty due to their foundational nature in mathematics. In complex systems, while some elements of innovation were acknowledged, the MRs were primarily viewed as extensions or adaptations of pre-existing concepts in the literature or human knowledge, rather than entirely novel ideas. This trend suggested that while the MRs incorporated new methods or perspectives in their approach to input and output relations, they primarily built upon established testing paradigms and techniques. The clarity of the MRs was generally high, but understanding them often required specific domain knowledge, especially in more complex applications.

5. **Computational feasibility of the MRs:**

   The computational feasibility of implementing the MRs varied across system categories. In basic computational functions, the MRs were generally straightforward to implement and automate due to the simplicity of the systems and the clarity of the operations. This allowed for efficient generation and validation of test cases, making the MRs highly practical in these contexts. However, as the complexity of systems increased, the computational feasibility of the MRs decreased. In these scenarios, the generation and automation of test cases became more challenging. The complexity inherent in these systems often requires more sophisticated algorithms and computational resources [28]. This was particularly true for systems with AI, where the unpredictability and intricacy of AI bahaviours [98] complicated the automation process. While the potential for automating test case generation based on MRs was acknowledged, the evaluations indicated that automating the entire MT process for the MRs was difficult and currently limited by technology.

In summary, the MRs generated by GPT-4 had strengths in completeness, relevance, and broad generalizability, demonstrating the model's adeptness in generating useful MRs for various systems. However, the correctness of these MRs, particularly in complex environments, highlighted areas for improvement through the inclusion of more detailed and specific constraints. The novelty aspect of these MRs stood out as a blend of innovation and established methodologies, indicating GPT-4's capability to integrate new ideas within traditional frameworks, though suggesting room for further originality. The computational feasibility and system-specific design of these MRs varied with system complexity, reflecting both the capabilities and limitations of GPT-4 in generating contextually appropriate and effective testing strategies. Therefore, while GPT-4's MRs were useful in system testing, particularly in standard applications, their effectiveness in complex or novel scenarios could have been enhanced through more focused development and refinement. It is also important to note that, despite these encouraging findings, outputs from LLMs like GPT-4 should primarily be viewed as suggestions, which means they do not remove the need for human oversight and judgment, especially in MR generation tasks that involve complex situations and specialized domain expertise.

4.3.4.3 Observations of the GPT and Human Evaluators

By summarizing the marks from the previous sections, it can be found that human evaluators tended to focus more on the practical and intricate aspects of MRs. They were critical and detail-oriented, especially regarding system-specific correctness and practical feasibility. GPT evaluation reflected a broader, possibly less detailed perspective. It tended to be more optimistic in its assessments, particularly regarding novelty, clarity, and computational feasibility.

4.3.5 Discussion: GPT as an MR Evaluator

4.3.5.1 Strengths and Weaknesses

This study highlights several areas where GPT demonstrates notable strengths over human evaluators in MR assessment. Firstly, GPT exhibits an impressive capacity for rapidly evaluating a diverse range of MRs. This is particularly advantageous for large-scale evaluations, where the volume of MRs would be impractical for human evaluators to assess within a reasonable timeframe.

Additionally, GPT's evaluations are characterized by remarkable consistency, free from the biases and fatigue that may affect human evaluators. In terms of novelty, clarity, and computational feasibility, GPT consistently delivered objective and broad assessments. This capability of GPT could be especially beneficial in the initial screening processes of MRs, where quick and extensive evaluations are necessary.

However, the experiment results also revealed limitations in GPT's evaluation, especially in its capacity to examine the details of MRs. A recurring observation was GPT's tendency to overlook the details of MRs when evaluating the correctness criterion, often resulting in higher scores. This tendency was particularly evident when evaluating MRs for large, complex systems, both with and without AI/ML components.

In contrast, human evaluators consistently demonstrated a more critical and detail-oriented approach, emphasizing the need for additional constraints to ensure MR correctness. This discrepancy indicates that while GPT can provide broad assessments, it sometimes struggles with the detailed and context-specific aspects that are crucial in evaluating complex systems—areas where human evaluators currently show greater adeptness.

However, it is important to note that such ability of human evaluators is largely

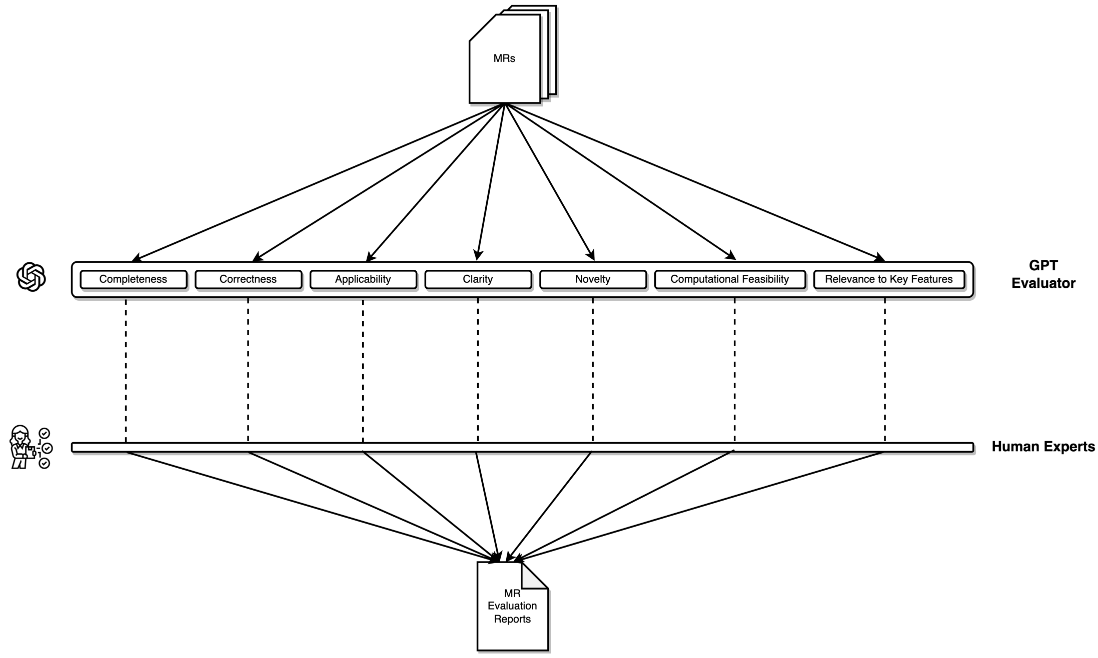


Figure 4.3: Human experts and ChatGPT work together to evaluate MRs

dependent on their specific knowledge bases. In scenarios where human experts are not familiar with certain systems, GPT's evaluations, benefiting from its extensive training data encompassing a wide range of contexts, can be more advantageous. In such cases, GPT can provide insights or identify potential issues that might not be immediately apparent to human evaluators with limited exposure to the specific domain. This highlights the potential of GPT as a complementary tool in MR evaluation, particularly in unfamiliar or less-explored domains for human evaluators.

4.3.5.2 Complementary Roles of GPT and Human Evaluators

Given these strengths and limitations, the author proposed a synergistic framework combining the capabilities of GPT with human expertise. Although GPT is capable of effectively handling the initial, high-level assessment of MRs, its tendency to overlook details requires a more thorough review by human evaluators. This is especially important for complex systems, where the depth and specificity of human judgment are essential to identify and rectify potential oversights by

GPT.

Therefore, as Figure 4.3 shows, GPT could serve as a first-line evaluator, rapidly assessing and filtering MRs based on broad criteria such as novelty and computational feasibility. Subsequently, human evaluators, with their attention to detail and critical perspective, could then thoroughly assess these MRs, ensuring that additional constraints are identified and applied to guarantee certain aspects, such as correctness, particularly in large and complex test cases. This hybrid approach not only maximizes efficiency but also leverages the unique strengths of both AI and human intelligence, leading to a more robust and comprehensive evaluation process.

4.3.5.3 Areas for improvement

The observed tendency of GPT to overlook MR details and overestimate correctness highlights a need for improvement in both the evaluation criteria and GPT configurations. To better align with human evaluators, GPT could benefit from integrating more sophisticated, context-aware evaluation metrics or training on more diverse and complex datasets. Further, integrating algorithms or configurations that mimic human-like critical thinking and attention to detail could improve its evaluative accuracy.

4.3.6 Limitations of the Current Study

This follow-up study's exploration into GPTs' ability to generate MRs for various SUTs is set against the fast development of GPT versions. While recognizing that each version of GPT might have a better performance, this rapid advancement does not diminish the relevance and significance of the current research findings. Rather, the current analysis of GPT-4's capabilities offers valuable benchmarks in the domain of MR generation. It is important to note, however, that although

MRs generated by GPT could potentially be improved with more expertly crafted prompts, this approach requires more specialized knowledge and a higher level of proficiency in using GPT [139]. Therefore, it falls outside the scope of the discussion. The studies discussed in this chapter aim to explore the performance of GPT in generating MRs with fewer prompts, providing a more focused evaluation of ChatGPT's capabilities.

Although MRs generated by GPT-4 demonstrated strengths in completeness, generalizability, and clarity, their limitations became evident when applied to complex or AI-driven systems. In such contexts, the generated MRs often lacked the necessary specificity and constraints required to ensure correctness and real-world applicability. Furthermore, the level of novelty remained moderate, with many MRs reflecting variations of existing ideas. These limitations suggest a potential overreliance on AI-generated outputs, which may require critical human oversight, especially in areas where specialized knowledge is crucial. To address this, future work could focus on combining AI generation with careful human review, structured refinement, and prompt design based on domain expertise. Additionally, incorporating example-driven fine-tuning or post-processing steps could also enhance the generated MRs more closely with the requirements of complex systems to make them not only technically robust but also contextually relevant.

Despite the rapidly evolving nature of specific GPT models like GPT-4, the methodologies and evaluation criteria developed in this study hold significant and lasting value. These methods offer a robust framework for assessing the generation of MRs in both current and future iterations of GPT models. As newer versions of GPT are released, the established criteria provide a consistent and reliable benchmark for comparison. This study, by evaluating the capabilities and limitations of GPT-4, offers critical findings that are immediately applicable. It also lays the groundwork for assessing future advancements in LLM technology. Additionally, the results emphasize the importance of improving the training approaches for LLMs, particularly in enhancing the novelty of the MRs. The methodologies introduced in this research can further serve as guidance for training students to effectively communicate with LLMs and select MRs, thereby optimizing the practical utility of these tools. Consequently, this research provides both a valuable assessment of GPT-4's current capabilities and a starting point for future explorations and improvements in LLM-driven MR generation.

4.3.7 Conclusion

The studies in this chapter conducted a comparative analysis of the MRs generated by GPT-3.5 and GPT-4, using established and then updated evaluation criteria. The initial comparison explicitly demonstrated GPT-4's advantages in generating high-quality MRs compared to GPT-3.5. Further, applying the refined criteria to a broader range of nine SUTs, including both simple and AI/ML complex systems, provided a more comprehensive and thorough analysis of the quality of the MRs generated by GPT-4. A novel aspect of this study was the use of both a customized GPT evaluator and human evaluators, offering comparative insight into the capabilities of AI versus human assessment of the MRs.

The findings highlight GPT-4's advanced capabilities in software testing and MR generation across various applications. They underscore the evolving capabilities of AI in software testing, especially in generating and assessing MRs, while also emphasizing the indispensable role of human expertise in critical and detail-oriented evaluation processes.

In the next chapter, ChatGPT was used to test the CARLA simulator, producing MRs that helped reveal defects. Furthermore, Chapter 6 compares GPT-generated MRs to those created by students with limited training, analyzing ChatGPT's role in supporting beginners to enhance the testing process. The

GPT-MR evaluator was also used to assess the MRs, providing valuable insights for enhancing teaching and training in MT.

Chapter 5

Advancing Autonomous Driving Simulator Testing through Metamorphic Testing

Publications related to this chapter

1. **Y. Zhang**, D. Towey, M. Pike, J. C. Han, Z. Q. Zhou, C. Yin, Q. Wang, and C. Xie, "Scenario-Driven Metamorphic Testing for Autonomous Driving Simulators", *Software Testing, Verification and Reliability*, vol. 36, Art. no. e1892, Jul. 2024. doi: 10.1002/stvr.1892.

2. **Y. Zhang**, D. Towey, and M. Pike, "Enhancing Autonomous Driving Simulations: A Hybrid Metamorphic Testing Framework with Metamorphic Relations Generated by GPT", submitted to *Information and Software Technology*, **under review**.

5.1 Introduction

As mentioned at the end of Chapter 3, identifying the origins of anomalies in ADS testing can be challenging [249]. Therefore, ensuring the validity of AD simulators is essential to the overall testing process of ADSs, an area that remains relatively underexplored in MT testing, and serves as a solution to help address *RQ1* of the thesis (*How can MT be effectively applied in ADS testing to uncover anomalies and enhance system understanding?*).

The chapter begins with an examination of MT on the NIO[1] AD simulator [249]. The simulator had not undergone MT previously, and the author employed scenario-based testing instead of conventional unit testing [56] to identify defects across various software components. The MT experiments revealed a significant number of defects that were largely undetected by conventional testing methods, most of which were related to scenario parsing and system logic, with many classified as high priority, highlighting the effectiveness of MT in identifying critical issues.

During the testing, the author proposed a set of MRPs and MRIPs to facilitate the generation of MRs. There were few MRPs/MRIPs that specifically target AD-related systems in the literature; and these systems often exhibit a higher level of complexity compared to other types of SUTs, such as language translation programmes or navigation software [174][219][229]. This increased complexity comes from the incorporation of multiple interconnected modules, intricate sensor integration, sophisticated decision-making algorithms, and the need to handle real-time data processing in diverse and dynamic driving scenarios [19]. To alleviate the challenges faced by practitioners unfamiliar with MT or ADSs and struggling with the abstract nature of MRPs, each MRP is accompanied by an

[1]NIO is a Chinese company specializing in the design, development, manufacturing, and sale of premium smart electric vehicles. Since its founding in November 2014, NIO has focused on developing and investing technologies in AD, digital systems, electric powertrains, and batteries [2].

MRIP. This addition aims to provide further guidance and support for effectively generating MRs. The MRs generated from these MRPs and MRIPs have been empirically demonstrated to effectively reveal defects within the AD simulator, which is crucial as the validation of the simulators is frequently overlooked during the validation process of ADS functions in the existing literature [223].

In addition, a scenario-driven MT framework for scenario testing that integrates ME and MT processes was proposed. The framework incorporates the complete testing cycle (from test-case generation to test-result validation). The process of implementing the framework is self-evolving, which means that the tester can iteratively improve the test cases (i.e., scenarios) and MRs, as well as enhance their understanding of the system, until a satisfactory result is achieved. This approach can significantly reduce the time and effort required for testers to prepare for testing a new system. An industry case study was also provided to highlight the strengths and limitations of this framework.

The successful outcomes of MT on the NIO AD simulator served as a motivation for subsequent experiments. Since the MT was conducted on the simulator in conjunction with the NIO ADS, one key challenge was identifying the root cause of MR violations: Whether they originated from the simulated data or the ADS algorithms [249]. To mitigate the influence of the ADS and better evaluate the effectiveness of MT in AD simulators, the author conducted MT experiments on the CARLA simulator, a widely adopted open-source platform [66].

Additionally, since Chapter 4 shows the effectiveness of ChatGPT in generating MRs, a human-AI hybrid MT framework was proposed to enhance MR generation and testing through the integration of LLMs. This framework combines human inputs with AI-driven automation to generate and refine MRs specifically tailored to assess various aspects of the CARLA simulator. Central to this is an open-access GPT-MR generator, a customized GPT configured using the MRPs proposed in testing the NIO simulator [249], designed to generate MRs according to user specifications. This innovation was made possible by OpenAI's newly introduced feature called *GPTs*, which allowed users to personalize ChatGPT for particular tasks without the need for coding expertise [92]. These tailored GPTs can incorporate specialized knowledge and perform designed functions, thereby enhancing the adaptability and utility of AI applications.

As shown in the experiments of evaluating the GPT-generated MRs (Section 4.3), these GPT models can produce correct MRs while still leaving room for enhancement, such as novelty and computational feasibility. Therefore, to maximize the effectiveness of MRs derived by the GPT-MR generator, these MRs are then selected and refined by MT experts. The framework also includes a test harness, designed to automate and enhance the efficiency of the testing process. This harness automates the creation and execution of test cases, offering an interface where users can input parameters that initiate specific functions. These include generating follow-up test scenarios from a source scenario and executing multiple scenarios in sequence. The harness is also equipped to generate vehicle control sequences that manage the movements of the vehicle within simulations. Furthermore, the parameters enable the activation of four internal modules within the harness, each dedicated to executing and creating scenarios for various MRs.

Experiments with CARLA led to the discovery of four defects of the simulator, highlighting the efficacy and importance of MT in improving the robustness and accuracy of AD simulators, as well as its role in uncovering hidden issues and validating simulation environments. This transformation from testing ADS to AD simulators also underscores the importance of validating simulators to ensure the reliability of ADS testing outcomes.

In conclusion, the findings of the experimental results successfully addressed *RQ1* of the thesis, given the irreplaceable role of the simulations in ADS testing [240]. To alleviate the challenges in generating MRs mentioned in *RQ2* (*How can the difficulty of generating MRs be reduced to assist beginners and testers in under-*

standing and testing new systems?), a set of MRPs/MRIPs were proposed. These patterns, both when used independently and when integrated into an LLM, have proven effective in facilitating the generation of MRs and identifying defects in simulators. For *RQ3* (*How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?*), a scenario-driven MT framework and a human-AI hybrid MT framework were introduced. The former notably reduced testers' preparation time and effort, while the latter automated the MR generation process, enhancing both the efficiency and effectiveness of testing.

5.2 Exploring MT with the NIO AD Simulator

5.2.1 Facilitating MR Generation with MRPs and MRIPs

Due to the complexity of the AD-related systems, it is not sufficient to construct test scenarios based only on the specifications of functions defined by the Function Definition Specification (FDS) [75]. Even if the expected results are obtained, including the passing and failure of test scenarios, there may still be unforeseen changes in the internal logic of systems that are not directly related to the specifications of the SUT (e.g., the routing results when emergency braking is triggered). Traditional testing methods, such as unit testing, generate independent test cases that lack interrelatedness [1]. Consequently, when assessing the adequacy of internal signals, the oracle problem arises, but can be alleviated through the usage of MT.

MRs are often designed at a high level of abstraction and may include not just individual relations, but also combinations of several relations [255]. This complexity has led to the introduction of MRPs to provide a generalized framework for these groups of MRs [255]. MRPs have subsequently given rise to two subclasses of patterns: MRIPs [45] and MROPs [176]. This section introduces three MRPs designed for evaluating the functionality of AD simulators [249]: $MRP_{ImpactAmplification}$, $MRP_{MarginalEnsembleRevision}$, and $MRP_{ScenarioCombinations}$. For each of these MRPs, a corresponding MRIP is specified. The purpose of these MRIPs is to assist practitioners in efficiently generating and implementing test scenarios and MRs that are applicable to their testing situations.

Since the simulator is applied closely to the ADS testing, it is essential to determine whether the issue originates from the simulator or the ADSs. However, this also means that the MRs derived by the MRPs and MRIPs can be used both for testing the ADSs and simulators. Note that there is a subset relationship between the MRs used for testing ADS functionalities and those used for simulator evaluation: While assessing simulator performance sometimes requires certain MRs (e.g., the spawning of actors within the simulation), a majority of the MRs applicable in ADS testing should also be able to reveal potential defects within the simulator.

The following MRPs and MRIPs were proposed and used to generate MRs for testing the NIO self-developed AD simulator with the SAPA function[2] [249]. The results indicate that they exhibit impressive efficacy in uncovering discrepancies within the simulator (the experiments can be found in Section 5.2.3). The potential reasons for this level of performance are detailed in Section 5.2.4.

5.2.1.1 The "Impact-Amplification" MRP and The "Revise-Relevant-Element" MRIP

The "Impact-Amplification" MRP involves changing the elements in the scenario that have impacts on the outputs of the SUT:

> $\boldsymbol{MRP_{ImpactAmplification}}$**:** Amplifying the "influential input factors"

[2]The system automatically searches for parking slots based on the vehicle's gear position. It takes control of steering, gear shifting, and movement to park the vehicle in or out of the detected slot when a suitable one is found and the driver confirms their intention [210].

> that have a determinative impact on the system's output aligns or intensifies the system's output, whereas altering the "non-influential factors" leaves it unaffected.

The phrase "influential input factors" refers to distinct elements within a given scenario that have a determinative impact on the successful execution of the SUT's functions. These elements may include, but are not limited to, the spatial arrangement of entities within the scenario, or particular environmental configurations. When these "influential factors" are increased or "amplified", they cause a corresponding change in the system's output. This change could be in the form of intensifying the output or aligning it more closely with the original outcome. On the other hand, there are input factors considered "non-influential": These inputs, when altered or modified, do not significantly affect the system's output. In other words, changes in "non-influential factors" do not lead to noticeable changes in how the system operates or in the results it produces. The types of outputs generated by the system can range from quantifiable metrics, such as signal readings or patterns of variation, to less quantifiable constructs, such as observable behaviours of vehicles. It is important to note that the input factors defined in the MRP are determined based on human knowledge and understanding of the systems, rather than being directly derived from the ML models, given that the inherent randomness of these models makes output control impractical. This consideration also applies to the later introduction of MRs and other MRPs/MRIPs in this study.

As an example, consider an MR where the source scenario features a driverless car (the ego vehicle) that is unable to park in a designated slot due to the sudden appearance of a pedestrian obstructing its path. This unexpected event prompts the SAPA function of the vehicle to abort the parking manoeuvre to ensure pedestrian safety. In this instance, the pedestrian is identified as the "influential input factor", directly influencing the ego vehicle's decision to abort its parking function. Record the position when the SAPA function has aborted. The follow-up scenario is constructed where an additional obstacle is placed closely behind the pedestrian, which does not affect the original impact of the pedestrian on the ego's final decision. This modification serves to amplify the "influential input factor" in the source scenario—the increasing the obstacles causing the SAPA function to abort. Despite the introduction of this additional element, the expected outcome of the scenario should remain consistent with the source scenario: The ego vehicle, adhering to its safety protocols, should still abort the parking manoeuvre in a position that is the same or close to the original one in the source scenario.

Note that this MRP has potential applicability beyond ADs. For example, in the field of search engine quality testing [256], one MR could be defined to integrate the top search result in the source test case into the query terms of the follow-up test cases. The underlying expectation is that this enhanced query, now more directly targeting the anticipated top site, should yield the same leading result.

However, $\text{MRP}_{\text{ImpactAmplification}}$ lacks comprehensive specifications for transforming the source scenario into subsequent follow-up scenarios. To help users to develop concrete MRs, $\text{MRIP}_{\text{ReviseRelevantElement}}$ was introduced. This input pattern focuses on the modification of the "relevant" entities and environments within the source scenarios that are marked as the "influential and non-influential factors" in $\text{MRP}_{\text{ImpactAmplification}}$, providing a structured way to change scenarios.

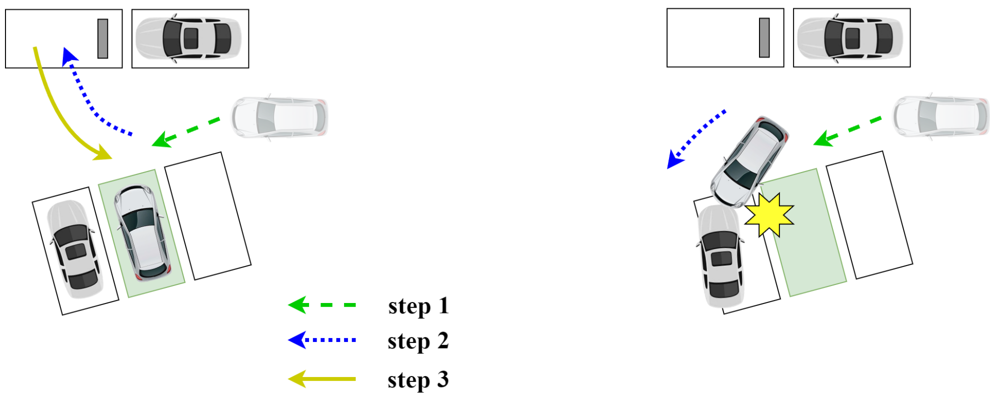
> ***MRIP$_{\textit{ReviseRelevantElement}}$*:** $\text{MRIP}_{\text{ReviseRelevantElement}}$ refers to construct follow-up inputs by adding or removing "relevant" elements, altering the behaviours of existing "relevant" elements, or modifying "relevant" environmental settings in the test scenario.

The "relevant" elements refer to those in the scenario that have a potential impact on the output of the systems. It involves adding, removing, and repositioning the vehicles (including the ego vehicle), pedestrians, and other interactive objects from the source input to construct follow-up inputs. Altering the behaviours of existing "relevant" elements involves changing how key elements within the scenario, like vehicles and pedestrians, behave. Influential behaviours, such as a change in the vehicle's speed or a pedestrian's movement, can directly affect the system's response. For example, a faster-approaching vehicle (an "influential factor" in the scenario) may prompt the system to take more decisive action. In contrast, non-influential behaviours, such as changing the colour of a vehicle, typically do not alter the system's output, serving to validate the system's ability to ignore "irrelevant" variables. Modifying environmental settings includes changing the driving environment, like road conditions or weather. Influential environmental changes, such as icy roads or heavy fog, can significantly impact the system's decision-making, requiring it to adapt appropriately. Non-influential environmental changes, like a slight variation in ambient temperature, should typically not affect the system's performance. These adjustments to the entities and environments within the simulation can generate a broad range of scenarios to test the SUTs that increase the possibility of revealing system defects.

Example MRs The following contents present two MRs that were generated by using $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ and $\mathrm{MRIP}_{\mathrm{ReviseRelevantElement}}$. One MR is generated by amplifying the "influential factors" of the passed scenarios:

> ***$MR_{FollowSuccess}$:*** Suppose that in a source scenario (S_0), the ego vehicle successfully executes a function. In the follow-up scenarios (S'_x, where x is the index of the follow-up scenarios), we modify S_0 in such a way that the modification does not interfere with the manoeuvres of the ego vehicle, and then the ego vehicle should still be able to execute the same function.

Consider a source scenario S_0, where the ego vehicle successfully parks in a designated space. According to $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ and $\mathrm{MRIP}_{\mathrm{ReviseRelevantElement}}$, if



(a) Source scenario

(b) Follow-up scenario (MR failed)

Figure 5.1: $MR_{FollowSuccess}$ example

an obstacle is removed from scenario S_0, the ego vehicle should retain its ability to park in the same space. By ensuring the modification does not interfere with the manoeuvres of the ego vehicle, the output relation can be generated: The ego vehicle should be able to execute the functions in both the source and follow-up scenarios. Figure 5.1 illustrates a detected violation of $\mathrm{MR}_{\mathrm{FollowSuccess}}$. In Figure 5.1(a), the source scenario involves the ego vehicle successfully parking in the middle parking space, marked in green. The arrows indicate the vehicle's movement path. For the follow-up scenario, shown in Figure 5.1(b), the obstacle vehicle was repositioned (initially on the below-left parking space) further from the ego vehicle's path. This alteration unexpectedly resulted in the ego vehicle not only failing to park but also colliding with the obstacle car, thereby violating $\mathrm{MR}_{\mathrm{FollowSuccess}}$.

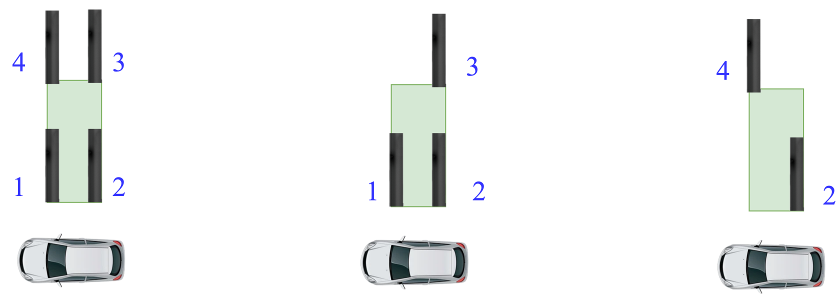
Building on $\mathrm{MR}_{\mathrm{FollowSuccess}}$, and in alignment with the MRP and MRIP, a complementary MR was proposed, $\mathrm{MR}_{\mathrm{FollowFailure}}$. This relation creates the opposite scenarios to $\mathrm{MR}_{\mathrm{FollowSuccess}}$, focusing on amplifying the situations where the system failed.

> $\boldsymbol{MR_{FollowFailure}}$**:** Suppose that in a source scenario (S_0), the ego vehicle fails to execute a function. In the follow-up scenarios (S'_x, where x is the index of the follow-up scenarios), we modify S_0 in such a way

that the modification does not interfere with the manoeuvres of the ego vehicle, and then the ego vehicle should still fail to execute the same function.

A detected violation of $MR_{FollowFailure}$ was shown in Figure 5.2. The source scenario showed the ego vehicle attempting to park in a space with poles at the four corners (Figure 5.2(a)), which resulted in the SAPA function aborting. The space between the poles (i.e., poles 1 & 2 in the image) positioned on the same side of the parking slot, was narrower than the ego vehicle's width, thereby preventing successful parking. Since the poles prevented the successful execution of the SAPA function, they were the "influential factors" of these failure scenarios in this case. The Abort Reason Code for why the function aborts was 47 (meaning the vehicle was blocked during parking). In follow-up scenario 1 (Figure 5.2(b)), removing one of the poles further away, the vehicle behaved the same as the source scenario and therefore, did not violate the MR. However, in follow-up scenario 2 (Figure 5.2(c)), removing the diagonal poles altered the SAPA Abort Reason Code from 47 to 37, and the system reported a planning failure.

The change in the Abort Reason Code from 47 to 37 between the source scenario and follow-up scenario 2 was a significant observation that revealed a bug in the SAPA function and the simulator. In the source scenario, an Abort Reason Code of 47 was observed, indicating that the SAPA function recognized the vehicle as being blocked, and aborted the parking attempt—aligning with the expectation that the ego vehicle cannot park due to the presence of poles and the size of the slot. In follow-up scenario 1, when one of the poles further away was removed, the SAPA function maintained the same Abort Reason Code of 47, indicating that it correctly recognized the parking situation. However, in follow-up scenario 2, when the diagonal poles were removed and the slot remained the same, the SAPA function changed its response to an Abort Reason Code of 37, signifying a planning failure of the function. Since removing one of the obstacles should not



(a) Source scenario (b) Follow-up scenario 1 (c) Follow-up scenario 2

Figure 5.2: $MR_{FollowFailure}$ example

cause the function to fail to plan the route (i.e., removing noise in the scenario should not affect the ability of the system), this inconsistency in function outputs among the three scenarios suggests that the function's decision-making process may not be correctly evaluating the parking situation. In addition, this change caused the ego vehicle to frequently shift gears and ultimately led to the function's failure (instead of aborting), which was not found in other scenarios, violating the MR. This issue was further confirmed as the problem with the pole in the simulator and has been fixed in later updates.

5.2.1.2 The "Marginal-Ensemble-Revision" MRP and The "Entity-Property-Reconfigurations" MRIP

The second MRP, $\text{MRP}_{\text{MarginalEnsembleRevision}}$, involves creating variations of the entities such that the scenario outputs should remain the same.

> ***$MRP_{MarginalEnsembleRevision}$*:** Develop variations with negligible impact on the entities that do not significantly alter the scenario's outcome, and the results should remain the same.

$\text{MRP}_{\text{MarginalEnsembleRevision}}$ includes alterations or adjustments to entities that have minimal impact on the scenario outputs. The word "negligible" implies that the changes are so small that they are almost non-existent or unimportant in terms of affecting the outcome. Examples include: rotating an entity within a small range; adjusting the initial position within a limited range; or switching to a similar entity type. These variations can be regarded as minor refinements that should not fundamentally change the outcome or have a noteworthy impact on the driving scenario.

The main difference between $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ and $\mathrm{MRP}_{\mathrm{MarginalEnsembleRevision}}$ lies in their approach to constructing follow-up scenarios. $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ focuses on adjusting the influence level of specific input factors: It amplifies the “influential factors” to observe the system’s intensified or aligned response and alters the “non-influential factors” to ensure that they do not affect the system’s output. In contrast, $\mathrm{MRP}_{\mathrm{MarginalEnsembleRevision}}$ emphasizes varying the attributes or types of entities in a negligible way within the scenario, while it does not need to specifically identify those factors. The changes made are not significant enough to change the SUT’s outputs.

In line with the approach described in the previous sections, an MRIP, which was made to show how input scenarios can be transformed based on the MRP, was presented.

> ***MRIP$_{EntityPropertyReconfigurations}$*:** The “Entity-Property-Reconfigurations” MRIP is defined as the process of constructing follow-up inputs by altering specific properties of the entities present in the source scenarios.

The entities in this MRIP can be either “influential” or “non-influential”. “Altering specific properties” means making changes to the characteristics or attributes of these entities: For instance, this could involve changing a vehicle’s size, colour, or speed; or a pedestrian’s walking pattern on the sidewalks (walking or running).

Example MRs Based on the proposed MRP and MRIP, two following MRs were introduced. These MRs changed certain properties of the target entity in ways that should not change the system outputs.

> ***$MR_{OrientationInvariant}$:*** Suppose that in the source scenario (S_0), the ego vehicle successfully executes the function. Then, in the follow-up scenarios (S'_x, where x is the scenario index of the follow-up scenarios), by changing the orientation of the components in the environment while ensuring the relative position between the ego vehicle and surrounding components necessary for the successful execution of the functions remains unchanged, the ego vehicle should still be able to execute the function.

$\mathrm{MR}_{\mathrm{OrientationInvariant}}$ refers to a collection of scenarios where a specific group of entities undergo rotations, resulting in the relative positions among them remaining unchanged. Figure 5.3 shows an example of the test cases of $\mathrm{MR}_{\mathrm{OrientationInvariant}}$. In the source scenario, the ego vehicle is able to successfully park in a parking slot that is perpendicular to the road. Rotating both entities from 0 to 90 degrees, the vehicle should still park in the slot. The scenarios from left to right (Figure 5.3(a) to (c)) show the scene after rotations of 0, 35, and 76 degrees as an example, with the road and other components in the scenarios remaining unchanged. The ego vehicle behaved consistently in these scenarios, and the MR was not violated.

Another implementation of the MRIP and MRP involves adjusting the initial position of the ego vehicle:

> ***$MR_{InitPositionTolerance}$:*** Suppose that in the source scenario (S_0), the ego vehicle successfully executes the function. Then, in the follow-up scenarios (S'_x, where x is the scenario index of the follow-up scenarios), by slightly adjusting the ego vehicle's initial position in such a way

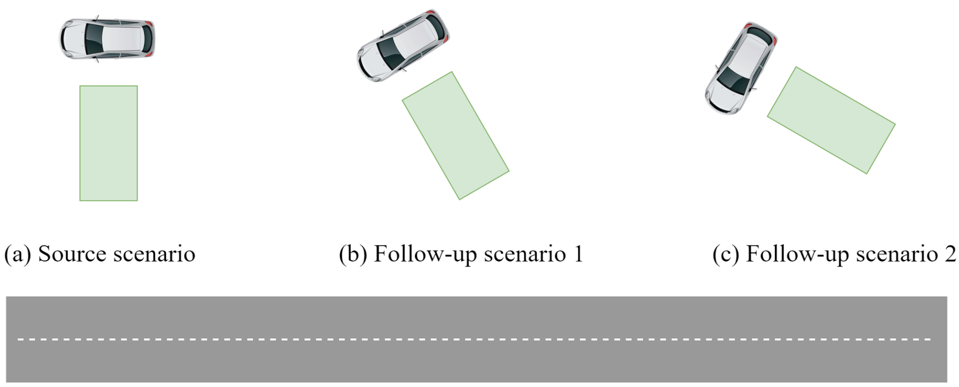


Figure 5.3: $MR_{OrientationInvariant}$ example

that the modification should not interfere with the manoeuvres of the ego vehicle, the ego vehicle should still be able to execute the function.

Figure 5.4 shows an example violation of this MR. The source scenario showed the initial position of the ego vehicle relative to the slot's centre as ($\delta x = 0m$, $\delta y = 5m$; where x and y refer to the horizontal and vertical coordinates on the map), allowing the ego vehicle to park successfully. Adjusting the initial position of the ego vehicle by 1 meter either horizontally or vertically should not affect the vehicle's ability to park successfully, as no obstacles were blocking the path of the ego vehicle and the ego vehicle did not encroach on any slot. However, in follow-up scenario 1 (Figure 5.4(b)), the ego vehicle was positioned closer to the slot ($\delta x = 1m$, $\delta y = 4m$), resulting in the SAPA function's abortion due to planning failure. Nonetheless, in follow-up scenario 2 (Figure 5.4(c)), the ego vehicle's initial position was fine-tuned to a little further ($\delta x = 1m$, $\delta y = 6m$), facilitating a successful parking outcome. The ego vehicle's inconsistent behaviours among the scenarios constituted a violation of the MR (the distance between the ego vehicle and the slot did not violate the function's specifications).

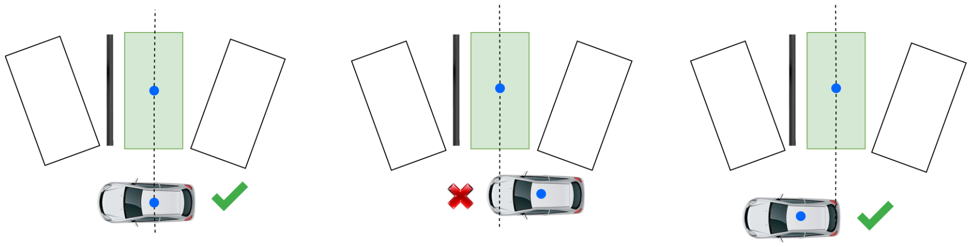


Figure 5.4: $MR_{InitPositionTolerance}$ example

5.2.1.3 The "Scenario-Combinations" MRP and The "Entity-Integrations" MRIP

To extend the scope of scenario application and increase the variety of inputs, the third MRP, $MRP_{ScenarioCombinations}$, was introduced. This pattern constructs MRs by connecting a series of source scenarios.

> $\boldsymbol{MRP_{ScenarioCombinations}}$**:** If multiple source scenarios are independent of each other and the system outputs consistent results, then by combining the source scenarios to generate the follow-up scenarios, the system output should remain unchanged.

In this context, "consistent results" does not strictly refer to identical sensor data. Rather, it includes the ego vehicle's behaviour, similar patterns in SUT outputs, or other criteria that can be used to combine different scenarios. "Independent scenarios" means that entities that lead to a specific system output in each scenario are separate from each other. For example, if moving obstacles such as a cat or a pedestrian are placed on the ego vehicle's path, causing the SAPA function to abort with the same abort reason, then they can be incorporated into the follow-up scenarios (as long as they do not interact with each other). Figure 5.5 shows a cat and a pedestrian placed together in the scenario, moving towards each other on the ego vehicle's path.

To help users to develop concrete MRs in merging scenarios, the following MRIP

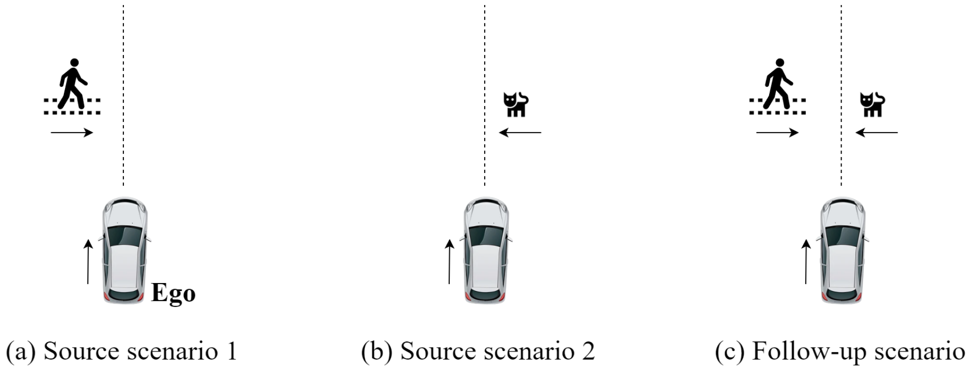


Figure 5.5: Independent scenario example

was proposed:

> ***MRIP**$_{EntityIntegrations}$*: The "Entity-Integrations" MRIP is defined as the process of constructing follow-up inputs by integrating entities that are independent of each other in a series of scenarios.

The integration of independent entities in various scenarios involves creating a new scenario where each entity retains its original properties and behaviours from the initial scenarios, unaffected by the presence of other entities in the new scenario. For example, as illustrated in Figure 5.5, the follow-up scenario is designed by placing a pedestrian and a cat in their respective positions from source scenarios 1 and 2, without influencing each other. As a result, the follow-up scenario output should be the same as the source scenarios 1 and 2.

Example MRs The following MR was constructed based on the proposed MRP and MRIP:

> ***MR**$_{ScenarioIntegration}$*: Suppose that in a series of source scenarios (S_x, where x is the scenario index of the source scenarios), the ego vehicle successfully executes the function with the same output. Ensuring the entities in the source scenarios are different and do not interfere with each other if they are put together. Then, by randomly combining

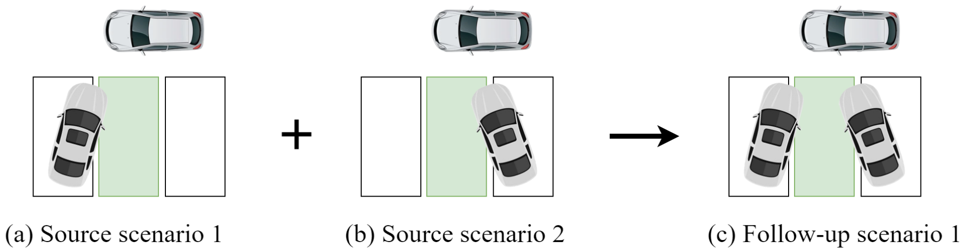
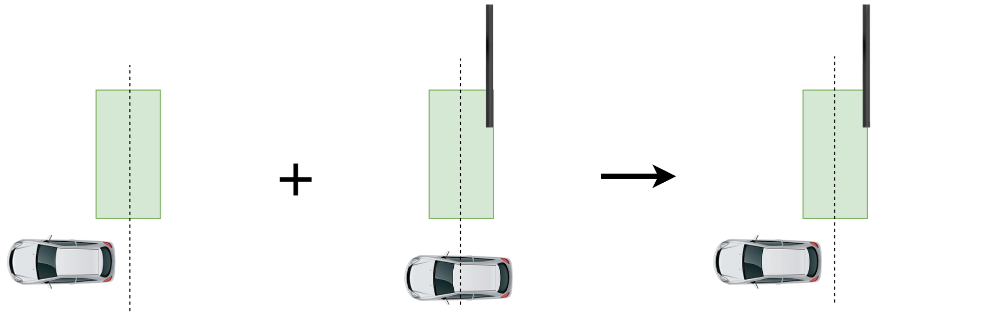


Figure 5.6: Example violation of $\mathrm{MR}_{\mathrm{ScenarioIntegration}}$

source scenarios into follow-up scenarios (S'_y, where y is the scenario index of the follow-up scenarios) in such a way that the modification does not interfere with the manoeuvres of the ego vehicle, the ego vehicle should still be able to execute the function.

A detected example of $\mathrm{MR}_{\mathrm{ScenarioIntegration}}$ violation was presented in Figure 5.6. In the source scenario, the ego vehicle was unable to park in the target slot due to the side vehicle in the left slot encroaching upon the target slot (Figure 5.6(a)). The SAPA Abort Reason in this scenario was 47, indicating that the ego vehicle was blocked. In source scenario 2 (Figure 5.6(b)), the ego vehicle behaved similarly (i.e., failed to park), and the SAPA Abort Reason remained the same. Intuitively, a new (follow-up) scenario created by combining the two source scenarios should not result in a different SAPA Abort Reason since the ego vehicle was still blocked. However, in the follow-up scenario (Figure 5.6(c)), the SAPA function became unresponsive instead of aborting, and the ego vehicle behaved differently to the source scenarios; therefore, the MR was violated because the SAPA Abort Reason was different.

Figure 5.7 shows another $\mathrm{MR}_{\mathrm{ScenarioIntegration}}$ violation example. In the first source scenario, the ego vehicle was positioned at ($\delta x = 1m$, $\delta y = 4m$) relative to the centre of a parking slot (Figure 5.7(a)). Here, the vehicle failed to park successfully, and the SAPA function aborted. In the second source scenario, the vehicle was initially placed in a location that could successfully park. However, a pole was positioned inside the slot, near the corner, preventing successful parking



(a) Source scenario 1 (b) Source scenario 2 (c) Follow-up scenario 1

Figure 5.7: Another example violation of $\mathrm{MR}_{\mathrm{ScenarioIntegration}}$

(Figure 5.7(b)). In this scenario, too, the SAPA function aborted with the same abort reason code as in the first source scenario.

These two source scenarios were then merged into a unified follow-up scenario in Figure 5.7(c), where the ego vehicle started from the initial position of the first source scenario, but with the parking slot containing the pole from the second scenario. According to the MR, the ego vehicle should not be able to park, and the function should abort, as had happened in the source scenarios. Contrary to expectations, the ego vehicle attempted to park but collided with the pole before the function aborted, thereby violating the MR.

5.2.1.4 Methodology for Generating MRs for Common ADS Modules

In order to simplify the usage of the provided MRPs/MRIPs, the author proposed a methodology to help generate effective MRs in a time-efficient manner. The first step involves identifying the target module and obtaining a fundamental understanding of its input/output and functionality. Common ADSs, such as those outlined in the literature [146][227][14], comprise multiple modules, including planning, perception, routing, control, and localization [19]. These modules have their own specific functions, yet they are interconnected [147]. For instance, the planning module requires information from both the vehicle (data from chassis, localization, and relative map modules) and the surrounding environment (data from perception and routing modules) to generate the trajectory for the control module [19]. Once the target module has been selected, the user can create a source scenario and choose the MRPs/MRIPs to construct MRs. The scenarios can be created either by existing guidelines like ASAM OpenSCENARIO [13], or following a template described in the next chapter of the thesis (Section 6.2.1). An example of using an MRP and MRIP to generate an MR for the planning module of an ADS is described below.

Suppose the user decides to test the AEB (Autonomous Emergency Braking) function [230] and creates a source scenario, which involves a single-lane road. The ego vehicle is initialized at the starting point with constant speed and is expected to drive along the road. A pedestrian is scheduled to cross the road horizontally, intersecting with the ego vehicle's path. To generate the MR, the user can select $\text{MRP}_{\text{ImpactAmplification}}$ and $\text{MRIP}_{\text{ReviseRelevantElement}}$ to create follow-up scenarios. One possible method is to make the pedestrian start earlier but ensure the distance to the ego vehicle is close enough to cause the ego vehicle to activate the AEB function. The sooner the pedestrian starts, the earlier the AEB is activated. Then if the ego vehicle successfully avoids collision in the source scenario, it should also avoid collision in the follow-up scenarios.

The final MR could be stated as follows: In the source scenario, the ego vehicle drives along the single-lane road and stops to avoid a pedestrian that suddenly crosses the road, with the AEB function activated. In follow-up scenarios, by making the pedestrian start earlier, while keeping other conditions unchanged, the ego vehicle should activate the AEB function sooner, and reach a final stop to avoid collision.

An issue was observed with the system where, despite the AEB being activated, the vehicle failed to stop in time to avoid a pedestrian, yet the final output showed no collision. After verification, it was confirmed that this was not an issue with the ADS but rather an issue with the output report from the simulator, which highlights the effectiveness of the MRPs/MRIPs in identifying flaws within the simulator.

5.2.2 A Scenario-Driven MT Framework

Alongside the MRPs and MRIPs, the author proposed a scenario-driven MT framework that integrates ME and MT to enhance the issue identification and reporting process. Unlike the ADS-based test harness discussed in Section 3.3.2, which focuses on the practical test execution stage, the scenario-driven MT framework encompasses the entire MT process, from generating MRs to identifying MR violations. Both frameworks have played significant roles in advancing MT testing and bug identification (Sections 3.3.3 and 5.2.2.3).

5.2.2.1 Framework Introduction

ME, as a variation of MT [255], aids testers in familiarizing themselves with the SUT by generating HMRs. A violation of an HMR signifies a potential gap in the tester's comprehension of the system, rather than necessarily a software fault. In some instances, HMR violations may prompt testers to enhance their knowledge of the system and improve their skills in testing [255]. Within the scenario-driven MT framework, ME plays an important role for helping testers quickly and smoothly understand SUTs. To derive HMRs, testers are recommended to employ two strategies:

1. Acquire knowledge of the system via available information sources: This strategy involves gathering knowledge about the SUT through available information sources, such as system specifications, documentation, and insights from other testers. This knowledge can help to identify potential HMRs that can be applied to test the system more effectively.

2. Utilize observed system behaviour and scenarios generated through scenario-driven MT: This strategy involves closely observing the SUT and scrutinizing earlier testing outcomes to identify the kinds of inputs that are more likely to reveal defects. By identifying such inputs, testers can generate suitable testing scenarios and use them in the scenario-driven MT procedure to generate relevant HMRs. This approach can help in identifying and testing critical functionalities of the software system while improving the efficiency and effectiveness of the overall testing process.

Initiating the ME process at the outset of testing a new function can accelerate testers' understanding of the system [140]. This is especially beneficial when testers no longer need to comprehend lengthy FDS documents, which may require a significant amount of time [166]. This process helps with the preparation for MT, and enhances the tester's proficiency in performing MT. Furthermore, additional iterations of the ME process can increase the probability of turning HMRs into MRs and generating scenarios that are more likely to reveal system problems. The HMRs are verified either by FDS or relevant stakeholders before they become MRs, and ME transforms into the MT process.

The scenario-driven MT framework integrates ME and MT processes to assist in the testing of ADSs and AD simulators. Figure 5.8 shows an overview of the framework. Unlike other MT approaches that produce MRs first [175], the scenario-driven MT framework develops the source scenario first, then the (H)MRs are generated, based on the assumption that certain components of the system are more susceptible to bugs. The (H)MRs are used to generate follow-up test scenarios. These generated scenarios are input into the SUT, and the results obtained are used to enhance the quality of (H)MRs and scenarios until MR violations are detected. Once identifying such violations, the results are communicated to the relevant developers based on the nature of the issues. If developers acknowledge it as a problem with the HMRs, testers can then enhance

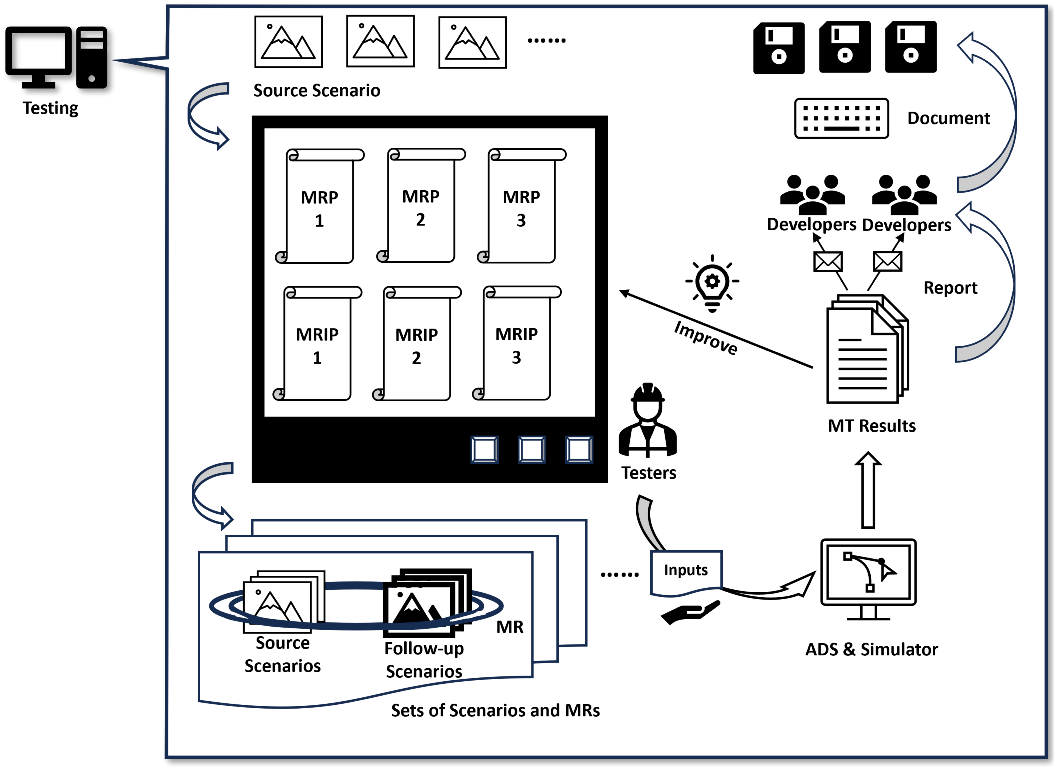


Figure 5.8: Scenario-driven MT framework overview

the HMRs and get ready for the next testing phase. Conversely, if it is determined to be a defect within the system, it will enter the subsequent bug-fixing process. After the tests are completed, the entire procedure, along with the (H)MRs, is documented and archived.

One limitation of the framework is that testers still need to manually create scenarios and generate test reports. The framework combines ME and MT, making it possible for testers to spend less time reviewing SUT specifications before starting testing (although this step remains necessary). This approach helps testers to use MT more efficiently and confidently, as reported in Section 5.2.2.4.

5.2.2.2 Framework Stages and Procedures

Figure 5.9 presents the detailed procedures of the scenario-driven MT framework. The complete framework has been grouped into three stages. The first stage (*STAGE 1 (ME)*) is ME preparation. This is where testers create the scenario as the source test case and then use it with the MRPs and MRIP to produce HMRs. Alternatively, they can use what they already know from their own experiences or the experiences of others, like coworkers. They then generate follow-up test cases (scenarios) based on the HMRs.

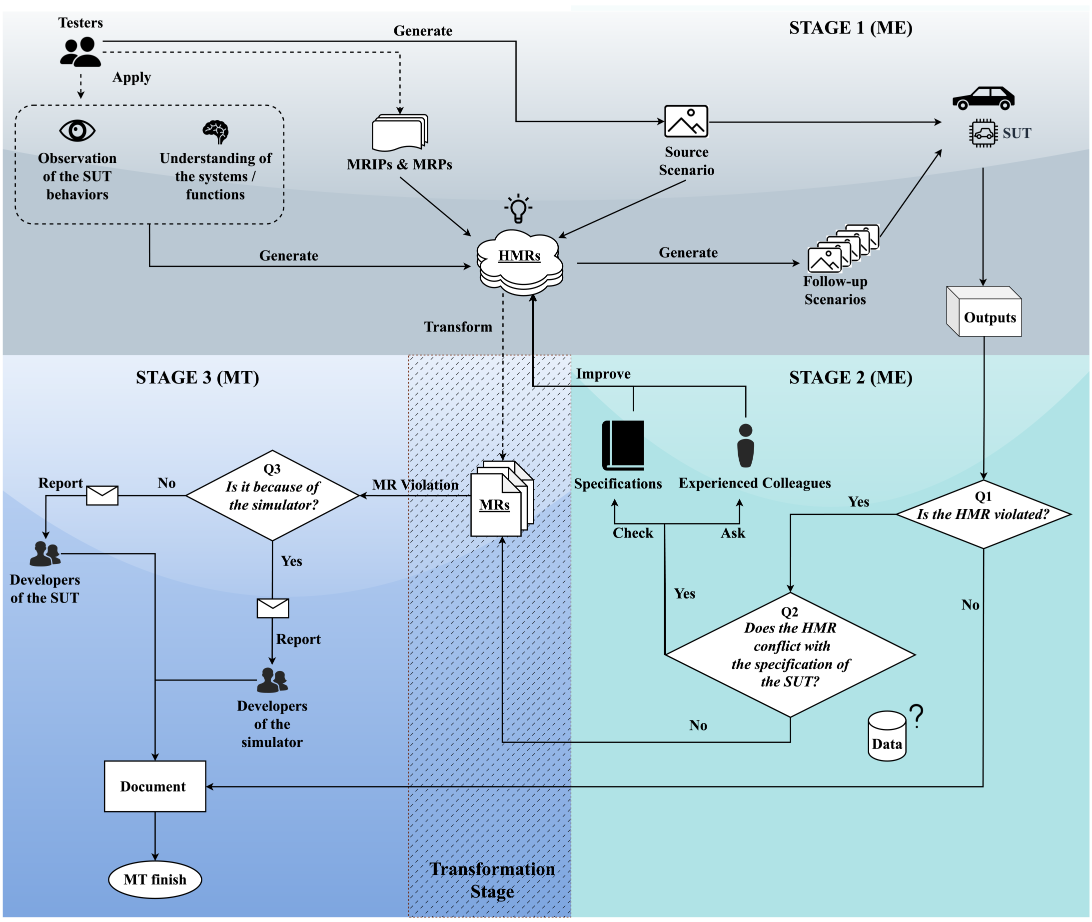


Figure 5.9: Scenario-driven MT framework details

In the second stage (*STAGE 2 (ME)*), the testers gather the outputs and evaluate them against the HMRs. When the HMRs are not violated (*Q1*), testers only need to document the results and proceed to the next testing round. In cases where HMR violations occur, testers are required to distinguish between problems arising from erroneous HMR definitions and those indicative of actual system problems (*Q2*). A common evaluation criterion is the determination of the “abnormal” data, which is assessed depending on both the SUT specification and on testers’ experiences. Such data can arise in situations where the system behaviour does not meet specifications or falls outside the testers’ previously encountered scenarios. Even when data appears to be normal, testers should stay alert by carefully reviewing the system requirements or consulting with knowledgeable colleagues for insights. Identification of any disparities between the HMR and system specifications can facilitate the refinement of the HMR and enhance testers' comprehension of the system. This ME process is iterated until it is verified that the MR violations are not related to the definition of the HMRs. In this case, the HMRs become MRs, and the ME process has transformed into the MT process (Transformation Stage in Figure 5.9).

The third stage (*STAGE 3 (MT)*) starts once the HMR has become an MR. Testers must now determine whether or not the simulator or SUT is at fault (*Q3*). As the ADS and sensor signals are simulated, the MR violations may be attributable to the simulator, necessitating the validation and verification of the simulator software. In such cases, testers should confirm the issue by checking whether or not the simulator operates according to its intended functionality. In both cases, testers should communicate with the relevant developers—either from the simulation or ADS teams—to clarify the nature of the issues. These unconfirmed issues should be classified as potential system flaws until they undergo thorough reproduction or are dismissed by the developers of the SUT. Finally, a proper documentation is recommended for revision and reference in the future.

5.2.2.3 Impact of the Framework on MT Performance

This section explores the impact of the scenario-driven MT framework on MT performance when used to test the NIO ADS and AD simulator. Table 5.1 provides a quantitative analysis of the efficacy of transforming ME into the MT process.

Number of (H)MRs Constructed This metric represents the total number of (H)MRs constructed through the scenario-driven MT framework (including those later transformed into MRs). The value of 36 represents an obvious in-

Table 5.1: Framework impact on HMR generation and transformation

Number of (H)MRs Constructed	Number of HMRs Confirmed	Number of HMRs Transformed	HMR-to-MRs Transformation Rate
36	5	31	86%

crease in the number of relations generated compared to the previous experiments with ADSs [246, 250] presented in Chapter 3. This highlights the framework's effectiveness in motivating users to generate (H)MRs.

Number of (H)MRs Confirmed Among all the (H)MRs constructed (36), five were confirmed as HMRs. These relations essentially help testers enhance their understanding of the SUT.

Number of HMRs Transformed A total of 31 HMRs were transformed into MRs, resulting in the bugs discussed in Section 5.2.3. This number indicates that the framework is effective in both MR generation and bug discovery.

HMR-to-MRs Transformation Rate The rate of transformation from HMRs to MRs is a key metric in evaluating the efficiency of the scenario-driven MT framework. This rate is calculated using the formula:

$$\text{HMR-to-MRs Transformation Rate} = \frac{\text{Number of MRs}}{\text{Total Number of (H)MRs}}$$

The value of 86% in Table 5.1 suggests a high rate of successful transition from HMRs to MRs. This reflects the framework's effectiveness in improving both MT efficiency and the testers' ability to carry out testing.

5.2.2.4 Impact of the Framework on Testing Process

This section outlines the impacts that the framework has had on the tester's perspective after interviewing several testers from NIO. By implementing the scenario-driven MT framework, the testing process has been standardized, leading to faster identification of defects and more efficient issue reporting. Additionally, it has enhanced testers' confidence and ability to use ME and MT effectively for system exploration and testing. The detailed improvements include:

1. **Enhanced issue reporting process:** Prior to the deployment of the scenario-driven MT framework, the testers faced considerable uncertainty regarding where to initiate the MT procedures. Additionally, the confusion extended to handling MR violations, as they were uncertain whether to report problems to the simulator developers or the ADS function developers. This uncertainty frequently led to delays and inefficiencies in the testing process.

   With the implementation of the framework, these uncertainties have been effectively resolved by establishing a clear and well-defined pathway for both testing and issue reporting. This approach has enabled quicker identification and reporting of defects, significantly reducing overheads and enhancing the overall efficiency of the testing process. As reported by one of the main users of this framework, the adoption of this framework led to a marked improvement in bug-finding efficiency, primarily due to the framework's role in streamlining the process of addressing MR violations. Notably, the bug discovery frequency—based on the estimated average time between each bug found—revealed that before the introduction of the framework, bugs were discovered at an average interval of six days. Following the framework's implementation, this interval decreased significantly, with bugs being discovered every two days on average. The framework facilitated quicker transitions between ME and MT, which enhanced testers' ability to promptly identify, report, and confirm issues.

2. **Improved decision-making process:** The framework's structure naturally guides the progression of testing steps. This progression begins with testers developing the source scenario and culminates in a collaborative effort between testers and developers to identify and resolve MR violations. This guided approach simplifies decision-making for testers, allowing them to focus more on the intricacies of each test case than the overarching process.

3. **Boosted in tester confidence:** Before the framework's implementation, the testers lacked familiarity and confidence with MT. The framework has also served as an educational tool in this context. As testers became more adept at employing the scenario-driven MT framework, their confidence in using MT techniques has notably increased, as reported by their oral feedback. This increase in confidence is not merely psychological: It is reflected in the enhanced efficiency of testing and has led to a quicker identification of system defects, as previously mentioned.

4. **Simplified testing process:** Unlike traditional methods where testers may spend days studying the FDS in detail, familiarity with the FDS is helpful but not mandatory for starting tests when applying this framework. It reduces the risk of forgetting system details during testing, which often leads to time-consuming revisits of the FDS. Here, the FDS acts as a reference guide, consulted by testers only as needed. Engineers focus mainly on the input-output relationships to construct MRs, where the FDS is used for necessary validation and verification, streamlining the testing process.

5.2.2.5 The Weaknesses of Scenario-Driven MT Framework

To maximize the framework's potential, two weaknesses must be addressed: To effectively identify the sources of MR violations, testers must work closely with developers. Feedback from developers plays a crucial role in the overall testing and learning process. Therefore, testers and developers of the SUT should establish and maintain an effective communication channel. Meanwhile, the determination of whether a given issue stems from the simulator or the self-driving system necessitates additional effort and time, which would result in a reduction in testing efficiency. If testing the simulator in isolation, any MR violations can be attributed directly to the simulators. This eliminates the need for a time-consuming assessment to determine whether the issue lies with the simulation team or the AD function development team. This approach would enhance testing efficiency and conserve time, representing a potential avenue for future research and optimization.

Secondly, the time span between initiating the test (ME) and identifying issues (MT) can impact key performance indicators (KPIs) in industrial practice, which are specific metrics used to evaluate the success of an organization or individual in achieving their goals (e.g., the number of bugs that need to be identified within a required time). While the ME process in the framework could enhance system understanding and MR quality, it does pose time-related challenges. Therefore, it is essential for testers to be well-informed and prepared for this time commitment.

5.2.2.6 Further Refinemens

Some potential further refinements that can improve the testing efficiency and expand the practicality of the framework is as follows:

1. The scenario-driven MT framework works with traditional testing ap-

proaches, such as the Software-in-Loop (SIL) test [52]. The SIL test is the testing work that focuses on the finite state machine transitions, which refers to the changes in the state of a system or application based on specific conditions or events [114]. This testing approach is commonly used to test the SUTs. However, these current methods require the tester to be familiar with the FDS documents before starting testing. Since the scenario-driven MT framework can simplify the testing preparation process (as mentioned in the previous section), it can be used before SIL testing, which is sometimes necessary in industrial settings [156].

2. The scenario-driven MT framework helps with simulator consistency verification. The goal is to ensure that the simulator consistently produces accurate and dependable data for its users. As described in the previous section, identifying the origin of the issues detected when testing the AD simulators is one of the challenges when applying this framework. Therefore, future enhancements of the framework could involve refining the MRPs and MRIPs within it. The objective would be to enable the generation of MRs that focus exclusively on the simulator, independent of the involvement of the ADSs.

3. The scenario-driven MT framework generates more testing scenarios for other tests to use. Since test-case generation was one of MT's original applications [47], scenario-driven MT can be used as a scenario-generation methodology. With scenario-driven MT generating specific types of scenarios, it may increase the likelihood of discovering bugs when these scenarios are utilised by other testing methods.

4. Exploring the overheads of the scenario-driven MT framework by conducting future research on beginners in software testing, and comparing the time and resources consumed with not using the framework.

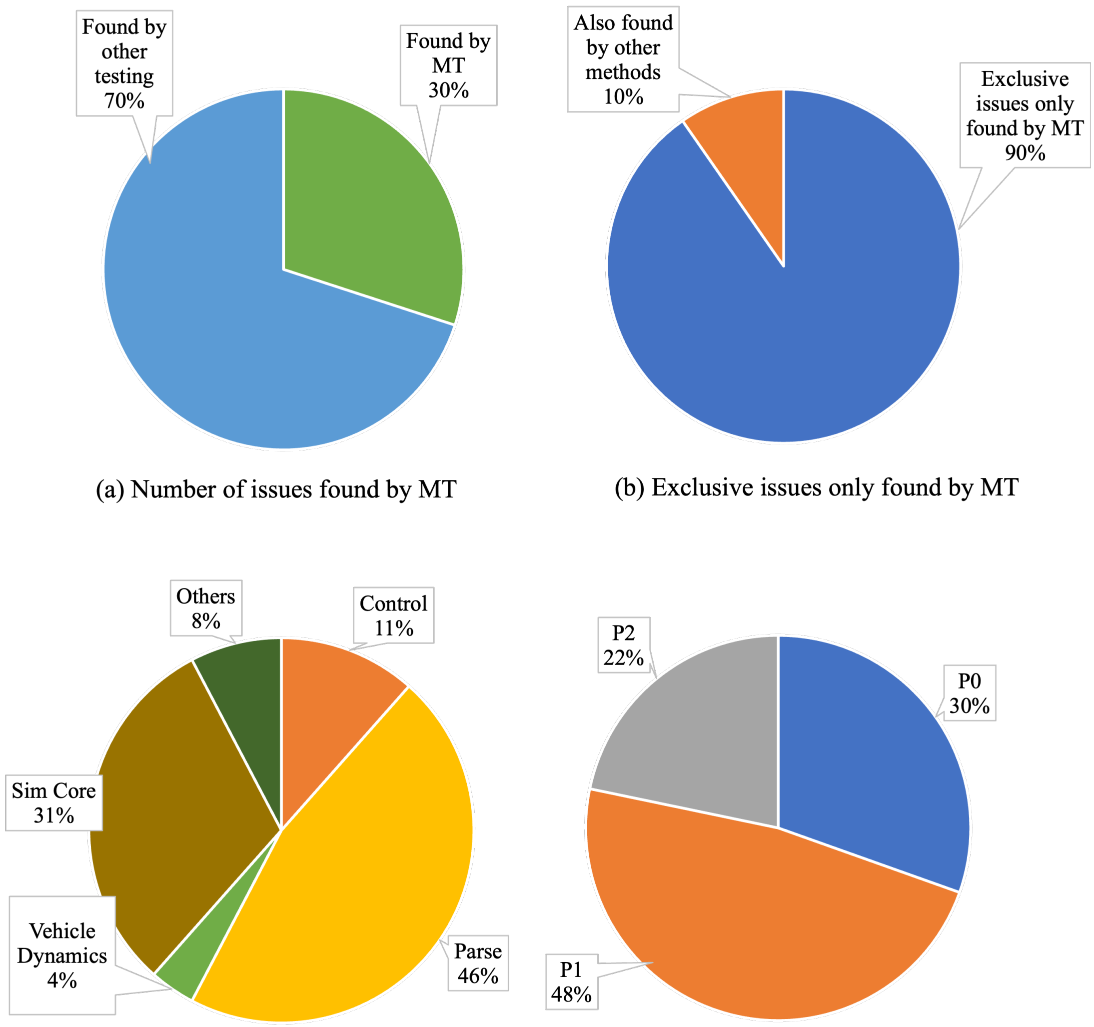
5.2.3 Discoveries and Insights

5.2.3.1 NIO AD WorldSim

This study conducted MT on the NIO self-developed simulator, NIO AD WorldSim. It is an advanced simulation tool that simulates virtual roadways and environments. This tool facilitates the establishment of closed-loop algorithmic functionality by employing virtual simulated scenarios and sensor emulation, alongside vehicle dynamics modelling. Its primary purpose is to facilitate the validation and assessment of ADS algorithms and performance. NIO AD WorldSim encompasses several crucial modules, including sensor simulation, map and scene simulation, agent simulation, and vehicle dynamics simulation. It is capable of generating synthesized data for perception model training and evaluation. It has 11 camera channels and one lidar channel, which operate on both the image and point cloud levels. Furthermore, it supports Hardware-in-the-Loop (HIL) [32] replay testing, enhancing its utility in practical applications.

5.2.3.2 Issue Overview

Figure 5.10 presents an overview of the defects discovered when the scenario-driven MT framework was used to test the parking function of the NIO ADS and AD WorldSim simulator. As indicated in Figure 5.10(a), MT identified 28 defects, constituting 30% of all the defects detected in the simulator—the remaining 70% were detected through conventional testing methods, such as unit testing. Furthermore, it is worth noting that among the identified defects, only three were also discovered through alternative testing methodologies. This finding underscores the effectiveness of the MT method, as it uniquely pinpointed a significant majority of the defects, accounting for a total of 90% of the defects identified (Figure 5.10(b)). This performance emphasizes the distinctive capabilities and advantages of the MT approach in uncovering system defects in the



(a) Number of issues found by MT

(b) Exclusive issues only found by MT

(c) Category of issues found by MT

(d) Priority of issues found by MT

Figure 5.10: Overview of defects found

tested scenarios.

The majority of the defects revealed by MT fall under the categories of *Parse* and *Sim Core*. The *Parse* category is related to the compilation of scenario elements during simulation processes. Meanwhile, *Sim Core* pertains to defects involving input signals and the underlying logic of the system. Together, they account for up to 77% of the defects (Figure 5.10(c)). Additionally, 78% of all defects identified were classified as high priority, falling under P0 and P1 (i.e., the top and second priority) categories (Figure 5.10(d)). Table 5.2 presents several noteworthy defects identified by MT across the different categories. For instance, the first issue in the table (regarding the steering angle) arose from incorrect calibration of the vehicle-dynamics parameters in a newer version of the simulator, which was identified while testing a scenario where the ego vehicle approaches a front vehicle in preparation for parking. The second issue, related to the way the poles were compiled and presented in the simulation scene, emerged when the alignment of the poles in the parking slot was adjusted. This scenario is linked to the MR examples discussed in Section 5.2.1.1. Similarly, the third issue was found when the inclination angle of the wall was modified, resulting from a misinterpretation of the wall's bounding box as an obstacle during parsing. This issue is related to the MR examples outlined in Section 5.2.1.2. Lastly, the final issue in the table arose from inconsistent results between different modules, reflecting a core algorithm problem within the simulator.

In conclusion, the results indicate that while conventional testing methods, such as unit testing [56], effectively identify a substantial portion of defects, MT stands out by examining the SUT at a more comprehensive, scenario-based level. This enables MT to focus on a broader range of system outputs, particularly those not explicitly defined in the FDS. Moreover, the study emphasizes the importance of paying attention to the specific categories of defects revealed by the MT approach, such as *Parse* and *Sim Core*, which may be more challenging to detect through

Table 5.2: Example defects found in NIO AD simulator

Issue Details	**Category**	**Priority**
The steering angle of the ego vehicle will change dynamically when approaching the front vehicle.	Control	P0
The cross-section of the poles is wrong, which will cause a core dump in the system.	Parse	P0
In this scenario, a wall, measuring 8 meters in length and 0.1 meters in width, is positioned behind the parking space. The wall is angled at 47 degrees relative to the horizontal plane. A significant discrepancy arises between the simulation and the data playback on the development side: the wall is mistakenly represented as a cube, which encroaches upon the parking space. This error triggers the vehicle's emergency braking system. Compounding the issue, the vehicle's perception signal incorrectly indicates that the parking slot is available, leading to potential operational confusion.	Parse	P0
In the simulation, a collision is detected due to the intersection of the vehicle models' bounding boxes. However, despite this reported collision, the distance signal from the parking application still indicates a small remaining distance.	Sim Core	P1

other testing methods due to the oracle problem. Traditional testing methods depend on predefined expected outcomes to assess system behaviour [178]. However, for *Parse* and *Sim Core* issues, determining the correct behaviour can be impractical or impossible. This uncertainty means that errors in parsing input data or core simulation logic may not result in obvious failures, making them difficult to detect with conventional testing.

5.2.3.3 Effectiveness of MRPs in Generating MRs to Reveal Defects

The following content compares the number of MRs generated by the MRPs with their corresponding MR failure rates, as shown in Table 5.3. The MR failure rate measures the proportion of MRs that are violated out of the total number of

Table 5.3: Comparison of the effectiveness of MRPs

MRP	**Number of MRs Generated**	**Number of Violated MRs**	**MR Failure Rate**
$\text{MRP}_{\text{ImpactAmplification}}$	17	7	41%
$\text{MRP}_{\text{MarginalEnsembleRevision}}$	11	4	36%
$\text{MRP}_{\text{ScenarioCombinations}}$	3	1	33%

generated MRs, which is calculated using the formula:

$$\text{MR Failure Rate} = \frac{\text{Number of Violated MRs}}{\text{Total Number of Generated MRs}}$$

Each violated MR resulted in a confirmed issue during testing. Compared to manually crafted MRs, which often have limitations in quantity and quality [241], both $\text{MRP}_{\text{ImpactAmplification}}$ and $\text{MRP}_{\text{MarginalEnsembleRevision}}$ can generate a substantial number of MRs (17 and 11) with relatively high MR failure rates (41% and 36%), meaning these MRs were effective in revealing defects in the SUT. The tester's preferences and the unique operational dimensions of each MRP have a major impact on the MRs and test scenarios generated, further impacting these metrics.

All three MRPs generated MRs with failure rates exceeding 30%. Although $\text{MRP}_{\text{ScenarioCombinations}}$ only generated three MRs, it achieved a failure rate of 33%. The lower number of MRs for this pattern is mainly due to its built-in limitations, including the need for more source scenarios with identical system outputs and the requirement for relative independence among the elements within those scenarios.

In summary, both $\text{MRP}_{\text{ImpactAmplification}}$ and MRP $_{\text{MarginalEnsembleRevision}}$ were able to generate numbers of MRs and achieve high failure rates. $\text{MRP}_{\text{ScenarioCombinations}}$, while operating under more restrictive conditions, also achieved a satisfactory violation rate. The results highlight the effectiveness of MRPs in producing successful MRs, providing valuable insights for MT in AD-related systems.

Table 5.4: Number of defects revealed by each MRP

MRP	Number of Defects Revealed
$\text{MRP}_{\text{ImpactAmplification}}$	14
$\text{MRP}_{\text{MarginalEnsembleRevision}}$	11
$\text{MRP}_{\text{ScenarioCombinations}}$	3

Table 5.4 further summarizes the effectiveness of the MRPs by presenting the number of defects revealed. $\text{MRP}_{\text{ImpactAmplification}}$, with 14 defects revealed, was highly effective in uncovering inconsistencies or faults in the system. This aligns with the observation from Table 5.3, where this MRP produced the most MRs, and achieved the highest violation rate. The significant number of defects uncovered is evidence of its ability to reveal system defects. $\text{MRP}_{\text{MarginalEnsembleRevision}}$ identified 11 defects. The focus of the MRP, centred on modifying the properties of existing elements, effectively enhances its ability to generate MRs. This is further illustrated by the violation rate presented in Table 5.3, where the number of actual defects found shows a correlation to the number of MRs constructed, highlighting the effectiveness of this MRP. Lastly, $\text{MRP}_{\text{ScenarioCombinations}}$ identified three defects, consistent with its more constrained operational context, as discussed in Section 5.2.3.3. Despite its inherent limitations in generating a large number of MRs, it was still able to uncover actual defects. These findings demonstrate the MRPs' effectiveness in producing high-quality and abundant MRs.

5.2.4 Factors Contributing to MT's Performance

Since the experiments were conducted using both the AD simulator and the ADS, a key question is: *Why were all the defects found related to the simulator rather than the function being tested?* Two potential reasons could explain this.

Firstly, it may be attributed to the nature of simulators, as they typically simulate real-world sensor signals that may not accurately reflect actual scenarios [223]. As a result, there is a higher likelihood that the simulated data may be inaccurate, leading to abnormal outputs of the SUT [167].

Secondly, the AD simulation is only one of the main approaches employed for ADS testing purposes. In the actual production environment, resolving bugs and addressing issues found by real-road testing takes precedence over the findings obtained from simulation testing. This phenomenon can be attributed to both limited human resources and time, and SUT developers' past experiences with false alarms due to simulator inaccuracies. Such situations might diminish the developers' confidence in simulation results, prompting them to prioritize defects found through real-world testing results [130]. However, this situation also underscores the critical importance of validating and ensuring the accuracy of the simulator to maintain its reliability as a testing tool.

5.2.5 Limitations and Conclusions

The experiments in this section highlight the effectiveness of MT in validating AD simulators. Three MRPs and MRIPs were proposed for the purpose of testing the ADS simulator and ADSs. Additionally, a scenario-driven MT framework was presented, for scenario testing that integrates ME and MT, which can significantly reduce the time and effort required for testers to prepare and test a new system. The case study within a real-world industrial setting presented in this study highlights the strengths and limitations of the framework and suggests ways to enhance its utility.

However, testing the simulator in conjunction with the ADS also brought the limitations of the experiments, making it challenging to identify the source of the detected defects. The determination of whether a given issue originated from the simulator or the self-driving system necessitated additional effort and time, resulting in a reduction in testing efficiency. Testing the simulator in isolation would allow any MR violations to be directly attributed to it, which would save time and effort. In the following section, the MT was performed in isolation using the well-known open-source AD simulator CARLA [66], uncovering several significant defects with the proposed MRPs in this section.

5.3 Further Applications of MT in the CARLA Simulator

To further explore the effectiveness of MT and MRPs on AD simulators, in this study, a human-AI hybrid MT framework was proposed to enhance the robustness and reliability of the CARLA simulator. The MT process has led to the discovery of four defects of the simulator[3], thereby highlighting the efficacy and importance of the MRP-generated MRs in ensuring software reliability.

Four fundamental aspects of the CARLA simulator were evaluated: ***Parse***, ***Sim Core***, ***Assessment***, and ***Stability***. Each aspect pertains to the simulator's fundamental performance. These aspects were chosen due to their critical importance in ensuring the simulator's overall performance and reliability, as well as their relevance based on issues identified in our previous research into simulator MT [249].

The ***Parse*** aspect relates to the compilation of scenario elements during simulation processes. Accurate parsing ensures that scenarios are correctly instantiated, forming the foundation for all subsequent simulation processes [173]. Similarly, the ***Sim Core*** aspect encompasses the internal logic and core functionalities that drive the simulation, directly impacting on the realism and accuracy of the simulated behaviours. The ***Assessment*** aspect involves evaluating the built-in

[3]The SUTs used CARLA simulator version 0.9.13 in conjunction with the *scenario_runner*, a separate program that enables scenario definition in ASAM OpenSCENARIO format [13] and initiates the simulator to execute these scenarios.

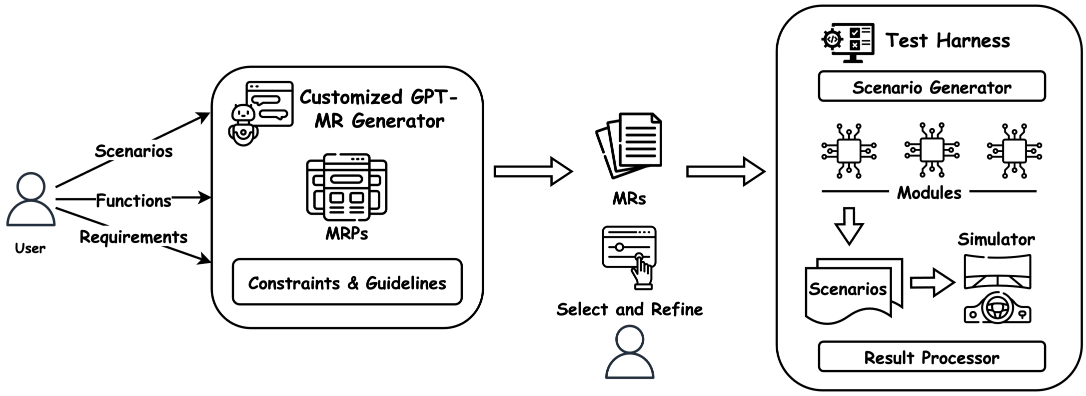


Figure 5.11: Overview of the human-AI hybrid MT framework

criteria for determining scenario success or failure. Effective assessment tools are essential for objectively measuring the performance of ADSs in the simulated environment [138]. Finally, the ***Stability*** aspect ensures that the simulator can consistently reproduce scenarios, which is fundamental for validating the results of repeated experiments and ensuring the reliability of the simulation outcomes.

The SUTs selected for testing are representative examples encompassing each aspect. For the ***Parse*** aspect, the spawn function was chosen, which is the initial step in generating actors in the simulation [136]. For the ***Sim Core*** aspect, the Traffic Manager was selected, which is the module responsible for controlling vehicles in autopilot mode [97]. For the ***Assessment*** aspect, the experiments focused on the built-in evaluation criteria used to determine whether or not a scenario was completed successfully. For the ***Stability*** aspect, the consistency of the simulator were evaluated by executing a single scenario continuously.

5.3.1 Designing a Human-AI Hybrid MT Framework

Figure 5.11 shows an overview of the human-AI hybrid MT framework. The process starts with the user providing essential inputs to generate MRs (including scenarios, functions under test, and specific requirements for testing). These inputs serve as prompts for the GPT-MR generator, a customized GPT model, where MRPs are embedded in the configuration to make the model generate relevant MRs automatically. The MRPs from Section 5.2.1, $\boldsymbol{MRP_{ImpactAmplification}}$ and $\boldsymbol{MRP_{MarginalEnsembleRevision}}$, have been proven effective at generating MRs for AD simulators. The GPT-MR generator operates within predefined constraints and guidelines, ensuring that the derived MRs are accurate and relevant to the given scenarios and functions.

Figure 5.12 shows an example of the MRs generated by the GPT-MR generator (The complete history of interactions can be found in Appendix B). The response starts with an introduction that outlines the goal of the MRs. It is then divided into individual MRs, each following a consistent format. The MRs generated following the same MRP are grouped together. Each MR begins with a source scenario. This is then followed by an MR input relation, which specifies how a follow-up scenario is constructed. For instance, the MRs generated by following $\boldsymbol{MRP_{ImpactAmplification}}$ are grouped under the "Metamorphic Relation Pattern: Impact Amplification" section in the image. Each MR begins with a source scenario, and then followed by an MR input relation, which specifies how a follow-up scenario is constructed. The name "Amplified Scenario" in the response indicates that the follow-up scenarios were generated by following the input relation pattern in the $\boldsymbol{MRP_{ImpactAmplification}}$. For instance, in MR1, the follow-up scenarios are constructed by increasing the density of actors by spawning additional actors in the same region, as specified in the "Amplified Scenario" MR input relation. Similarly, those input relation that follows the pattern in $\boldsymbol{MRP_{MarginalEnsembleRevision}}$ are named by the "Marginal Revision Scenario". The expected outcome describes the MR output relation, which is what the simulator is expected to output. Lastly, the rationale provides the reason for constructing this MR.

Once the MRs are derived, they are passed to the user for review. The user can refine these MRs to ensure that they meet the testing needs. The refined MRs enhance accuracy and abstraction compared to the originally derived MRs. This collaborative step integrates human expertise with AI efficiency, enhancing the

I want to test the spawn function of the simulator. Specifically, I want to test its capability of preventing the actors from spawning at the location that would cause collisions. I want you to generate MRs for me.

To test the spawn function of the ADS simulator, focusing on preventing actors from spawning at locations that would cause collisions, we can generate Metamorphic Relations (MRs) using the provided Metamorphic Relation Patterns (MRPs).

Metamorphic Relation Pattern: Impact Amplification

MR1: Amplified Actor Density

- **Scenario**: Start with a scenario where multiple actors are spawned in a region with no initial collisions.
- **Amplified Scenario**: Increase the density of actors by spawning additional actors in the same region.
- **Expected Outcome**: The simulator should prevent actors from spawning in locations that would result in collisions. The system should either not spawn the additional actors or adjust their spawn points to avoid collisions.
- **Rationale**: This tests the simulator's ability to handle higher densities and ensure collision-free spawning.

MR2: Amplified Complexity

- **Scenario**: Begin with a scenario where actors are spawned around static obstacles (e.g., buildings, barriers) without causing collisions.
- **Amplified Scenario**: Increase the complexity by adding more static obstacles in close proximity to the original spawn points.
- **Expected Outcome**: The simulator should continue to prevent actors from spawning at locations that would cause collisions, despite the increased obstacle density.
- **Rationale**: This ensures the spawn function can handle more complex environments and still avoid collisions.

Metamorphic Relation Pattern: Marginal Ensemble Revision

MR3: Slight Positional Shifts

- **Scenario**: Define a baseline scenario where actors are spawned without causing collisions.
- **Marginal Revision Scenario**: Slightly shift the spawn points of actors by small margins (e.g., a few centimeters or inches).
- **Expected Outcome**: The system should maintain collision-free spawning, as these minor positional changes should not lead to collisions.
- **Rationale**: This tests the precision and robustness of the spawn function in handling small adjustments without resulting in collisions.

MR4: Minor Orientation Changes

- **Scenario**: Start with a scenario where actors are spawned without causing collisions.
- **Marginal Revision Scenario**: Adjust the orientations of the actors by small angles (e.g., 5-10 degrees).
- **Expected Outcome**: The simulator should still prevent any collisions upon spawning, as minor orientation changes should not affect the collision avoidance mechanism.
- **Rationale**: This verifies that the spawn function consistently avoids collisions despite minor changes in actor orientation.

By applying these MRs, you can thoroughly test the spawn function of the ADS simulator to ensure it effectively prevents actors from spawning at locations that would cause collisions, even under varying conditions and complexities.

Figure 5.12: A snippet of answers from the GPT-MR Generator of the spawn function

overall quality and applicability of the MRs, as well as the productivity of their generation.

The refined MRs are then fed into the test harness component. The component includes a scenario generator that creates detailed scenarios in the ASAM OpenSCENARIO format [13], widely used in the industry, based on refined MRs and aimed at various simulator modules. These scenarios are executed automatically under the control of the harness. The result processor analyzes the outcomes, extracts key information from the results, and provides insights into any MR violations.

Unlike existing ADS testing frameworks [154, 249] that depend on predefined MRs for test-case generation and execution, this human-AI hybrid MT framework uses LLMs to streamline MR generation specifically for AD-simulator testing. For example, Pan et al. [154] used predefined MRs to automatically generate test scenarios for foggy images, while Zhang et al. [249] proposed a framework to regulate MT processes and defect orientation in AD simulators. This strategy of bringing LLMs into MR generation could broaden framework applicability to diverse SUTs, though users should possess basic knowledge of MT and the SUT to effectively select and refine MRs.

5.3.1.1 Configuring a GPT-MR Generator

The configuration of the GPT-MR generator is organized into distinct sections: ***Role and Goal***, ***MRPs***, ***Constraints***, ***Guidelines***, ***Clarification***, and ***Customization***. This structured approach ensures that each component is explicitly defined and systematically applied, enhancing clarity, efficiency, and effectiveness in the model response. Configuration presents the detailed configuration of the GPT-MR generator. Notably, while this configuration was applied to customize a ChatGPT model, it does not contain any information specific to

ChatGPT. This suggests that it is general and can also be used to customize the responses of other LLMs.

The ***Role and Goal*** section sets the GPT-MR generator's purpose and intended outcomes. By defining the model's role as an automated generator of MRs for AD simulators, and its goal to enhance robustness and reliability by exploring system behaviour under varying conditions, the model can comprehend user requirements and provide appropriate responses.

The ***MRP*** section outlines the mechanisms of the MRPs in the MR-generation process, including $MRP_{ImpactAmplification}$ and $MRP_{MarginalEnsembleRevision}$. These sections provide detailed examples to illustrate how these patterns are applied, in order to make abstract MRP concepts more concrete and easier to comprehend and adopt for the GPT model.

In the ***Constraints*** section, the boundaries were established within which the GPT-MR generator operates, which ensures that the generated MRs remain relevant to AD simulators and the function under test. Clearly stated constraints help the GPT model to understand the limitations and appropriate use cases of the MR generator.

The ***Guidelines*** section standardizes the MR generation process, ensuring consistency and reliability in outcomes. It provides instructions on the proper steps for how to apply the MRPs.

The ***Clarification*** section addresses scenarios where information might be incomplete. This helps the model to generate the MRs with adequate details and reasonably infer missing specifics from user prompts (e.g., only specify the name of the SUT), which would enhance the adaptability and usability of the GPT-MR generator. This approach allows for effective performance even with partial inputs, improving its utility across various testing environments and user expertise levels.

Finally, the ***Customization*** section tailors the GPT-MR generator's responses to be concise and technical, suitable for developers and testers. It requires the model to output clear explanations and rationales for the generated MRs, enhancing usability and helping users to better understand and refine the results.

5.3.1.2 Designing and Implementing the Test Harness

To enhance testing efficiency, a test harness component was developed. This component automates both the generation and execution of test cases. It provides an interface for users to input parameters that trigger specific functions. These functions include the generation of follow-up test scenarios from a source scenario and the continuous execution of multiple scenarios. The harness is also capable of producing vehicle control sequences to direct the movements of the ego vehicle in simulations. The test harness has four internal modules, each responsible for executing and generating scenarios for different MRs. These internal modules are introduced in the later sections along with the experiment results.

Several features are implemented in the test harness to enhance the testing process:

1. **Dynamic Scenario Creation and Configuration:**
   The harness can dynamically generate new scenarios by processing existing scenarios and modifying their parameters. For example, it can take a source scenario and create multiple variations with different vehicle types or speeds, ensuring a comprehensive coverage of possible scenarios. This capability allows for flexible and reusable test scenarios, significantly improving the scenario-generation efficiency.

2. **Parameterized Randomization:**
   Scenario parameters can be randomized based on the definitions of the MRs, such as the type and speed of actors, or their starting locations within

Configuration GPT-MR MR Generator

1: **Role and Goal:**
- Role: Automated tool to create MRs for ADS simulators.
- Goal: Enhance robustness and reliability by exploring system behaviour under amplified influential factors and marginally revised non-influential factors.

2: **$\mathbf{MRP}_{\mathbf{ImpactAmplification}}$:**
- Amplifies influential input factors that significantly impact system's output.
- Ensures that amplifying these factors either aligns or intensifies the system's output.
- Demonstrates that altering non-influential factors leaves the system's output unaffected.
- Example: Adding more obstacles behind an initial obstacle to observe heightened or mirrored reactions of an ego vehicle.

3: **$\mathbf{MRP}_{\mathbf{MarginalEnsembleRevision}}$:**
- Develops variations with negligible impact on entities not significantly altering scenario's outcome.
- Ensures that the results remain consistent despite minor changes.
- Example: Slight rotations of a vehicle or small positional shifts that do not fundamentally change the vehicle's behaviour or the final result of the scenario.

4: **Constraints:**
- The MR generator must only use the provided MRPs to generate MRs.
- Avoid generating MRs that are outside the scope of ADS simulators or unrelated to the function under test.

5: **Guidelines:**
- Identify influential and non-influential factors in the given scenario.
- Apply $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ to influential factors to observe amplified or consistent outcomes.
- Apply $\mathrm{MRP}_{\mathrm{MarginalEnsembleRevision}}$ to non-influential factors to ensure outcomes remain unchanged.
- Apply $\mathrm{MRP}_{\mathrm{ImpactAmplification}}$ again to initially generated MRs to create further MRs for verifying the reproducibility of outputs.

6: **Clarification:**
- The MR generator should bias towards generating an MR if sufficient information is provided.
- If any information is missing, the MR generator should infer reasonable details based on the provided context.

7: **Customization:**
- The MR generator responds in a concise and technical manner, suitable for developers and testers of ADS simulators.
- It provides clear explanations of the generated MRs and the rationale behind them.

specified ranges. This feature enables the generation of diverse scenarios by slightly altering parameters, ensuring a wide range of possible situations is covered.

3. **Continuous Scenario Execution with Consistent Starting Points:** Multiple scenarios can be executed continuously, with the option to reset the simulation starting state for each one. This ensures consistent starting conditions, eliminating variations caused by residual effects from previous tests. It is particularly useful for MRs that require controlled environments where the initial starting conditions of the scenarios must remain the same.

4. **Queue-based Command Execution for Vehicle Control:** Command sequences can be generated to control the ego vehicle, simulating specific manoeuvres and actions during tests. For instance, sequences such as accelerating for a set duration followed by braking, or executing a series of turns, can be generated. These sequences test the vehicle's response to predefined actions, ensuring consistent and repeatable test conditions.

These features enhance the efficiency and effectiveness of the testing process, enabling thorough exploration of different test scenarios and configurations according to the MRs.

5.3.2 Investigating Actor Spawning Protocols

5.3.2.1 Metamorphic Relations

The spawn function in the CARLA simulator is used to create and place a new actor within the simulation environment [66]. An actor refers to anything that plays a role within the simulator, including cars, pedestrians, and traffic lights [13].

An expected behaviour of the CARLA spawn function would be to prevent the

Table 5.5: Part of MRs from the GPT-MR generator for the spawn function

MR	Source Scenario	Follow-up Scenario	Expected Outcome
$\mathbf{MR_{SlightPositionalShifts}}$	Actors are spawned without causing collisions.	Slightly shift spawn points by small margins.	The system should maintain collision-free spawning despite minor positional changes.
$\mathbf{MR_{AmplifiedActorDensity}}$	Multiple actors are spawned in a region with no initial collisions.	Increase the density by spawning additional actors in the same region.	The simulator should prevent spawning in locations that result in collisions.

instantiation of actors in locations that would lead to collisions. According to the official documentation [136]:

> *"The actor will not be spawned in case of collision at the specified location."*

In other words, the spawn function should prevent the creation of actors if the intended spawning location overlaps with existing actors in a way that would cause a collision. This mechanism ensures that new actors are only generated in clear, unoccupied spaces.

The GPT-MR generator produced four MRs, with two MRs generated under each MRP. The generator was only provided with the information about the function to test. Table 5.5 shows some of the MRs generated by the GPT-MR generator as an example (the complete list of MRs can be found in Appendix B). The MRs under $MRP_{MarginalEnsembleRevision}$ were refined into a single MR, $\boldsymbol{MR_{CollisionAvoidance}}$. This MR focuses on adjusting the starting position and orientation properties of actors within a specific range that reflects the conditions triggering the spawn function's collision-avoidance mechanisms. By carefully varying the parameters, the experiments aim to investigate how the spawn function reacts to situations that could potentially lead to collisions. This approach helps to systematically test and reveal the robustness and responsiveness of the spawn function under different collision-prone scenarios.

> $\boldsymbol{MR_{CollisionAvoidance}}$: In the initial scenario, the first actor is instantiated at a randomly determined location, followed by an attempt to instantiate another actor at that exact position. The spawning function should prevent the second actor's instantiation. Subsequently, in the follow-up scenarios, minor modifications are introduced to the initial position of the second actor, involving adjustments on a per-meter basis along the x, y, or z coordinates, and per-degree variations of the orientation in specific test cases. Provided that these adjustments are executed in a manner that would lead to a collision upon the actor's instantiation, the function should always prevent the instantiation of the second actor, mirroring the behaviour observed in the initial scenario.

To judge whether or not $MR_{CollisionAvoidance}$ violation was occasional, a new MR, $\boldsymbol{MR_{CollisionConsistency}}$ was derived. This MR complemented $MR_{CollisionAvoidance}$ when a violation occurred: If $MR_{CollisionAvoidance}$ was violated, indicating a potential flaw in the spawn function, $\boldsymbol{MR_{CollisionConsistency}}$ replicated the conditions of the initial violation by introducing additional actors in the same manner, to test whether or not the previous MR violation is occasional.

> $\boldsymbol{MR_{CollisionConsistency}}$: If $MR_{CollisionAvoidance}$ is violated, identifying particular scenarios where the spawn function permits the instantiation of two actors resulting in a collision, those scenarios are designated as the source scenario of $\boldsymbol{MR_{CollisionConsistency}}$. In the follow-up scenarios, an attempt is made to instantiate a third actor using the

same approach as in $MR_{CollisionAvoidance}$. In the event that the violation of the previous MR is not an isolated occurrence, actor spawning should continue under predefined conditions, resulting in new instances of collision.

If $\boldsymbol{MR_{CollisionConsistency}}$ is not violated, it suggests a systematic issue with the limitations of the spawn function detected in the violation of $MR_{CollisionAvoidance}$: $\boldsymbol{MR_{CollisionConsistency}}$ was created based on the assumption that the violation of $MR_{CollisionAvoidance}$ could be reproduced. Therefore, no violation of this MR would indicate that the $MR_{CollisionAvoidance}$ violation is not occasional, such that the spawn function's behaviour is uniformly impacted by the identified limitations, providing evidence of a fundamental problem within the implementation of the function itself. This MR and testing would reinforce the reliability of the findings regarding the defects within the spawn function.

5.3.2.2 Harness Module

Figure 5.13 shows the mechanism of the testing module for the spawn function. The process begins by initializing a scenario that includes only the ego vehicle. Upon starting this scenario, the module attempts to introduce another actor close to the ego vehicle, either horizontally or vertically, intending to trigger a collision. If the spawn function operates as designed, the simulator will block the creation of this actor, displaying a warning that spawning is impossible due to a potential collision. The module initiates a subprocess to create follow-up scenarios by systematically adjusting the actor's coordinates (x, y, or z) and orientation, restarting the scenario until the actor is successfully generated. It then checks for collisions and generates a report on the results, including all the necessary information for investigation and debugging.

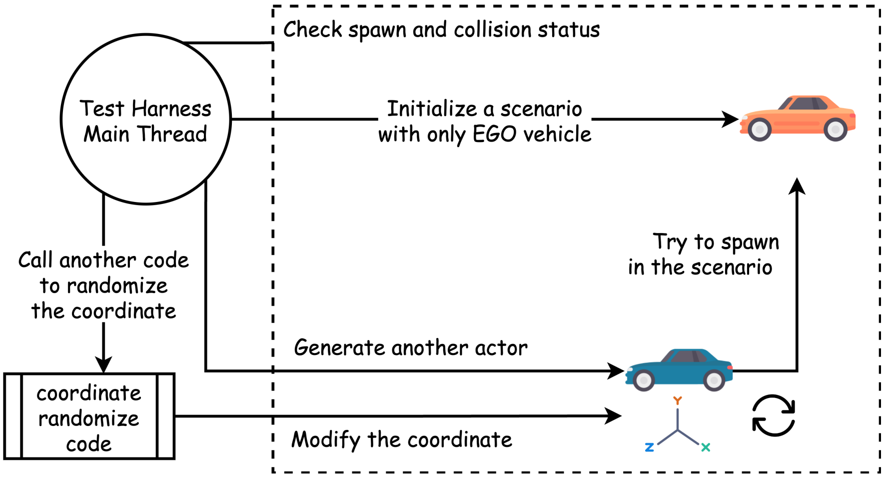


Figure 5.13: Harness module for testing the spawn function

5.3.2.3 Test Results

During the evaluation of $\boldsymbol{MR}_{\boldsymbol{CollisionAvoidance}}$ with the CARLA simulator, a significant flaw was identified related to the actor-spawn function. One violation of this MR occurred when an actor (the upper vehicle in Figure 5.14a) was tried to be spawned at a certain position—a placement that logically led to a collision.

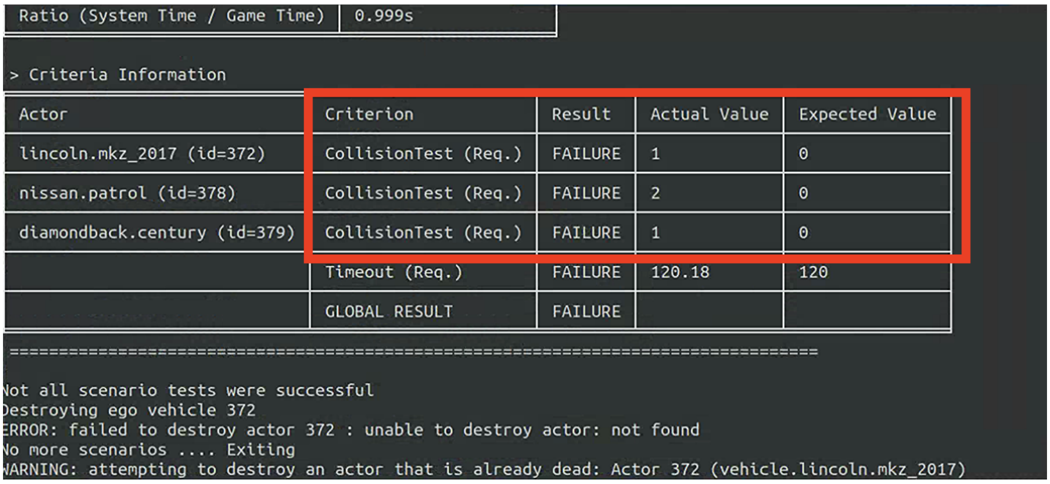
Ideally, the spawn function should recognize such overlapping positions and prevent the actor's creation, thereby avoiding the collision and potentially halting the simulation to address the error (Figure 5.15b). However, contrary to the expected behaviour, the actor was initialized and placed at the problematic location. The result was a physical overlap and actual collision between the actor and the ego vehicle, as illustrated in the simulation outputs (Figure 5.14a and 5.14b). Note that this specific instance represented an MR violation instance regarding the vehicle's certain coordinates and orientation. In other scenarios, such as the upper vehicle oriented left by one degree, the spawn function successfully prevented the actor's spawning. This issue highlights a potential vulnerability in the spawn function to manage actor placements and enforce collision-avoidance protocols.

Further tests were conducted according to $\boldsymbol{MR}_{\boldsymbol{CollisionConsistency}}$ to determine whether or not the observed issues were occasional or indicative of a systemic

(a) The actor was permitted to spawn at a position that would cause a collision

(b) Simulation details indicated that two actors intersect with each other

Figure 5.14: Example of MR violation cases in testing the spawn function

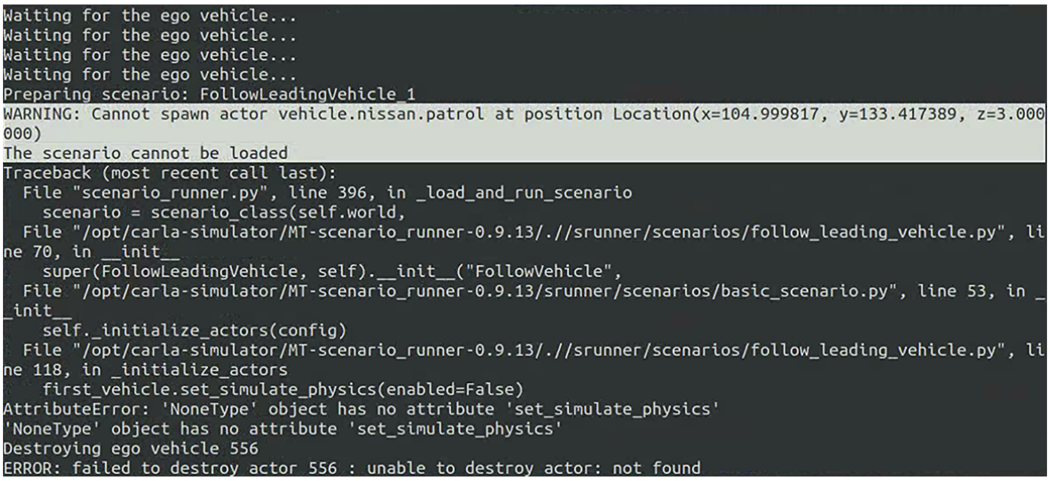


```
Ratio (System Time / Game Time) | 0.999s

> Criteria Information
```

Actor	Criterion	Result	Actual Value	Expected Value
lincoln.mkz_2017 (id=372)	CollisionTest (Req.)	FAILURE	1	0
nissan.patrol (id=378)	CollisionTest (Req.)	FAILURE	2	0
diamondback.century (id=379)	CollisionTest (Req.)	FAILURE	1	0
	Timeout (Req.)	FAILURE	120.18	120
	GLOBAL RESULT	FAILURE		

```
================================================================================

Not all scenario tests were successful
Destroying ego vehicle 372
ERROR: failed to destroy actor 372 : unable to destroy actor: not found
No more scenarios .... Exiting
WARNING: attempting to destroy an actor that is already dead: Actor 372 (vehicle.lincoln.mkz_2017)
```

(a) The collision report proves three actors have collided



```
Waiting for the ego vehicle...
Waiting for the ego vehicle...
Waiting for the ego vehicle...
Waiting for the ego vehicle...
Preparing scenario: FollowLeadingVehicle_1
WARNING: Cannot spawn actor vehicle.nissan.patrol at position Location(x=104.999817, y=133.417389, z=3.000
000)
The scenario cannot be loaded
Traceback (most recent call last):
  File "scenario_runner.py", line 396, in _load_and_run_scenario
    scenario = scenario_class(self.world,
  File "/opt/carla-simulator/MT-scenario_runner-0.9.13/.//srunner/scenarios/follow_leading_vehicle.py", li
ne 70, in __init__
    super(FollowLeadingVehicle, self).__init__("FollowVehicle",
  File "/opt/carla-simulator/MT-scenario_runner-0.9.13/srunner/scenarios/basic_scenario.py", line 53, in _
_init__
    self._initialize_actors(config)
  File "/opt/carla-simulator/MT-scenario_runner-0.9.13/.//srunner/scenarios/follow_leading_vehicle.py", li
ne 118, in _initialize_actors
    first_vehicle.set_simulate_physics(enabled=False)
AttributeError: 'NoneType' object has no attribute 'set_simulate_physics'
'NoneType' object has no attribute 'set_simulate_physics'
Destroying ego vehicle 556
ERROR: failed to destroy actor 556 : unable to destroy actor: not found
```

(b) Under normal circumstances, the spawn function should prevent actors from being spawned above each other which would cause collisions

Figure 5.15: Terminal results of MR violation cases in testing the spawn function

(a) The third actor was also permitted to spawn at a position that would cause a collision

(b) Simulation indicates that three actors collided immediately after the physics simulation was enabled

Figure 5.16: Follow-up MR violation cases in testing the spawn function

problem. Here, the same approach was applied to spawn a third actor. This action was also permitted by the simulator, leading to additional collisions (Figure 5.16a). Notably, when activating the physics simulation option of the simulator (i.e., gravity in the simulation environment), an immediate and simultaneous collision occurred among all actors upon spawning (Figure 5.16b). This outcome confirmed that the spawn function failed to prevent these problematic placements. The collision reports (Figure 5.15a) showed collisions involving all three actors, meaning the MR violation was because of a consistent error within the simulator's spawn function. These findings reveal a significant flaw in the system, indicating a need for reviewing and correcting the spawn function.

After the study of ADSs, the underlying issue was hypothesised to reside in the adjudication process of the models intersecting along the z-coordinate[4]. This hypothesis emerged from two critical observations. Firstly, the modifications on the x or y coordinates did not trigger the bug, suggesting that these dimensions are not the root cause. Secondly, changing the types of actors has also been found to cause MR violations. This suggests that the issue was related to the internal logic of the spawn function.

5.3.3 Assessing Traffic Light Compliance in Autopilot Mode

5.3.3.1 Metamorphic Relations

To ensure vehicle safety in simulations, it is essential to implement functions that adhere to traffic regulations and provide reliable sensor data. One of the key components of CARLA's toolkit is the Traffic Manager, a module that controls vehicles (including the ego vehicle) in autopilot mode in a scenario [97]. The autopilot mode for the actors populates a simulation with realistic urban traffic conditions, allowing for the assessment of various AD algorithms [97]. It provides researchers and developers with the opportunity to test their algorithms in a virtual environment before deploying them in the real world. This is different to testing the traditional ADS functions, where the focus is on evaluating the

[4] In the simulation context, the x-, y-, and z-coordinates represent the three-dimensional spatial positioning of actors in the simulator, where the x- and y-axis define the horizontal plane, and the z-axis represents the vertical height [240].

Table 5.6: The violated MR from the GPT-MR generator for the autopilot function

Source Scenario	**Follow-up Scenario**	**Expected Outcome**
The ego vehicle approaches a junction with a red traffic light under standard environmental conditions.	Introduce minor environmental changes such as slight variations in weather (e.g., light rain or fog) or time of day (e.g., dawn or dusk).	The ego vehicle should still recognize and stop for the red traffic light, regardless of these minor environmental changes.

system's ability to perceive the environment, make decisions, and control the vehicle accordingly [132]; in contrast, testing the Traffic Manager's autopilot mode serves to assess the reliability and fidelity of the simulator itself.

The generator was asked to generate MRs involving a typical scenario of the ego vehicle approaching a junction when the traffic light was red. The GPT-MR generator produced five MRs (the complete list of MRs can be found in Appendix B). Among the generated MRs, MR violations were observed specifically when testing $\boldsymbol{MR_{MinorEnvironmentalChanges}}$, which is presented in Table 5.6. This MR focuses on slight positional shifts, where the ego vehicle's starting position is slightly adjusted by a few meters forward or backward from a junction with a red traffic light. Despite this positional change, the expected outcome is that the vehicle should still recognize the red traffic light and stop. However, scenarios in which the ego vehicle passed through a red light when it was positioned near the stop line at a junction were observed. To better identify the issue, the MR was refined into $\boldsymbol{MR_{StopLineSteadiness}}$, described as follows:

> $\boldsymbol{MR_{StopLineSteadiness}}$: In the source scenario, position the ego vehicle before the stop line at an intersection. The traffic light remains red in all the scenarios. Record the ego vehicle behaviour once the autopilot mode is triggered. In the follow-up scenarios, make slight adjustments to the ego vehicle's initial position, moving it either forward or backward by up to one meter from the stop line. The ego vehicle should

behave the same as in the source scenarios.

Compared to the original MR, this MR has specified the starting position of the ego vehicle, as anomalies were observed in the vehicle's behaviour when it was near the stop line. Observations found that the ego vehicle has an initial movement, a brief deceleration phase, and then an acceleration phase that causes it to run a red light. Importantly, the deceleration phase indicates that the ego vehicle might be able to recognize the red light and take action when it approaches the junction. Therefore, to identify whether or not the defects are within the autopilot system, similar to the testing of the spawn function, $\boldsymbol{MR_{RedLightViolationConsistency}}$ was formulated, based on $MRP_{ImpactAmplification}$. Here, the previous MR violation scenario was set as the source scenario, and follow-up scenarios were generated by placing the starting position of the ego vehicle closer to the junction, specifically at a position where it was in the deceleration phase in the source scenario. Since the ego vehicle would have equal or less time to react to the red light, it should still run the red light, as in the source scenario.

> $\boldsymbol{MR_{RedLightViolationConsistency}}$: When activating the autopilot mode, assuming that starting the ego vehicle at a specific location would result in running the red light. Repositioning the ego vehicle closer to the intersection, approximating the position where the ego vehicle decelerates in the first scenario, the ego vehicle should still proceed through the red light.

5.3.3.2 Harness Module

Figure 5.17 illustrates the functionality of the testing module for the autopilot mode in the simulator's test harness. The main process initializes a scenario where the ego vehicle approaches a turn at a junction with a red traffic light. It then activates the autopilot mode for the ego vehicle through a separate process.

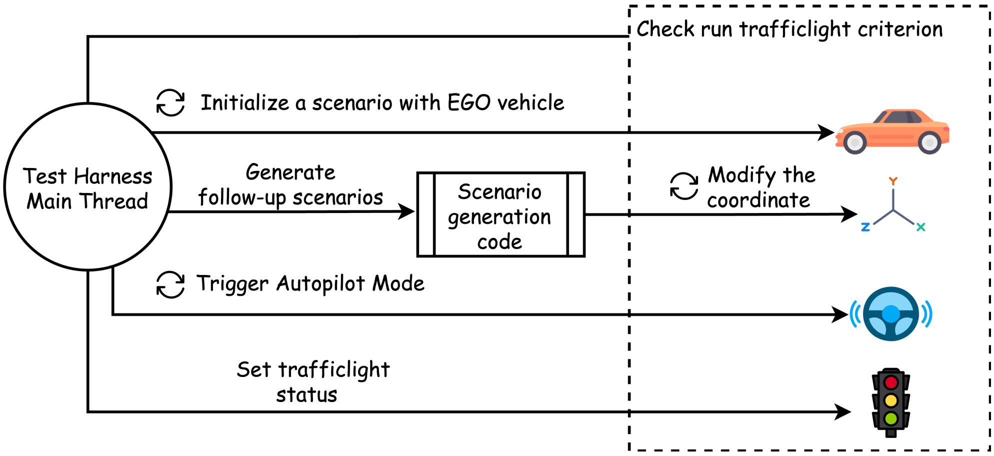
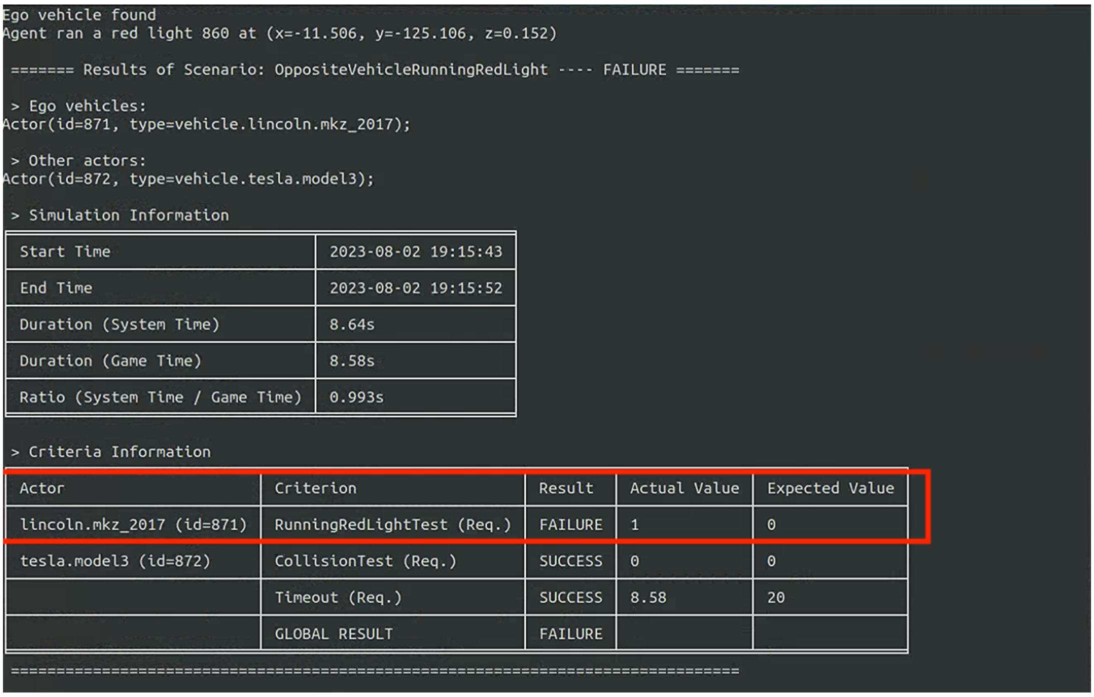


Figure 5.17: Harness module for testing the autopilot mode

Following this, the main process generates follow-up scenarios by adjusting the initial coordinates of the ego vehicle based on the MRs. Furthermore, a subprocess is deployed to monitor whether or not the ego vehicle runs the traffic light, and to compile a test report upon scenario completion. This sequence of processes is repeated until the predetermined number of scenarios has been executed.

5.3.3.3 Test Results

During the testing of $\boldsymbol{MR_{StopLineSteadiness}}$, an anomaly was identified and documented: When starting the autopilot mode of the ego vehicle within a certain range of distances to the stop line in the simulation environment, the ego vehicle would run the red light. To validate and document this behaviour, an evaluation metric integrated within the simulator was used, called the *RunningRedLightTest* [158]. This was designed specifically to assess whether or not vehicles respond correctly to red light signals, making it a tool for verifying traffic-law compliance in simulated environments. The test results consistently indicated a failure of the ego vehicle to stop at red lights. Figure 5.18 shows the results of the test, where *RunningRedLightTest* confirms the run red light behaviour of the ego vehicle (the expected value is 0, indicating the False value in the criterion, while the actual value is 1, such that the test failed).



```
Ego vehicle found
Agent ran a red light 860 at (x=-11.506, y=-125.106, z=0.152)

======= Results of Scenario: OppositeVehicleRunningRedLight ---- FAILURE =======

> Ego vehicles:
Actor(id=871, type=vehicle.lincoln.mkz_2017);

> Other actors:
Actor(id=872, type=vehicle.tesla.model3);

> Simulation Information
```

Start Time	2023-08-02 19:15:43
End Time	2023-08-02 19:15:52
Duration (System Time)	8.64s
Duration (Game Time)	8.58s
Ratio (System Time / Game Time)	0.993s

```
> Criteria Information
```

Actor	Criterion	Result	Actual Value	Expected Value
lincoln.mkz_2017 (id=871)	RunningRedLightTest (Req.)	FAILURE	1	0
tesla.model3 (id=872)	CollisionTest (Req.)	SUCCESS	0	0
	Timeout (Req.)	SUCCESS	8.58	20
	GLOBAL RESULT	FAILURE		

```
================================================================================
```

Figure 5.18: The *RunningRedLightTest* shows the ego vehicle ran past the red light

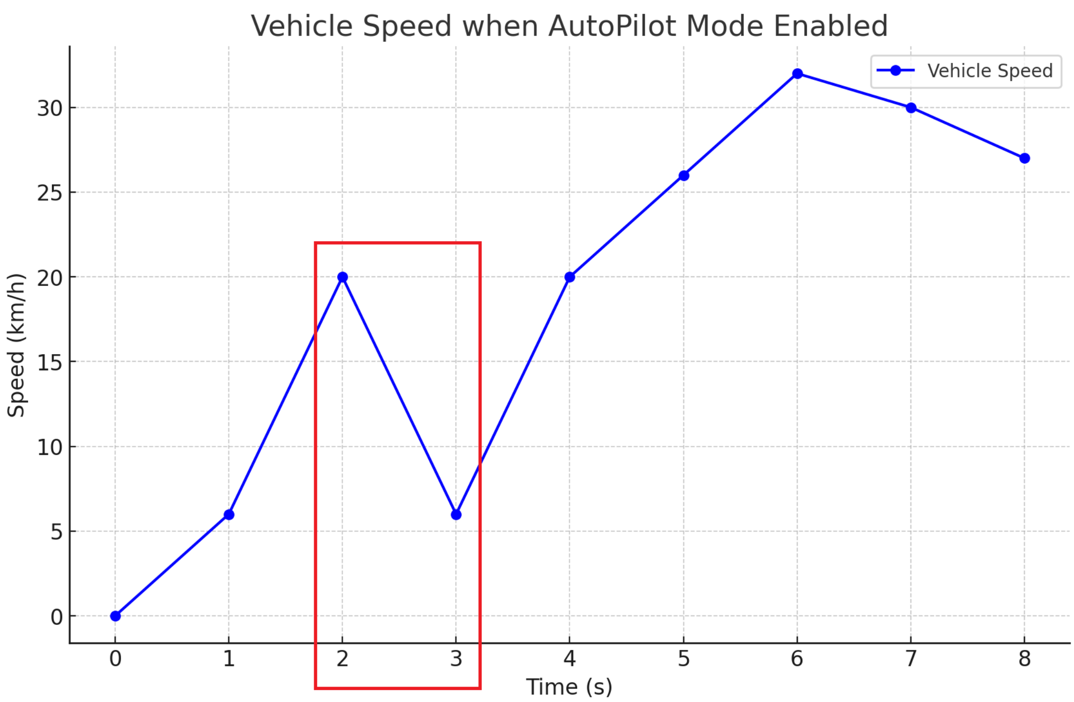


Figure 5.19: The ego vehicle's speed when autopilot mode was enabled in the source scenario

After a thorough examination of the recorded position and speed data, it was clear that the ego vehicle initially decelerated as it approached the red light. Figure 5.19 shows the speed data of the vehicle, with a clear deceleration between time segments '02' and '03'. This indicates that the system had correctly identified the red light and reduced the speed. However, instead of coming to a complete stop, the vehicle accelerated again and continued to run the red light. The initial slowing step of the ego vehicle is shown in Figure 5.20a. It was discovered that the ego vehicle decelerated to a low speed (6 km/h, shown in Figure 5.19) at the stop line, then accelerated again and ran the red light.

To investigate the underlying factors contributing to this behaviour, $\boldsymbol{MR_{RedLightViolationConsistency}}$ was generated and a comparative test was executed: The previous MR violation scenario was chosen to be the source scenario for this MR. In the follow-up scenarios, the ego vehicle was set 10 meters away from the traffic light, specifically at the point where it began to decelerate in the source scenario, and its response to the red light was observed. According to the MR, the ego vehicle should still run the red light at this position. Surprisingly, in this follow-up scenario, the vehicle demonstrated different behaviours to the source scenario: After a short period of acceleration, it accurately recognized and stopped before the red light, which violated $\boldsymbol{MR_{RedLightViolationConsistency}}$. Figure 5.20b illustrates the ego stop location in the follow-up scenario. As a result, the ego vehicle managed to stop before the stop line when the traffic light was red.

Examination of the experimental data points the root of this issue to the internal logic governing the behaviour of the autopilot mode in the *Traffic Manager* module, particularly in the logic of algorithms that respond to the traffic light. These algorithms are designed to create a hazard zone for the system to react to traffic regulators such as traffic lights, stop signs, and priority rules at intersections [97]. Specifically, the issue originates in the *Traffic Manager*'s Traffic

(a) In the source scenario, the ego vehicle failed to stop before the stop line; When it passed the stop line, it accelerated again and ran the red light

(b) In the follow-up scenarios, by placing the ego vehicle slightly closer to the stop line, the ego vehicle could have equal or less time to respond to the red light and stop before the stop line

Figure 5.20: Differences in start positions between the two test scenarios led to distinct behaviours of the ego vehicle

Light Stage, which is tasked with detecting traffic signals and adjusting vehicle behaviour accordingly: When the ego vehicle approaches a red light, the Traffic Light Stage activates a traffic hazard condition to prompt the vehicle to decelerate and stop. The vehicle's initial deceleration indicates correct recognition of the red light. However, if the vehicle does not come to a complete stop before the stop line and slightly crosses it, the *Traffic Manager*'s logic does not consider the vehicle under the influence of the red light (According to the documentation [97], the *Traffic Manager* only "*Sets a traffic hazard if a vehicle is under the influence of a yellow or red traffic light or a stop sign*"). Therefore, once the vehicle crosses the stop line, the Traffic Light Stage removes the traffic hazard condition. As a result, the Motion Planner Stage, which calculates vehicle movements based on active hazards [97], no longer identifies any reason to keep the vehicle stationary. It then commands the vehicle to accelerate and proceed through the intersection, despite the traffic light remaining red.

This logic fails to address instances where a vehicle has partially crossed the stop line but still needs to wait for the green light. Removing the traffic hazard too soon results in the vehicle running the red light. In cases where the vehicle approaches the stop line at a slower speed, it has enough time to come to a complete stop before reaching the line. The traffic hazard condition is still in effect since the vehicle has not left the designated influence zone. As a result, the vehicle correctly remains stopped at the red light.

To address this problem and enhance compliance with traffic regulations, it is recommended that the Traffic Light Stage's logic be modified to extend the traffic light influence zone beyond the stop line: Redefine the hazard zone to include an area beyond the stop line, ensuring that vehicles remain under the influence of the red light even if they have marginally crossed the line. This adjustment would align with realistic driving scenarios where vehicles must wait for the green light even if they have partially crossed the stop line. Furthermore, it is important to

Table 5.7: Contents of $\mathrm{MR}_{\mathrm{SubtleBehaviouralChanges}}$

Source Scenario	Follow-up Scenario	Expected Outcome
A scenario where the ego vehicle drives at a constant speed and adheres to all traffic rules, completing the route successfully.	Introduce subtle changes in the ego vehicle's behaviour, such as slight variations in speed or minor adjustments in lane positioning, without violating traffic rules.	The evaluation criteria should still assess the scenario as successful if the ego vehicle completes the route correctly, despite the subtle behavioural changes.

maintain the traffic hazard condition until the light turns green, ensuring that the removal of the hazard is based not only on the position of the vehicle but also on the status of the traffic light. This means the hazard condition should remain in effect until the light changes to green to ensure the safety of roads and passengers.

5.3.4 Evaluating the Robustness of Built-in Criteria

5.3.4.1 Metamorphic Relations

Whether the ego vehicle is controlled by an ADS-like function such as autopilot, or operated by a human, it is important that the evaluation criteria can reliably determine the validity of the scenario. The *scenario_runner* is a component of the CARLA simulator that can start simulations from predefined scenario files [79]. It contains multiple evaluation criteria that can be used to analyze whether or not a scenario was completed successfully [158]. For instance, checking the collision status of the vehicle, or whether or not the vehicle has invaded other lanes during the simulation.

The GPT-MR generator was asked to generate MRs to test the built-in simulation criteria in the CARLA simulator. The generator produced five MRs, which can be found in Appendix B. Experiments with these MRs did not reveal MR violations. Among these, $\boldsymbol{MR_{SubtleBehaviouralChanges}}$ (see Table 5.7) was selected for closer examination because it specifically addresses the internal consistency of vehicle behaviour—a critical aspect that directly impacts the reliability of the simulation results. The other MRs primarily focus on external factors, such as environmental complexity or spatial adjustments, which, while important, do not examine the underlying behaviour of the vehicle itself.

In Section 5.3.3, MR violations exposed defects in the autopilot function, particularly in its logic for handling traffic lights. This function, due to its complexity and centrality to vehicle control, has a high potential for revealing additional defects of other parts of the simulator. By focusing on behavioural changes, $\boldsymbol{MR_{SubtleBehaviouralChanges}}$ is suited for identifying inconsistencies in how the ego vehicle's behaviour impacts the evaluation criteria when minor adjustments are made.

One of the limitations of $\boldsymbol{MR_{SubtleBehaviouralChanges}}$ is its reliance on the control method (e.g., autopilot) remains constant, potentially overlooking variations in outcomes that could arise when different control methods are applied. In real-world scenarios, the ego vehicle may be controlled by different methods, either from automated systems like autopilot, or from manual human control. These varying control methods can introduce different behavioural dynamics, which the evaluation criteria must be able to consistently assess to ensure the scenario's validity. For instance, an ADS might respond more quickly and predictably to obstacles than a human driver, who may have slower reaction times and more unpredictable behaviours. If the evaluation criteria do not account for these differences, they might not accurately measure performance across various control methods. Therefore, the evaluation framework needs to be adaptable, providing fair assessments regardless of how the vehicle is controlled.

Therefore, to ensure the robustness of criteria functionalities across different control methods, $\boldsymbol{MR_{ControlMethodCriteriaConsistency}}$ was developed, as an extension of $\boldsymbol{MR_{SubtleBehaviouralChanges}}$ using $MRP_{MarginalEnsembleRevision}$. This MR states that

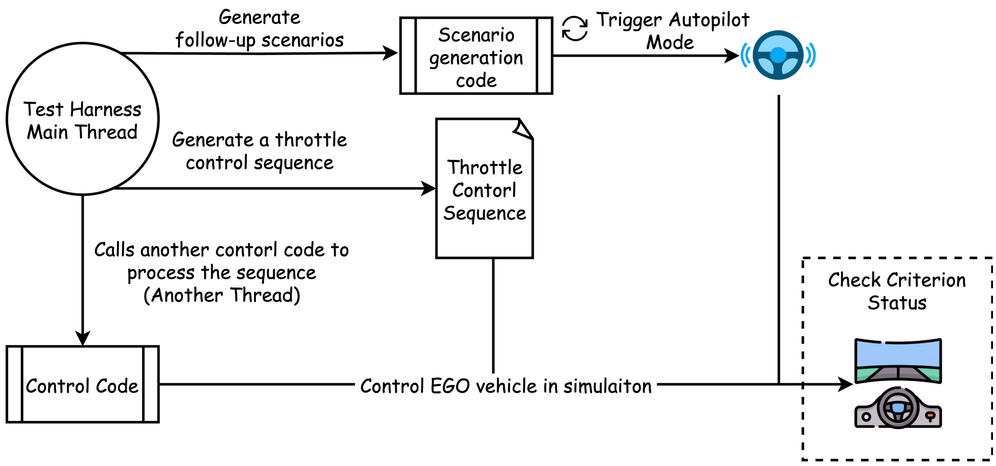


Figure 5.21: Harness module for testing the evaluation feature

changes in the control methods—whether automated, manual, or code-driven—should not affect the ability of the criteria under test to accurately monitor the state of the actors in success or failure.

> $\boldsymbol{MR_{ControlMethodCriteriaConsistency}}$: Whether the ego vehicle is controlled by autopilot or manual control (such as code sequences), changing how the vehicle is controlled should not affect the final outcomes of the evaluation criteria in the scenarios.

5.3.4.2 Harness Module

Figure 5.21 illustrates the operation of the testing module for evaluating criteria within the simulator's test harness. The harness's main process generates a series of control sequences, each representing a test case based on MRs. These sequences simulate typical keyboard inputs for vehicle control in the simulator, such as the 'w', 'a', 's', and 'd' keys, along with their timing. For instance, the sequence "$[[w, 4], [s, 2]]$" directs the simulator to fully engage the throttle (like holding down the 'w' key) for four seconds and then apply brakes for two seconds. To ensure that these inputs are recognized by the simulator, the main process of the test harness calls a separate control script in another process to interpret the sequence and manage the ego vehicle's movements in the simulation. This

```python
velocity = CarlaDataProvider.get_velocity(self.actor)

self.actual_value = max(velocity, self.actual_value)

if velocity > self.success_value:
    self.test_status = "FAILURE"
else:
    self.test_status = "SUCCESS"
```

(a) Original piece of the *CheckMaximumVelocity* test code

```python
velocity = CarlaDataProvider.get_velocity(self.actor)

self.actual_value = max(velocity, self.actual_value)

if self.actual_value > self.success_value:
    self.test_status = "FAILURE"
else:
    self.test_status = "SUCCESS"
```

(b) Updated piece of the *CheckMaximumVelocity* test code

Figure 5.22: Part of the *CheckMaximumVelocity* test code

approach guarantees precise control of the throttle and brakes, enhancing testing efficiency while minimizing the risk of human error in the results' validity.

The test harness also calls a scenario-generation module and generates follow-up scenarios with the autopilot mode enabled. The source and follow-up scenarios (one controlled by the throttle-control sequence and the other controlled by the simulator) are then executed in sequence, with a result-comparison program automatically comparing the outputs of the criterion to verify their equivalence.

5.3.4.3 Test Results

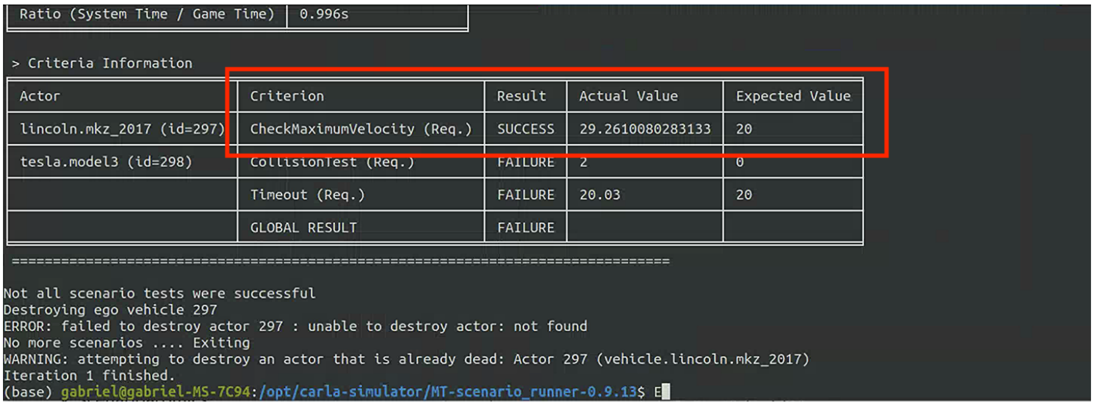
When testing the $\boldsymbol{MR}_{\boldsymbol{ControlMethodCriteriaConsistency}}$ in a scenario involving the ego vehicle's motion along a straight road and passing through a junction, an issue emerged within the testing procedures. This issue was related to the *CheckMaximumVelocity* [158] criterion. It criterion compares the ego vehicle's actual maximum speed against an expected value. If the actual speed is less than the expected value, the criterion will give the "SUCCESS" value; otherwise, it will report the "FAILURE" value (Figure 5.22a).

In the validation of this criterion, the control sequence was designed to fully engage the throttle of the ego vehicle for an intended period, allowing the vehicle to reach high speeds. This setup facilitates the testing of various maximum speeds and enables the creation of follow-up test scenarios that explore different speed thresholds. The main thread generated and executed the follow-up scenarios by altering the speed values within the criterion, while another process monitored the test criterion's outputs, compiling a report upon the completion of each scenario.

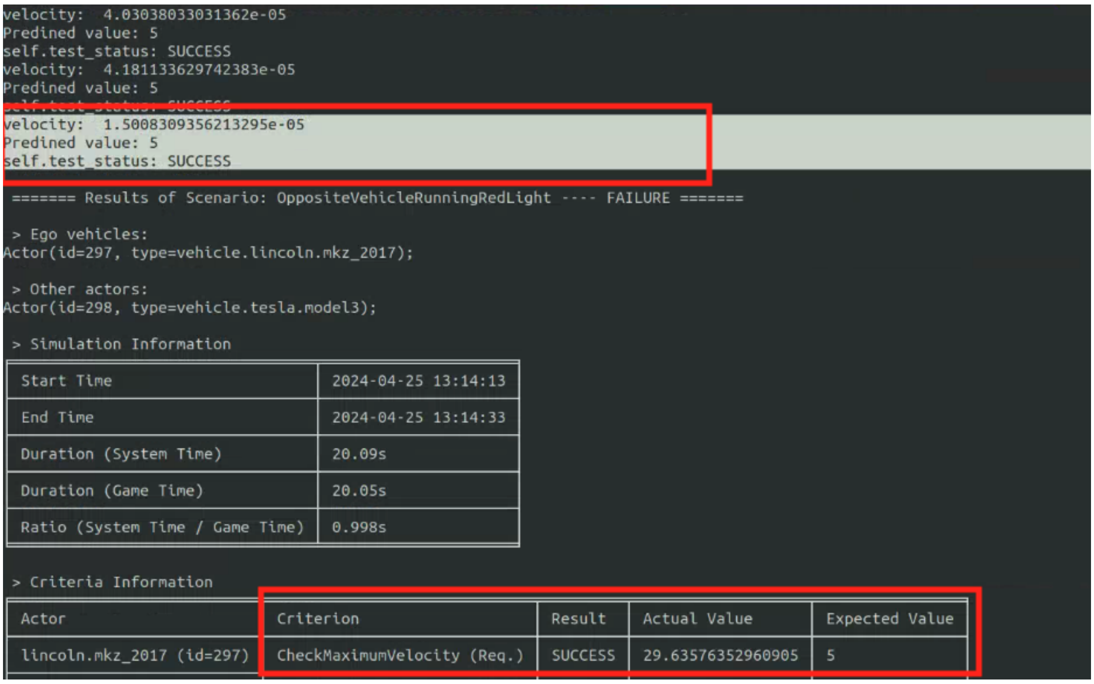
During the experiments, an MR violation when switching the control methods of the ego vehicle from autopilot to control sequence was identified: When the ego vehicle was driving under the autopilot mode, the *CheckMaximumVelocity* criterion gave the correct output. However, when the code sequence was employed to control the ego vehicle's throttle, such as pushing it to its limit for a long period to simulate a relatively extreme case, the anomaly became evident: The *CheckMaximumVelocity* criterion reported "SUCCESS" instead of "FAILURE" when the actual maximum speed value exceeded the expected value. In other tests, the results revealed that the *CheckMaximumVelocity* criterion consistently reported "SUCCESS", regardless of whether or not the actual maximum speed surpassed the predefined threshold (Figure 5.23a).

An examination of the module's internal logic identified the problem: the code did not use the correct maximum speed value for comparison with the predefined threshold. As shown in Figure 5.22a, the criterion update was based on comparing the real-time velocity to the expected value rather than the actual maximum speed ('self.actual_value'), despite the developer having updated the variable in the previous step, which may because of inaccurate coding practices. By modifying the code to compare the actual maximum speed with the predefined threshold (illustrated in Figure 5.22b), the problem was successfully resolved, and the criterion gave the correct output when the maximum value exceeds the expected value (Figure 5.23c).

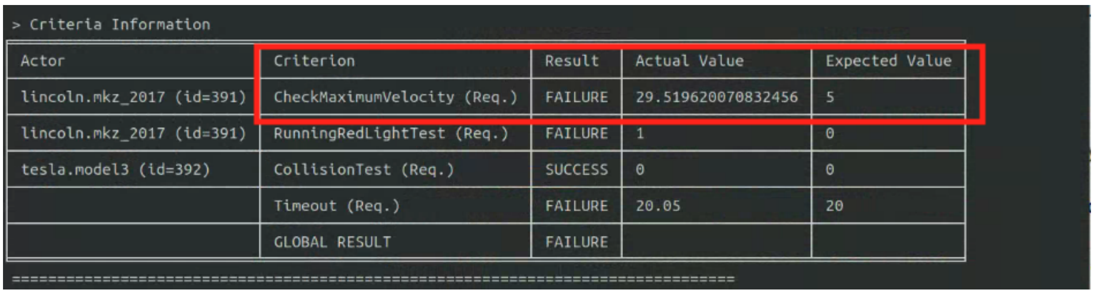
Additionally, the potential cause of why this issue was found only when switching from autopilot mode to manual control was revised. The default expected speed set by the developer was not achieved in autopilot mode due to considerations of other road factors such as speed limits, junctions, and traffic lights. As a result, the issue was not revealed under autopilot mode. However, the manual code



(a) The *CheckMaximumVelocity* test shows SUCCESS when the actual value exceeded the expected value



(b) The *CheckMaximumVelocity* test has not updated the final result to the global result: it only stores the final state



(c) After revising the inner logic of the criteria, the *CheckMaximumVelocity* test gave the correct the results

Figure 5.23: The "CheckMaximumVelocity" test results in scenarios with different value thresholds

control uses a programmed control sequence (for instance, pushing the throttle to the max for ten seconds) that does not consider such limits: The actual maximum speed value of the ego vehicle can exceed the predefined expected value, thus revealing the bug.

5.3.5 Analyzing Simulation Consistency Across Multiple Runs

5.3.5.1 Metamorphic Relations

Ensuring consistent simulation performance across multiple runs—where the simulation produces the same results under the same conditions—is an important objective in the context of simulators [201]. In simulations, actors should respond consistently and promptly to certain triggers. Among all the MRs generated by the GPT-MR generator (the complete list of MRs can be found in Appendix B), when testing the $\boldsymbol{MR_{RepeatedRunsWithAmplifiedFactors}}$ (Table 5.8), MR violations were identified in a scenario involving the interaction between the ego vehicle and a cyclist. This MR tests the consistency of a simulator when the same scenario is run repeatedly under identical conditions. In this MR, the ego vehicle drives through a predefined route with standard traffic and environmental conditions. The scenario is executed several times (for example, 10 times), with the expectation that the simulation results remain consistent throughout all iterations, meaning that the actors and environments behave similarly across different runs. To enhance the simplicity and applicability of the MR, it was refined into $\boldsymbol{MR_{ScenarioReproducibility}}$. This MR aims to identify potential flaws or limitations in the simulation engine by comparing repeated and individual scenario runs.

> $\boldsymbol{MR_{ScenarioReproducibility}}$: Executing the same scenarios, whether run repeatedly or individually, should always produce consistent outputs.

Table 5.8: Contents of $\text{MR}_{\text{RepeatedRunsWithAmplifiedFactors}}$

Source Scenario	Follow-up Scenario	Expected Outcome
A baseline scenario where the ego vehicle drives through a predefined route with standard traffic and environmental conditions.	Increase the number of runs for the same scenario under identical conditions (e.g., 10 repeated runs).	The simulation outputs should be consistent across all runs, showing similar ego vehicle behaviours, routes, and interactions with other entities.

Disparities between these runs could indicate issues with the simulator's reliability and repeatability, making this validation process essential for ensuring consistent and accurate simulation results.

5.3.5.2 Harness Module

Figure 5.24 illustrates the mechanism of the testing module designed to evaluate the consistency of the simulator. This is achieved by executing the same scenario multiple times, as indicated in the diagram. The main process of the test harness is responsible for initiating and repeatedly running the scenario until the number of executions reaches the value specified by the tester. The module can facilitate the sequential execution of multiple scenarios while reloading the simulation state for each scenario. This ensures that each simulation starts from the same point, eliminating variations due to residual effects from previous tests. Additionally, a subprocess is employed to compare the scenario outputs. This helps to identify any differences in the simulation outputs across different runs of the same scenario.

5.3.5.3 Test Results

While executing a scenario focused on an interaction between a vehicle and a cyclist during a right-turn manoeuvre, inconsistencies were observed in the cyclist's behaviour across multiple runs. This inconsistency in behaviour led to different simulation outcomes, prompting a comprehensive examination.

The scenario simulates a simple object collision between the ego vehicle and a

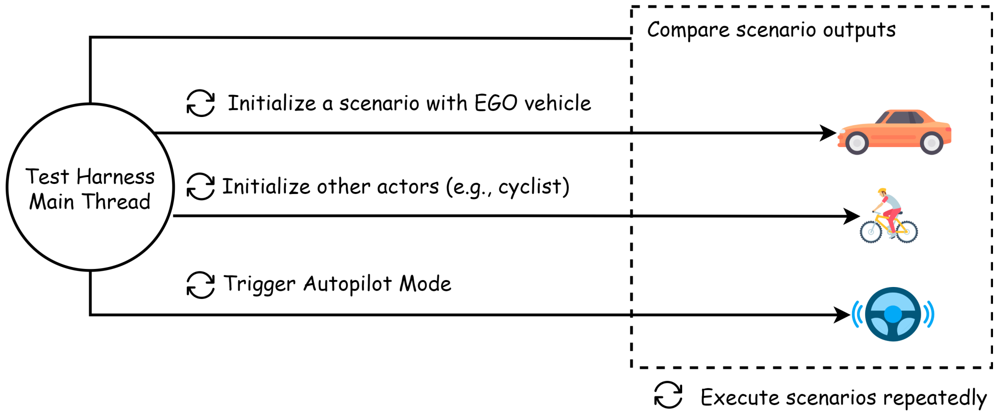


Figure 5.24: Harness module for testing the consistency of the simulator

cyclist. According to the predefined scenario parameters, the ego vehicle turns on autopilot mode and navigates through a road until encountering a cyclist after a right turn. The cyclist is positioned to await the entry of the ego vehicle in a certain region. Once this condition is met, the cyclist begins crossing the road. The objective is for the ego vehicle to successfully avoid a collision following the right turn, subsequently resuming its journey after the road is clear.

During multiple executions of the scenario, a discrepancy was observed in the cyclist's behaviours when it started. The issue relates to the cyclist's acceleration phase, which is triggered when the distance between the cyclist and the ego vehicle reaches a predefined threshold. In the first iteration of the scenario, the cyclist immediately accelerated upon the trigger condition was satisfied—a behaviour that aligns with expectations. The simulation had the ego vehicle wait until the cyclist had completed the crossing, and then continued driving (Figure 5.25a). However, in other iterations, the cyclist exhibited an obvious delay (e.g., one or two seconds) in acceleration after the condition was satisfied. In such circumstances, the ego vehicle could not stop in time to avoid the collision (Figure 5.25b).

An in-depth analysis of the open-source *scenario_runner* codebase led to an examination of the *KeepVelocity* class. According to the specification [78], this class

(a) The source scenario where the cyclist responded appropriately

(b) The follow-up scenario where the cyclist activated unexpectedly

Figure 5.25: The trigger timings of the cyclist show a large difference between the source and follow-up scenarios

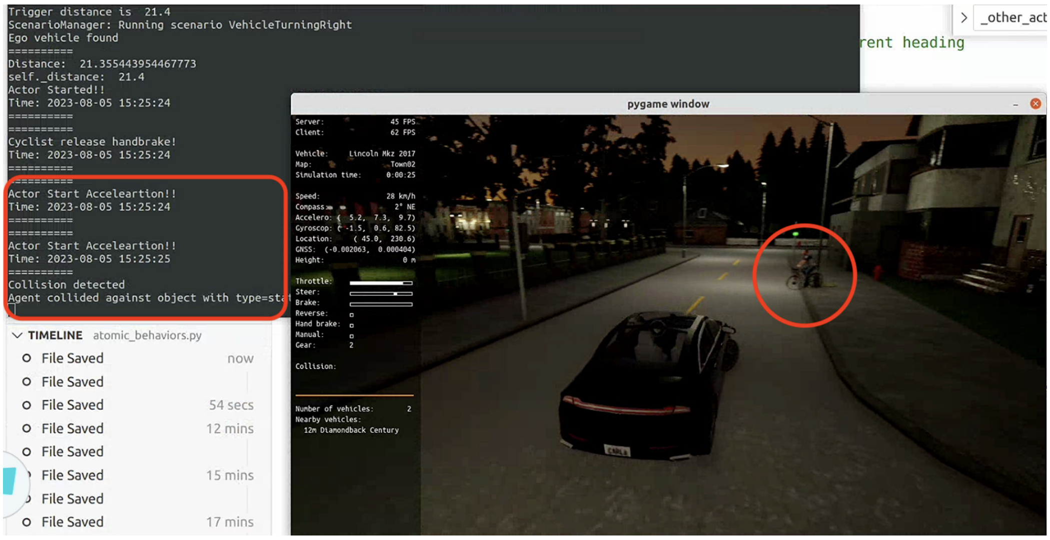
includes a function to maintain the specified velocity: The controlled traffic participant should accelerate as quickly as possible until achieving the target velocity, which would then be maintained for the duration of the behaviour. To record and compare the starting time of the cyclist's acceleration process, supplementary code was introduced into the *KeepVelocity* class. This added code was designed to log the precise timing when the cyclist started acceleration.

The data revealed a delay between the output of the "Start Acceleration" message in the terminal and the actual execution of the cyclist's road-crossing action. This was visually apparent in the simulation window, providing evidence of the latency (Figures 5.26a and 5.26b). The hypothesis was that this latency might be attributed to computational overhead within the *KeepVelocity* class or its associated components, especially when executing the scenario multiple times continuously, considering the issue was not apparent when executing the scenario in singular times. Additionally, the scenario_runner might contain timing inconsistencies or race conditions that become more pronounced during repeated executions. The variability in system load during continuous runs could also influence the simulation's performance, with higher loads causing greater inconsistencies in the behaviour of traffic participants [115]. To address this issue, the potential improvements would be identifying and resolving any bottlenecks or inefficiencies within the *KeepVelocity* class. This could involve using performance analysis tools to locate and improve the algorithms where the code is consuming excessive computational resources or causing delays.

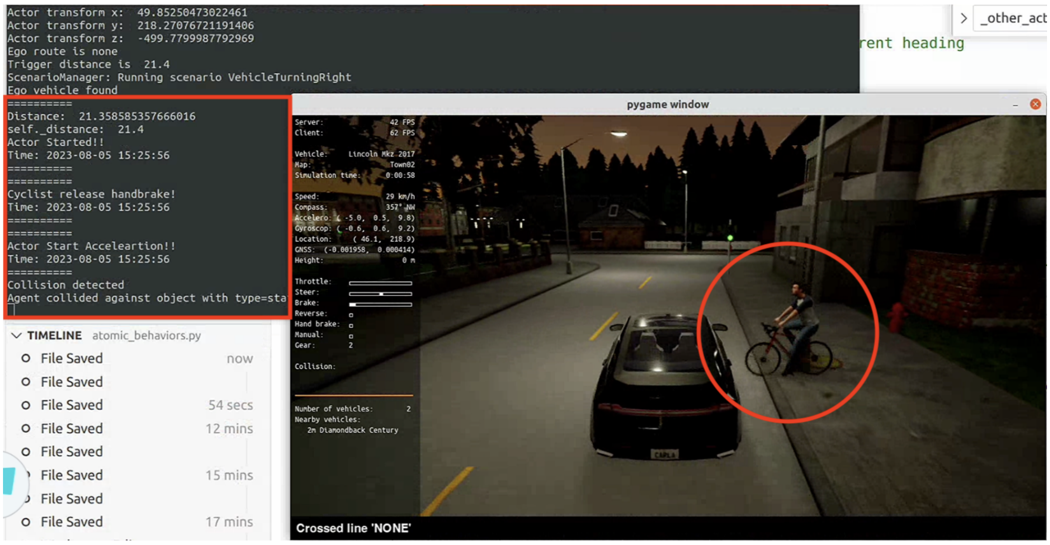
5.3.6 Discussion

5.3.6.1 Effectiveness of MT in Testing the CARLA Simulator

In this study, the use of MT has demonstrated efficacy in verifying the reliability and accuracy of the CARLA simulator. The integration of MRPs into this testing



(a) Under normal circumstances (source scenario), when the acceleration start message is sent, the cyclist should start to cross the road



(b) In the abnormal scenarios (follow-up scenarios), the cyclist experienced a clear delay to start acceleration when the message was sent

Figure 5.26: The comparison of the cyclist actions in the source and follow-up scenarios

framework has been valuable, serving to accelerate and simplify the process of generating MRs. The inclusion of configuring a GPT would be especially beneficial for individuals new to the CARLA simulator, providing a structured guide that facilitates effective testing even for those with limited familiarity with the system and MT.

The MRs in this study led to the rapid discovery of four significant and diverse issues in the simulator, each impacting its real-world applicability in distinct ways. For example, the last issue associated with latency in code execution pinpointed a potential bottleneck that could reduce the simulator's efficiency and accuracy in the continuous execution of scenarios.

The issues detected in this study could have easily remained hidden using conventional testing methods, reinforcing the role that MT—and specifically MRPs—plays in deeply testing the simulator's functionality. For instance, in the testing of $\boldsymbol{MR_{CollisionAvoidance}}$, traditional software tests with fixed oracles, might conclude their evaluation after verifying that the spawn function behaves correctly under several certain test cases, such as preventing the instantiation of a second actor at the same location as the first actor, as described in the source scenario of the MT. In contrast, the MT approach goes further by slightly altering the location of the second actor to find the potential point where the two actors collide, and using the same approach to spawn a third actor to prove the existence of the error. Through the application of MT, researchers can methodically uncover inconsistencies or unexpected behaviours arising from complex interactions among the simulator's components. This, in turn, contributes to improving the overall reliability and performance of the simulator.

In summary, the application of MRPs within MT not only aids in identifying issues but also significantly increases the ability of derived MRs to expose a broad range of flaws. The variety and importance of the issues uncovered serve as a testament to the efficacy of MT, further enhanced by the optimized MR-generation approach provided by MRPs. It ensures a more comprehensive and efficient evaluation, making it a robust tool for uncovering vulnerabilities and improving system reliability.

5.3.6.2 Threats to Validity

While the findings presented in this study offer valuable insights into the effectiveness of MT for testing the CARLA simulator, the author also acknowledges certain potential threats to the validity of the results. These threats stem from factors that could potentially impact the generalizability and reliability of the outcomes.

1. MR Selection: The selection of MRs in this study may have shaped the kinds of issues that were uncovered. Due to time and resource constraints, the author focused on specific, limited aspects of the simulator for testing. Although the MRs chosen have facilitated the identification of issues in various aspects of the simulator, exploration of alternative relations could potentially yield further insights into the simulator's behaviour.

2. Simulator Configuration and Versions: The issues identified could be influenced by the specific configuration and version of the CARLA simulator used in the study. For example, such issues may manifest within particular versions of the software or may be obscured or mitigated by updates to other components. Different configurations or simulator versions might yield varying results.

3. Test Scenario Representation: The test scenarios used in this study could not encompass the full spectrum of real-world driving situations. Variations in scenarios and environmental conditions could lead to different issue-discovery rates.

4. Human Bias: Human involvement in selecting the MRs and interpreting the results could introduce biases that impact on the objectivity of the study.

5. Reproducibility of Results: The evolving nature and stochasticity of LLMs might affect the reproducibility of the results [44]. However, the proposed configuration in this study standardizes the response mechanism without modifying the underlying training data. As models are further trained with expanded datasets, their accuracy is expected to improve, potentially enhancing the quality of the generated MRs.

5.3.6.3 Performance of the GPT-MR Generator in This Study

Previous research [247] found that ChatGPT can generate useful MRs, but the quality and correctness were unimpressive. In other words, not all the MRs generated were correct and effective in the initial response of the model. This performance may be attributed to the quality of training data and the capability of the models (GPT-3.5 in that study [234]). However, with the usage of GPT-4 in this study (the base model behind the GPTs by OpenAI, and the current most powerful ChatGPT version [151]), and the configurations involving the MRPs and guidelines, the MRs generated by the GPT-MR generator have shown obvious improvements in quality and clarity. Specifically, the MRs were correctly generated following the MRPs; they could be applied directly to testing; and some have led to the discovery of defects. Additionally, the situation where users needed to provide multiple prompts to improve the quality of MRs in the previous study did not occur with the GPT-MR generator: The configurations limited the search space of the customized GPT and improved the accuracy of the answers, allowing the GPT to provide responses that could be directly adopted [92].

However, based on the observations from the MRs generated (Appendix B), it also found that the diversity of the MRs was limited. For instance, among the MRs derived by the model for different functions under test, there is always one MR that includes increasing the traffic density/complexity of the scenario. In other words, the innovative features of the answers are limited, which is a common issue among GPT-generated responses [31]. Additionally, the selection and refinement of the MRs derived by the GPT still require users to possess a basic knowledge of MT, even though this knowledge can be easily obtained through user interactions with the GPT.

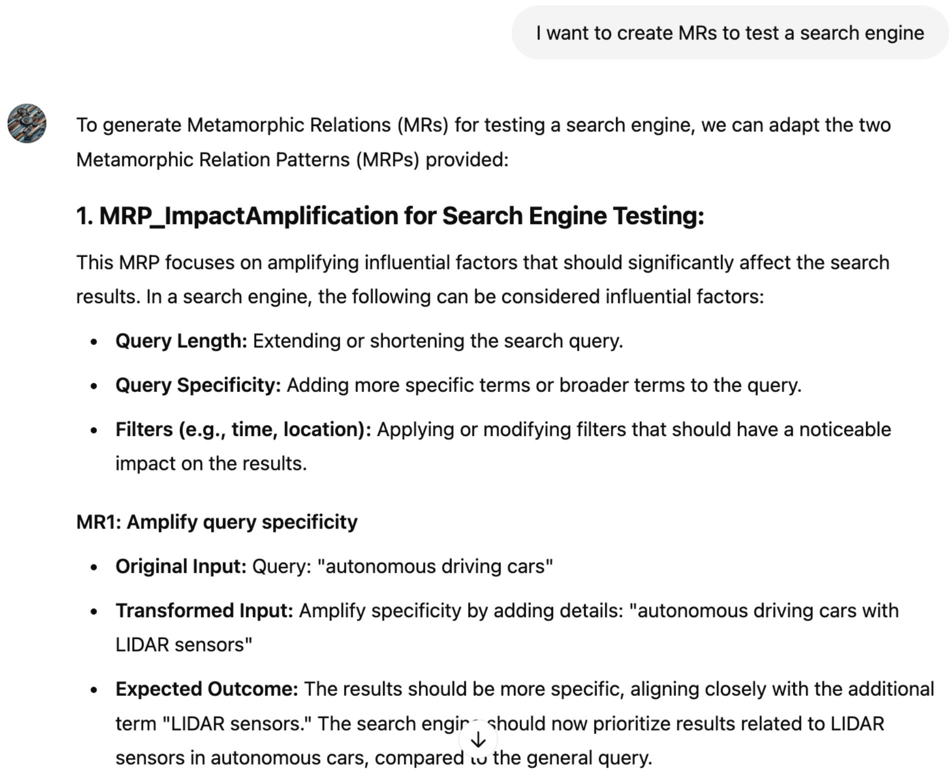
Although this GPT-MR generator was originally designed for AD simulator testing, it can be easily applied for deriving MRs for other domains. A short study was conducted by asking the model to derive MRs for SUTs in different domains. Two types of the SUTs that have been tested by MT (i.e., there were MRs created to test them and revealed bugs in the literature) were selected — search engine and navigation software, each is a well-represented example in MT [256][30] with a high citation count (100+), and they are commonly used in everyday scenarios, such as Bing for search engines [256] and Google Maps for navigation [30]. To provide simple and straightforward evaluation of these MRs, the evaluation criteria, $Criteria_{V0}$, described in Section 4.2.1, was used [247]. Furthermore, to avoid any potential bias from previously published MRs that could have influenced the training of ChatGPT, the model was not asked to derive MRs for any specific systems from the existing literature. Instead, straightforward and generalized prompts was used, such as "I want to create MRs for a search engine" and "I want MRs for the routing function of a navigation application", simulating scenarios that beginners might encounter while using this tool. For instance, Figure 5.27a shows an example snippet of asking the GPT-MR generator to derive MRs for the search engine.

Additionally, to address potential validity concerns regarding the use of these MRs in training the GPT and to ensure the model is "inferring" answers [213] rather than "retrieving" the MRs from the database, the GPT-MR generator was also asked to produce MRs for SUT types that had not been tested by MT.

Specifically, the model was tasked with generating MRs for an email program and an ordering system, both of which are widely used in real life. Figure 5.27b shows an example of MRs derived for the email app. The complete list of MRs can be found in Appendix C.

Four experienced evaluators from diverse domains assessed the quality of the MRs. The average marks for each system were calculated and are presented in Table 5.9. The table indicates that the MRs for all four SUTs scored well (over three out of five) in correctness, applicability, clarity, relevance, and computational feasibility. This suggests that the MRs are generally accurate and useful, demonstrating that the GPT-MR generator can effectively produce valuable MRs for various SUTs beyond just ADS testing. The novelty and overall usefulness scores were relatively lower, consistent with findings from a study on ChatGPT's MR generation capabilities [247]. Notably, some MRs for SUTs tested with MT (such as search engines and navigation software) showed similarities to existing literature, as shown in Table 5.10. For example, both the MRs generated for the search engine and those found in the literature involved reordering query terms, while both MRs for navigation software involved minor positional shifts in starting and destination points that would leave the route unchanged. This suggests that the generator is capable of producing MRs that are comparable in quality to those created by humans. Furthermore, for the types of SUTs that are untested with MT, the MRs derived were also valuable. Table 5.11 presents two examples of MRs from the email app and ordering system. Both examples have clear input-output relations with appropriate rationales, making them easy to understand and effective for testing. These evaluation findings and MR examples highlight the capabilities of the GPT-MR generator in producing MRs across different domains.

In summary, using the capabilities of GPT-4, along with the configurations of MRPs and guidelines, has enhanced the quality and applicability of MRs derived



I want to create MRs to test a search engine

To generate Metamorphic Relations (MRs) for testing a search engine, we can adapt the two Metamorphic Relation Patterns (MRPs) provided:

1. MRP_ImpactAmplification for Search Engine Testing:

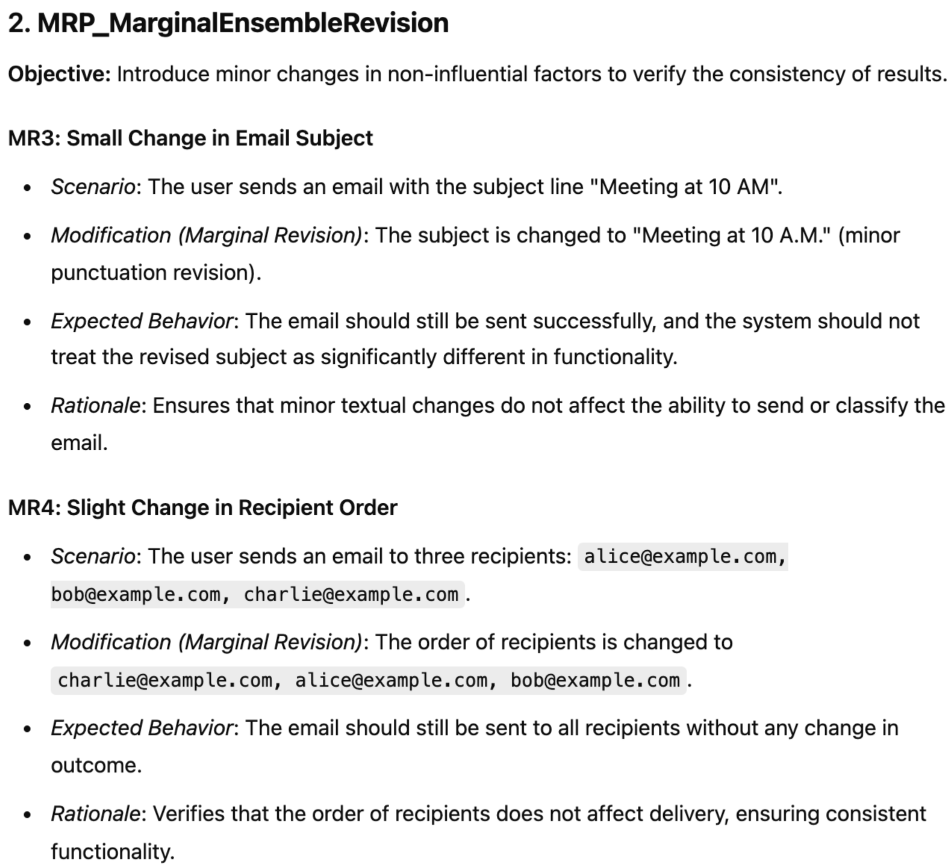
This MRP focuses on amplifying influential factors that should significantly affect the search results. In a search engine, the following can be considered influential factors:

- **Query Length:** Extending or shortening the search query.
- **Query Specificity:** Adding more specific terms or broader terms to the query.
- **Filters (e.g., time, location):** Applying or modifying filters that should have a noticeable impact on the results.

MR1: Amplify query specificity

- **Original Input:** Query: "autonomous driving cars"
- **Transformed Input:** Amplify specificity by adding details: "autonomous driving cars with LIDAR sensors"
- **Expected Outcome:** The results should be more specific, aligning closely with the additional term "LIDAR sensors." The search engine should now prioritize results related to LIDAR sensors in autonomous cars, compared to the general query.

(a) An example snippet of the derived MRs for the search engine



2. MRP_MarginalEnsembleRevision

Objective: Introduce minor changes in non-influential factors to verify the consistency of results.

MR3: Small Change in Email Subject

- *Scenario*: The user sends an email with the subject line "Meeting at 10 AM".
- *Modification (Marginal Revision)*: The subject is changed to "Meeting at 10 A.M." (minor punctuation revision).
- *Expected Behavior*: The email should still be sent successfully, and the system should not treat the revised subject as significantly different in functionality.
- *Rationale*: Ensures that minor textual changes do not affect the ability to send or classify the email.

MR4: Slight Change in Recipient Order

- *Scenario*: The user sends an email to three recipients: `alice@example.com, bob@example.com, charlie@example.com`.
- *Modification (Marginal Revision)*: The order of recipients is changed to `charlie@example.com, alice@example.com, bob@example.com`.
- *Expected Behavior*: The email should still be sent to all recipients without any change in outcome.
- *Rationale*: Verifies that the order of recipients does not affect delivery, ensuring consistent functionality.

(b) An example snippet of the derived MRs for an email app

Figure 5.27: Example MRs derived for other domains

Table 5.9: Evaluation of SUT types across different metrics

Metrics	**Search Engine**	**Navigation Software**	**Email App**	**Ordering System**
Correctness	4.1	3.6	3.8	3.6
Applicability	4.3	4.3	3.8	4.2
Novelty	2.4	2.8	2.1	2.6
Clarity	3.7	4.0	3.5	3.8
Relevance to SUT	4.4	4.7	4.3	3.7
Overall Usefulness	3.3	3.8	3.0	2.8
Computational Feasibility	4.6	4.0	4.4	4.6
Average	3.8	3.9	3.6	3.6

by the GPT-MR generator. Experiments have demonstrated that this generator can produce MRs across multiple domains. However, the results showed limited variation and innovation. This indicates that while the GPT-MR generator has improved the MR generation, there is still potential for further improvements in order to boost the diversity and creativity of its responses.

Furthermore, although the GPT-MR Generator could effectively generate MRs for AD-simulator testing, the use of generative AI in safety-critical applications still faces challenges [155]. For instance, the tool may be sensitive to input variations, and could also carry retain biases that exist in the training data. To address these concerns, the framework proposed includes expert review to ensure that the tool not only enhances efficiency, but also adheres to the safety standards required in ADS testing.

Therefore, to maximize the capabilities of the GPT-MR Generator, users are advised to be familiar with MT and the SUT before using the MR generator. While the model can produce MRs with simple prompts, enriching the details of the requests would make the results more accurate and useful.

Table 5.10: Similar MRs derived by the model and those in the literature

SUTs	MRs derived by the model	MRs in the literature
Search Engine	**MR: Slight reorder of query terms** **Source Input:** Query: "electric car prices" **Relation:** Reorder query: "car electric prices". The search results should remain unchanged, as the reordering does not alter the intended meaning of the query.	**MR: SwapJD** [256] **Description:** The source query A contains only two words (without quotation marks) and the follow-up query B is constructed by swapping the two words. A stable search engine should return similar results for A and B if these two queries have similar meanings, regardless of their word orders.
Navigation Software	**MR: Minor Position Shift** **Source Input:** The navigation software generates a route from Point A to Point B. **Relation:** Slightly shift the starting point by a few meters (e.g., 5 meters in any direction). The system should still generate the same route, as this minor change should not influence the overall route. This MR verifies that the system can handle small GPS inaccuracies without deviating from the expected route.	**MR: MRSimilar** [30] **Description:** Slight variations in starting or ending points should not result in drastically different route recommendations.

5.3.6.4 Future Work

In terms of future research, several directions present promising opportunities to build upon the current study's findings. Firstly, the MRs derived could be extended to evaluate a broader range of functionalities within the CARLA simulator, thereby providing a more comprehensive evaluation of its overall reliability and performance. Furthermore, a systematic quantitative analysis comparing GPT-generated and human-derived MRs can be conducted, using metrics like fault detection rates and coverage improvement along with methods for identifying biases.

Table 5.11: Example of valuable MRs from SUTs not tested with MT

SUT	MR Description
Email App	**MR: Small Change in Email Subject** **Scenario:** The user sends an email with the subject line "Meeting at 10 AM". **Modification (Marginal Revision):** The subject is changed to "Meeting at 10 A.M." (minor punctuation revision). **Expected Behaviour:** The email should still be sent successfully, and the system should not treat the revised subject as significantly different in functionality. **Rationale:** Ensures that minor textual changes do not affect the ability to send or classify the email.
Ordering System	**MR: Amplify Order Quantity** **Original Scenario:** Place an order with 10 units of an item. **Amplified Scenario:** Increase the order to 100 units of the same item. **Expected Outcome:** The system should show an amplified increase in total price and inventory update. If inventory is insufficient, the system should prompt a warning or partial order. **Rationale:** This MR ensures that changes in order quantity lead to proportional or consistent changes in total cost and inventory levels, helping verify if the system appropriately handles large orders.

Beyond utilizing the existing MRPs, there is potential for developing alternative MRPs that are tailored to specific functionalities or complexities of the simulator. These new MRPs could offer a more specialized or streamlined approach to MR generation, optimizing the testing process for distinct use cases. Additionally, these new MRPs could involve patterns of output transformations, an aspect that the current MRPs not possess (they focus on output equivalence relation in the derived MRs), to enhance the diversity of the MRs generated and the capabilities of the GPT-MR Generator.

Thirdly, the performance of the framework can be further explored by applying it to the testing of other SUTs, which will improve its overall effectiveness. Conducting a comprehensive quantitative analysis on how the framework enhances MT efficiency would help reveal areas for improvement. Moreover, as mentioned in Section 5.3.1.1, the configuration used to customize the GPT-MR Generator could also be used to enhance the responses of other LLMs. This would expand the framework's applicability, offering users the flexibility to select their preferred platform. Finally, optimizing the test harness architecture to integrate advanced LLM functionalities, such as automated test scenario generation and automated MR evaluation and validation, can further improve efficiency and reduce human effort.

5.3.7 Conclusion

In conclusion, the human-AI hybrid MT framework has proven to be an effective method for identifying bugs in the CARLA simulator. The customized GPT-MR generator facilitated the generation of MRs and simplified the testing process, making it more user-friendly, especially for users less acquainted with the simulator and MT. The test harness developed further automated this process, enhancing the efficiency of generating and executing test scenarios while allowing for extensive examination of the simulator's diverse functionalities. The issues unearthed through the MT underscore the importance of the evaluation of various facets of the simulator's functionality to ensure its reliability and real-world applicability. While there are acknowledged threats to validity, the insights gained from this study are valuable, highlighting both the strengths and limitations of employing MT, enhanced by MRPs, for evaluating complex systems. Future research work might include the development of additional or alternative MRPs, refinement of existing MRs, and the expansion of the framework, all aimed at achieving an even more comprehensive and effective issue-detection process.

Chapter 6

Promoting Metamorphic Testing through Education

Publications related to this chapter

1. **Y. Zhang**, D. Towey, M. Pike, Z. Q. Zhou, and T.Y. Chen, "Enhancing ADS testing: An Open Educational Resource for metamorphic testing", in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, Osaka, Japan, 2024, pp. 138-147, doi: 10.1109/COMPSAC61105.2024.00029.

2. **Y. Zhang**, D. Towey and M. Pike, "Enabling effective metamorphic-relation generation by novice testers: A pilot study", in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, Osaka, Japan, 2024, pp. 2398-2403, doi: 10.1109/COMPSAC61105.2024.00384.

6.1 Introduction

In the previous chapters, various frameworks and guidelines were proposed to enhance MT efficiency and alleviate challenges. However, these approaches still require users to possess fundamental knowledge of MT. Additionally, finding effective methods to educate students, professional software engineers, testers, and end-users about MT has become a critical concern [47]. To promote MT and address *RQ3* of the thesis (*How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?*), an OER[1] was created [251]. It empowers users to easily generate test cases and MRs for ADS functions by providing accessible, comprehensive resources and tools, specifically aimed at reducing the learning and application hurdles of MT for beginners. To enhance the learning experience and help users develop their skills, a notable feature of the OER is the inclusion of a scenario template: This template is designed to guide users through the process of generating test scenarios, a challenging aspect of MT for ADSs. Furthermore, the OER encompasses a specific methodology to simplify and standardize the process of generating MRs. These educational tools are designed to make the learning process more intuitive and structured, enabling both novices and experienced software testers to better understand and apply the concepts. The launch of this OER is an important contribution to both the educational and practical aspects of MT. It not only bridges the knowledge gap but also promotes a deeper understanding of MT principles and their application in real-world scenarios.

Alongside providing tools to enhance MT learning, two studies were conducted to explore the capabilities of novice testers, with little to no prior experience in software testing or ADSs, in generating MRs for a typical ADS function [214]. The initiative was based on the hypothesis that even individuals new to the field could contribute valuable insights when provided with appropriate guidance [127]. The

[1] Link: `https://www.mt-expertsystem.tech/welcome/`

two studies, which include a pilot study with a small number of participants and comparing the MRs with GPT-3.5, followed by a larger study that recruited additional participants and compared MRs with those produced by GPT-4, demonstrate the potential of beginners with appropriate guidance to effectively generate MRs that exhibit similar qualities to those produced by powerful LLMs. Moreover, these studies underscore areas for improvement in the teaching and training for both humans and LLMs.

6.2 Developing an OER

6.2.1 Creating Templates for ADS Scenario Generation

In the context of MT for testing ADSs by using scenarios, a clearly defined scenario format is crucial for the generation and evaluation of MRs [22]. This thesis introduced a scenario template that standardizes the format and elements for scenario creation, which not only simplifies the initial steps in scenario creation but also reduces the difficulty in generating MRs, making it more efficient and understandable for beginners who are unfamiliar with MT and testing ADSs. This is particularly important in complex domains like testing ADSs, where the generation of effective MRs is critical, but can be challenging for beginners [247]. The use of a template can significantly reduce the complexity and variability in describing MRs [177]. The results of using this template, which is described in Section 6.2.6, show that such a structured approach not only facilitates easier and more accurate testing of ADSs but also contributes to the overall progression and adoption of MT techniques [251].

6.2.1.1 Scenario Template Compoenents

This section introduces the components of the scenario template. A scenario is made up of five separate parts: the scenario type; scenario initialization; actor behaviours; specialized information that affects outcomes; and scenario outputs. Below are the explanations for each component:

1. **Scenario Type:** This component describes the types of scenarios employed in the testing framework, which can be classified into two categories: "in the source scenario" and "in the follow-up scenario". The term "in the source scenario" indicates a scenario as the source test case, while "in the follow-up scenario" refers to those scenarios that have been derived from the source. It is important to note that the types of these scenarios are interchangeable: Scenarios defined as follow-up test cases are equally capable of being utilized as source test cases in a different MT practice.

2. **Scenario Initialization:** The initial setup of a scenario in testing involves two steps: initializing the actors involved; and defining the environment (including any unique infrastructure elements related to the ADS functions, such as traffic lights or speed signs). This stage is crucial for establishing the baseline conditions of the scenario. It contains in-depth information about the actors, such as the ego vehicle and other entities with which it might interact. Additionally, the environmental setup is specified, incorporating features that might impact the scenario, such as parking areas equipped with poles. This setup ensures that all relevant variables are taken into consideration within the testing environment.

3. **Actor Behaviour Definition:** This component describes the initial actions of the actors, such as the ego vehicle attempting to park. This description completes the initialization of the actors by introducing the initial state and activities of the key participants.

4. **Special Information That Affects Outcomes:** The "special information that affects behaviours" component specifies the particular conditions or events in the scenario that impact the SUT's outputs. It highlights the reason-and-cause logic within the scenario. For example, consider a situation where an ego vehicle is navigating a roadway. In this context, an unforeseen obstacle that necessitates the vehicle to initiate emergency braking is the "special information affecting behaviour" component within this scenario. This knowledge helps testers generate MRs and follow-up scenarios, as described in detail in Section 6.2.1.3.

5. **Scenario Outputs:** This final component specifies the results of the scenario, usually focusing on the aspects of the ego vehicle, such as the behaviours or sensor values.

6.2.1.2 Example of a Scenario Generated by Following the Template

An example of a scenario that follows this template is as follows:

> In the source scenario (***Scenario Type***), spawn the ego vehicle behind a front vehicle V1 on a straight road (***Scenario Initialization***). Both vehicles drive along the road at a constant speed (***Actor behaviours***). When the front vehicle V1 suddenly reduces the speed (***Special Information that Affects Outcomes***), the ego vehicle triggers emergency braking and stops before V1 (***Scenario Outputs***).

6.2.1.3 Enhancing MR Generation Through Scenario Templates

The author proposed a methodology on how to use the scenarios created with the template to generate those MRs that can be split into input relations and

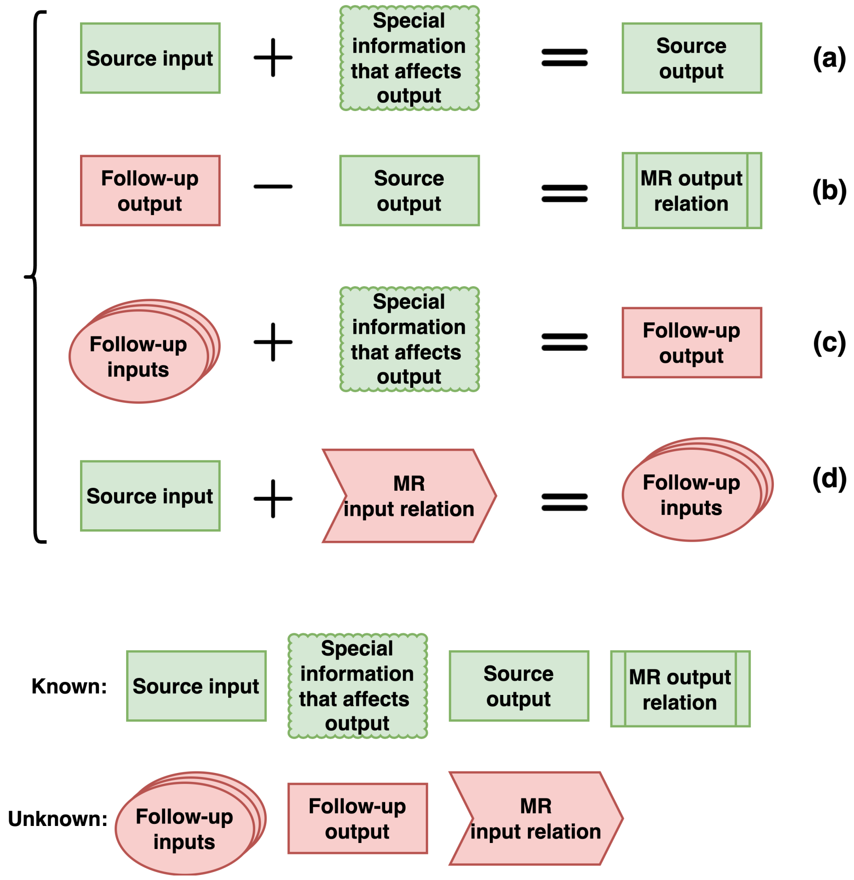


Figure 6.1: Abstractions of the relationships among inputs and outputs of the scenarios and MRs

output relations [176]. An input relation is used to generate follow-up test cases from the source test case, and an output relation is used to evaluate the system's correctness by comparing it against the outputs. The MR output relation is a user-defined rule that specifies the expected relationship between the source output and follow-up output. This is based on the understanding of the system's behaviour.

Figure 6.1 shows the abstractions of the relationships among inputs and outputs of the scenarios and MRs. The source input contains the scenario initialization and actor behaviours defined in the template. Since the scenario type only indicates the scenario as a source or follow-up, it is not included in the equations. Therefore, **equation (a)** can be seen as the inner logic of a source scenario, where the source input plus special information that affects outcomes will lead to the source output (e.g., cause the ego vehicle to take certain actions). **Equation (b)** specifies a typical MR validation process, where the follow-up output and the source output are compared against the MR output relation. If the equation holds, the MR is not violated, and vice versa. However, since the MR output relation is human-defined, testers can combine it with the source output to derive the expected follow-up output. For instance, the MR output relation can be defined as the follow-up output has the same value as the source output. This is a typical MR output relation that has been proven effective in many studies [176][250].

Once the follow-up output is derived, the follow-up inputs can be derived through **equation (c)**. The rationale behind **equation (c)** is that the source and follow-up scenarios should focus on testing the same aspect of the SUT, such that the scenario outputs can be comparable and meaningful for MR validation. However, the “special information that affects output” in **equations (a) and (c)** are not necessarily the same, as long as they have the same impact on the output. For instance, it can be “a pedestrian suddenly appears in front of the ego vehicle” that makes the ego vehicle trigger emergency braking in the source scenario, and “a cyclist suddenly appears in front of the ego vehicle” that makes the ego vehicle trigger the same emergency braking in the follow-up scenario—the only difference being changing the pedestrian to a cyclist. The follow-up inputs are derived by retaining such details and generating variations on the source input. There are many kinds of variations, with each kind linked to unique MR input relations. Once the follow-up inputs are derived, the MR input relations can be determined by applying **equation (d)**, and then the MR can be confirmed by combining the input and output relations.

In summary, the MR generation process can be described as follows: It begins with the identification of metamorphic output relations. This allows the tester to derive the follow-up output based on the known source output. Then the tester uses their understanding of the “special information that affects behaviours” to derive the follow-up input. This step is similar to reverse engineering [34], where the effect is known and the cause is sought. Finally, by comparing the source input with the new follow-up input, the tester can determine the metamorphic input relations.

An example of such an MR generation process could be: assume a source scenario has been set to be the one defined in Section 6.2.1.2. Then the MR output relation is set to be equal, meaning the follow-up scenario outputs remain the same as the source scenario output (i.e., "the ego vehicle triggers the emergency braking and stops before the front vehicle V1"). The special information that affects the ego vehicle's behaviour is the event that the front vehicle suddenly stops. To derive the follow-up inputs, variations need to be generated from the source inputs. For instance, it could be changing the initial speed of the front vehicle to be less than in the source scenario. Then, by comparing and summarising the changes in the inputs of the scenario, the MR input relations can be determined: For instance, in this case, it is reducing the initial speed of the front vehicle. Finally, the MR can be derived as follows:

> $\boldsymbol{MR_{ReduceSpeed}}$:
> In the source scenario, spawn the ego vehicle behind a front vehicle V1 on a straight road. Both vehicles drive along the road at a constant speed. When the front vehicle V1 suddenly stops, the ego vehicle triggers emergency braking and stops before V1.
> In the follow-up scenarios, reducing the initial speed of the front vehicle V1. Keep the other conditions unchanged. When the front vehicle V1 suddenly stops, the ego vehicle triggers the same emergency braking and stops before V1, as in the source scenario.

Differences to the MR Generation Methodology with MRPs/MRIPs in Section 5.2.1.4

Section 5.2.1.4 describes a methodology for generating MRs using MRPs and

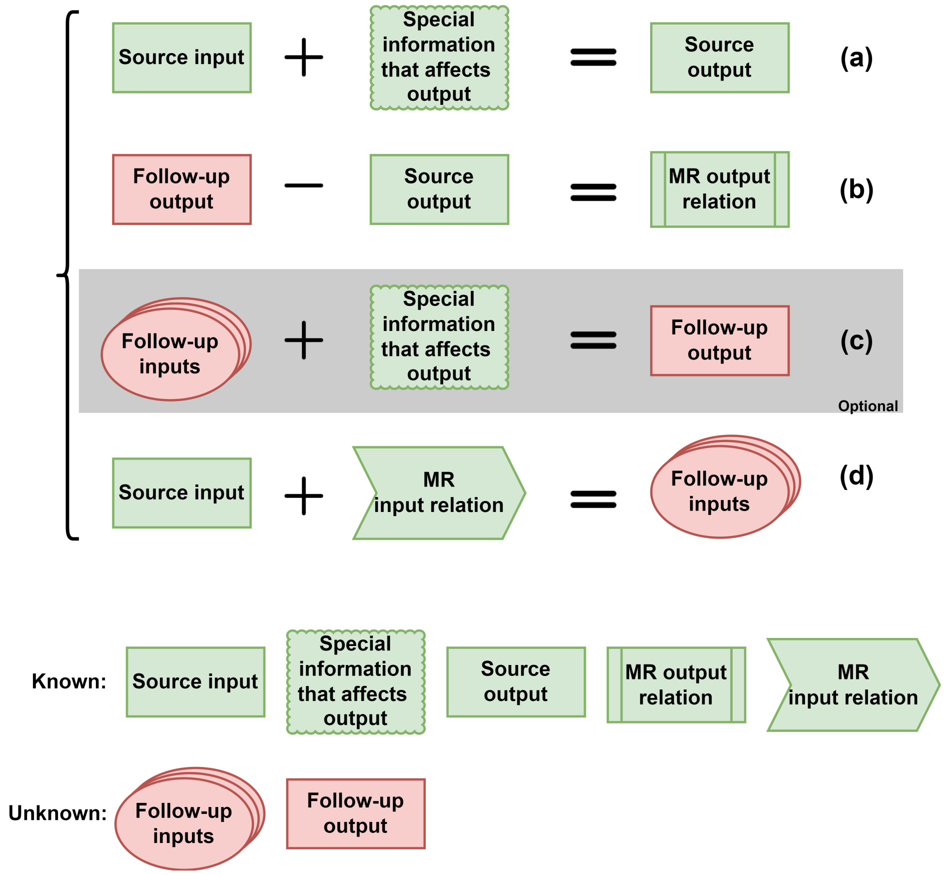


Figure 6.2: Updated MR generation methodology from the original ones (Figure 6.1)

MRIPs. Unlike the previous methodology, the approach presented here is more comprehensive. It includes four equations that describe the relationships between the MR and test scenarios, detailing the logic flow for generating MRs without necessarily requiring prior knowledge about MRPs.

A comparison of the steps in both methodologies shows that both start by setting up the initial source scenarios. Therefore, the previous approach can complement the methodology presented here: Specifically, since the MRP includes both the input and output relations of MRs, users only need to determine the follow-up input and output (Figure 6.2) to construct the complete MRs. Consequently, Equation (c) from Figure 6.1 becomes optional in this case.

Figure 6.2 presents the updated methodology that integrates the approach discussed in Section 5.2.1.4. For instance, according to $\text{MRP}_{\text{ImpactAmplification}}$ (Section 5.2.1.1), the follow-up outputs of scenarios are intensified based on the outputs of the source scenario. Consequently, these follow-up outputs can be derived from Equation (b), where the ego vehicle still activates the AEB function but with earlier timing. Referring to Equation (d), since the input relation in the MRP involves amplifying influential input factors in the source scenario (e.g., the pedestrian's starting time causing the ego vehicle to activate the AEB function), the follow-up inputs can be established by adjusting the pedestrian's starting time earlier than in the source input. If the user is unaware of the "influential input factors" in the source scenario, they can refer to Equation (a) to identify which elements impact the ego vehicle's behaviour in the scenario outputs. By following this process, the same MR introduced in Section 5.2.1.4 can also be reproduced.

6.2.1.4 Comparing the Scenario Template with the ASAM OpenSCENARIO Standard

ASAM OpenSCENARIO [165] is a standardized file format used for defining driving and traffic simulations. It is primarily aimed at the automotive industry, providing a structured way to describe both simple and complex driving manoeuvres and traffic scenarios.

In the OpenSCENARIO definition [13], the complete scenario description is represented by the "storyboard", which encompasses the *who*, *what*, and *when* aspects. It starts with an "Init" element that sets the initial conditions of the scenario, such as actor positions and speeds. Optional "Story" elements group various scenario components and structure larger scenarios, and the scenario concludes with a "stopTrigger" element. Within this framework, "Story" instances contain "Acts" that define when actions occur, regulated by start and stop triggers. "Acts" include "ManoeuverGroup" elements specifying who performs actions, while "Maneuvers" detail what happens, organizing events and actions under common conditions.

```xml
<Storyboard>
    <Init>
        <Actions>
            <Private entityRef = "Ego">
                <PrivateAction>
                    <!-- Set Ego to its initial position -->
                    <TeleportAction>
                        <Position>
                            <WorldPosition x = "-2.51"
                                           y = "-115.75"
                                           z = "0"
                                           h = "1.57"
                                           p = "0"
                                           r = "0" />
                        </Position>
                    </TeleportAction>
                </PrivateAction>
                ...
                <!-- Similar actions -->
            </Private>
        </Actions>
   </Init>
    ...
</Storyboard>
```

Figure 6.3: Excerpt from an XML script showcasing the 'Init' element in ASAM OpenSCENARIO examples [13]

However, scenarios in OpenSCENARIO are defined using a structured, XML-based language, instead of natural language. XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable [111]. For instance, Figure 6.3 displays a segment of an XML script, illustrating the 'Init' element within the ASAM OpenSCENARIO examples. When creating a scenario in ASAM OpenSCENARIO, each aspect of the scenario is represented as an XML element or attribute. Simulation tools designed to work with ASAM OpenSCENARIO can read and interpret these XML files. Once the XML file is loaded into the simulation tool, it executes the scenario as defined by the XML script. The tool translates the XML instructions into dynamic actions within the simulation environment, like moving vehicles, changing traffic lights, or altering weather conditions.

Compared with the scenario template proposed in this section, the ASAM OpenSCENARIO format shares some common types of components, such as initialization and actions. The main differences between the two are related to the usage of the scenarios and certain scenario elements. The proposed scenario template emphasizes a narrative, text-based approach, aiming to enhance its accessibility and comprehensibility for beginners, while XML code can be challenging to read and understand due to its structured format and syntax rules [111]. The scenario template is crafted to describe scenarios in a natural language format, which is inherently more intuitive and user-friendly. This design choice is particularly beneficial for constructing MRs, as it includes specialized components like "special information that affects outcomes". This feature is important for explaining the cause-and-effect logic intrinsic to each scenario, thereby streamlining the MR generation process. In contrast, the ASAM OpenSCENARIO standard [13] uses a standardized, machine-readable format. While effective for developing ADSs, it is more focused on enhancing the efficiency of the simulation tools instead of clarity in reading and comprehending.

6.2.2 Building the OER with Django Framework

In the development of the OER, Django [171], a web framework, was used to enhance user experience. It primarily handled backend operations such as data management and server-side logic [195], which are essential for a dynamic online educational platform [180]. Examples of such successful implementations include a resource-sharing website [182] and a content repository for an education project [185].

The user interfaces of the OER were created to provide learners with an experience in navigating and interacting with the educational material. A key aspect of this was Django's content management system [171], which allowed for dynamic content updates and efficient maintenance of the website. This capability ensured that the content was delivered seamlessly to the front end, resulting in a user-friendly and engaging learning experience.

The development of the backend was primarily focused on smoothly handling user requests and interactions. This was achieved through a structure designed for processing actions such as submitting answers and accessing educational content. The requirement for a comprehensive and scalable architecture, necessary for managing data flow and ensuring responsive user interactions on the website, served as the primary factor in the choice of Django [195].

6.2.3 Principles Guiding OER Design

The design of the OER followed the principles below:

- **Universal accessibility and ease of use**: The OER focused on universal design to ensure a consistent user experience across various devices and to accommodate diverse learners [55]. To achieve this, the user interface was designed to be straightforward and easy to navigate, eliminating potential barriers to learning and engaging. This includes implementing features such as clear and consistent navigation menus, text with high contrast for better readability, and the inclusion of alternative text for images to support learners with visual impairments. Furthermore, the content is structured to support different learning styles, incorporating multimedia elements like videos and interactive quizzes, along with traditional text-based materials [55].

- **Responsive design for device diversity**: Considering the diversity of devices used in accessing online resources, the OER was developed as a website. This decision was influenced by several key factors. Firstly, using a website ensures that users do not need to install any additional applications, providing immediate and hassle-free access to the educational content. This is particularly advantageous for learners who may have constrained resources (e.g., limited storage space). Secondly, developing a web-based

OER reduces the complexity of development and maintenance. Unlike applications that may require separate versions for different operating systems (iOS, Android, etc.), a single website can cater to all users, regardless of their device type. This approach significantly simplifies the development process and allows for more efficient updates and bug fixes. Thirdly, a web-based platform ensures consistency in content appearance and functionality across different devices and browsers [258]. Using responsive web design techniques, the OER automatically adjusts its layout and content to fit the screen size and resolution of any device, from desktop computers to smartphones. This consistency is essential for maintaining the quality and effectiveness of the educational experience.

- **Clarity and focus on educational content**: The visual design of the OER was organized to emphasize the educational content. A clean and uncluttered layout was employed to enhance content readability and learner focus, with careful consideration given to the choice of typography, colour schemes, and spacing.

6.2.4 Licensing

To align with the OER licensing guidelines [179], the OER uses an Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license. The license was chosen for this OER to comply with OER licensing standards and the principles of Creative Commons. This license allows educational content to be freely used, adapted, and shared. For instance, a teacher could adapt and redistribute the material for their classroom without legal barriers, ensuring wide accessibility while protecting creators' rights [54].

This license encourages a collaborative environment for continuous development and sharing of educational resources, contributing to a self-sustaining community.

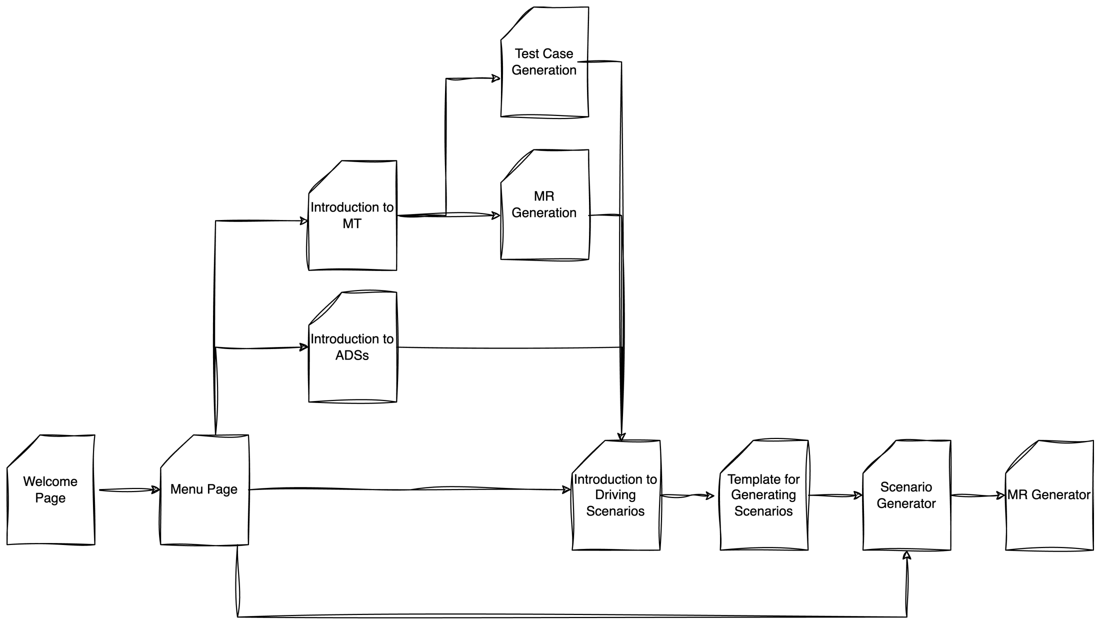


Figure 6.4: The flow of the OER

Under this license, the works provided on this OER can be widely used and adapted, while also contributing to a growing community of shared educational resources.

Below is the licensing information that appears on the OER:

> This website © 2024 by Yifan Zhang is licenced under Attribution-ShareAlike 4.0 International

6.2.5 Implementing the OER: Features and Functionality

The flow of the OER is presented in Figure 6.4: It starts with a Welcome Page that directs users to a Menu Page, acting as a gateway to different features. Users can learn about MT or ADSs through dedicated introductory pages, which provide explanations on these concepts.

Upon selecting either MT or ADSs from the Menu Page, users are guided to the respective sections that introduce these concepts. The MT section provides a comprehensive introduction to MT. It outlines what MT is, elaborates on MRs,

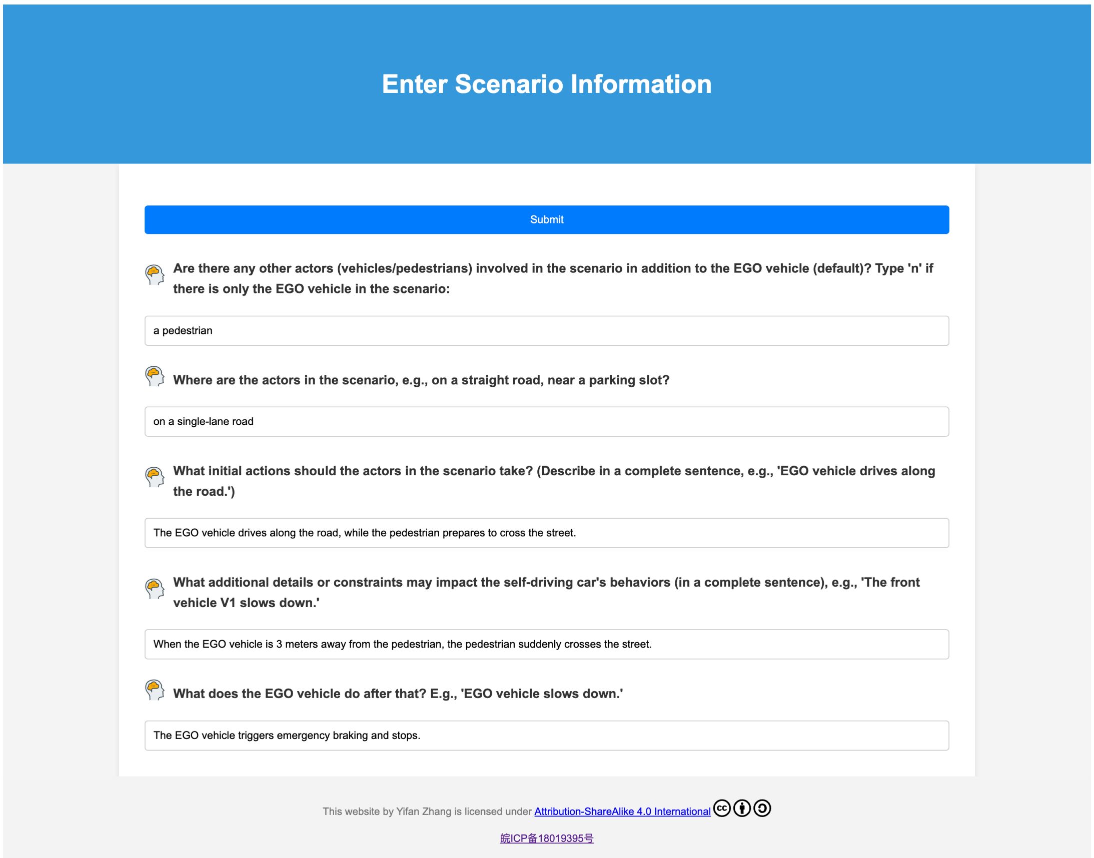


Figure 6.5: Questions of the Scenario Generator that encourage users to think like experienced MT practitioners

and explains how these are used in generating test cases. It also outlines the basic steps for performing MT. The ADSs section, on the other hand, caters to beginners with an introduction to self-driving technologies, providing insights into the features of these systems. It highlights popular functions that contribute to the autonomy of these systems, such as obstacle detection, lane keeping, and adaptive cruise control.

After introducing the theoretical groundwork of MT and ADSs, users are guided to a template for generating scenarios, which offers a structured approach to create diverse testing scenarios reflecting real-world complexities. The template, as detailed in Section 6.2.1.1, is presented in a visually appealing, colourful, and user-friendly format, enhancing the learning experience by making complex concepts more accessible and engaging.

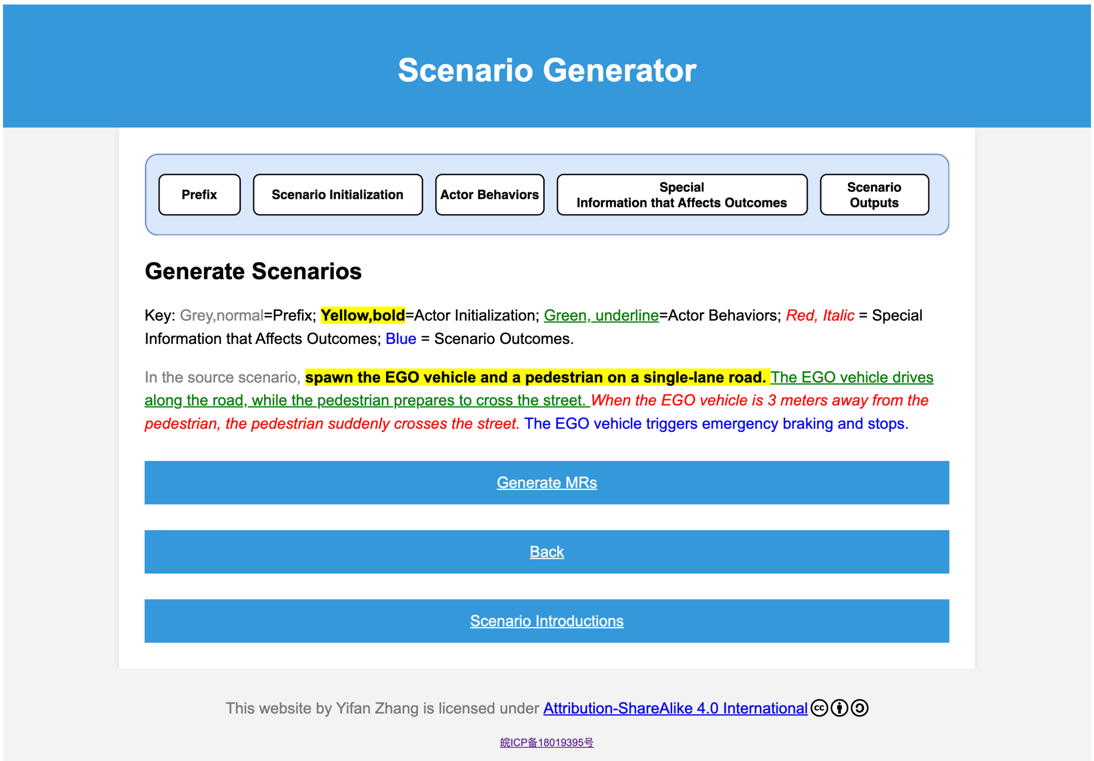


Figure 6.6: Example generated scenario

An important element in the OER is the Scenario Generator. It provides a form to users so that they can define and customize their scenarios based on the template. Figure 6.5 shows the interface of the Scenario Generator: It begins by asking whether any additional actors, such as vehicles or pedestrians, are involved in the scenario, allowing users to specify other entities interacting with the ego vehicle. Next, users are prompted to indicate the location of these actors, such as on a straight road or near a parking slot. This helps to specify the environment in which the scenario takes place. The form also requires users to define the initial actions the actors should take, which is important to establish a clear starting point for the scenario, like "the ego vehicle drives along the road". Further, users can specify additional details or constraints that might impact the ego vehicle's behaviour, such as "the front vehicle slows down", to introduce dynamic conditions. Finally, users need to describe the actions the ego vehicle takes in response to these conditions, such as "the ego vehicle slows down", to determine the expected behaviour of the ADS.

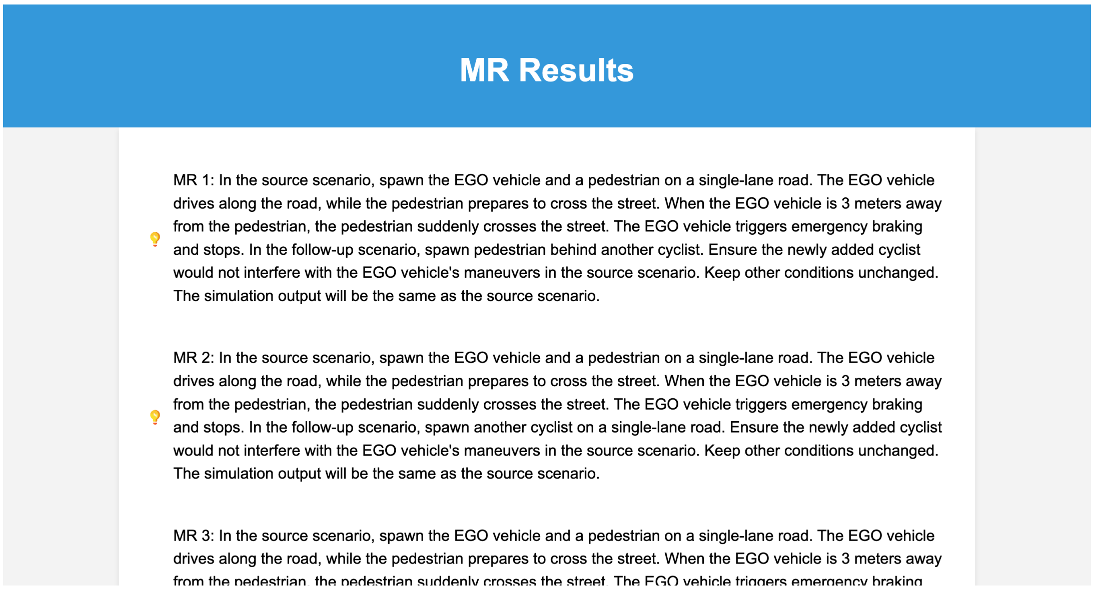


Figure 6.7: Example MR results

After users define their scenarios, the OER organizes the user answers into the complete scenario definition, following the template defined in Section 6.2.1.1. Figure 6.6 shows the scenario generated after the user has submitted answers. It uses a colour-coding system to differentiate various components of a test scenario, enhancing the clarity and comprehension of the narrative. This method of colour differentiation aids users in quickly identifying the critical sections of the scenario. For instance, the yellow bold highlights the starting positions and states of the actors, green underlines indicate their actions, and red italics draw attention to special events that affect the behaviours of SUT, leading to the outcomes in blue. This visual strategy not only simplifies the process of reviewing the scenario but is also especially beneficial when generating MRs that require careful consideration of how input variations influence the outcome. Located at the bottom of the webpage are three interactive buttons. The first button allows for the generation of example MRs directly tailored to the scenarios input by the user, designed to follow the guidelines in Section 6.2.1.3. For those wishing to review or modify their inputs, the second button redirects back to the scenario entry page. Lastly, for additional guidance or to revisit the introduction of the concepts, the third button links directly to the scenario introduction page.

Table 6.1: Survey questions for assessing the OER

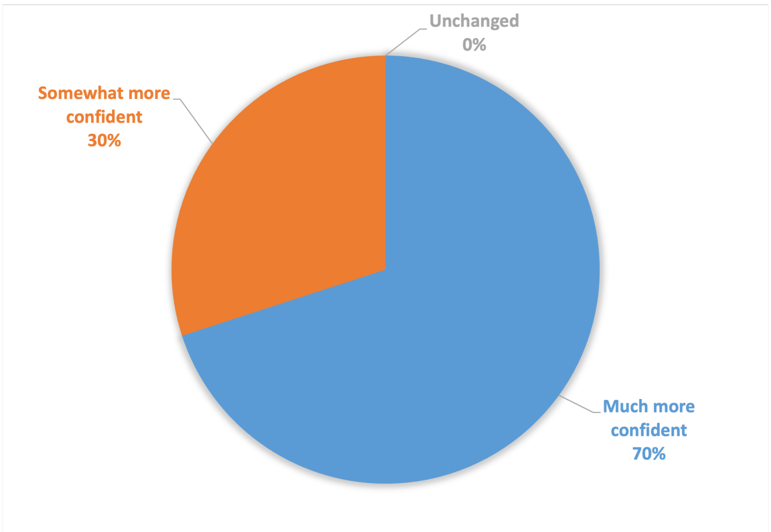
Section	Q. No.	Question Content	Response Options
Content and Usability	1.1	How would you rate the relevance and clarity of the OER content for learning about metamorphic relations and test scenarios?	Excellent, Good, Fair, Poor
	1.2	How easy was it to navigate and use the OER?	Very easy, Somewhat easy, Neutral, Somewhat difficult
Learning Outcome	2.1	After using the OER, how confident do you feel in generating metamorphic relations and test scenarios for autonomous driving systems?	Much more confident, Somewhat more confident, Unchanged
Overall Satisfaction	3.1	Overall, how satisfied are you with the OER?	Very satisfied, Somewhat satisfied, Neutral, Somewhat dissatisfied
Additional Comments	4.1	Please provide any suggestions for improving this OER.	Open-ended response

Once users choose the “Generate MRs” button, the OER calls a program on the server to process the defined scenarios, and generate example MRs. Figure 6.7 shows a screenshot of the MRs generated by processing the scenario in Figure 6.6. Users can choose to save the MR results into text files or go back to other pages to explore other features.

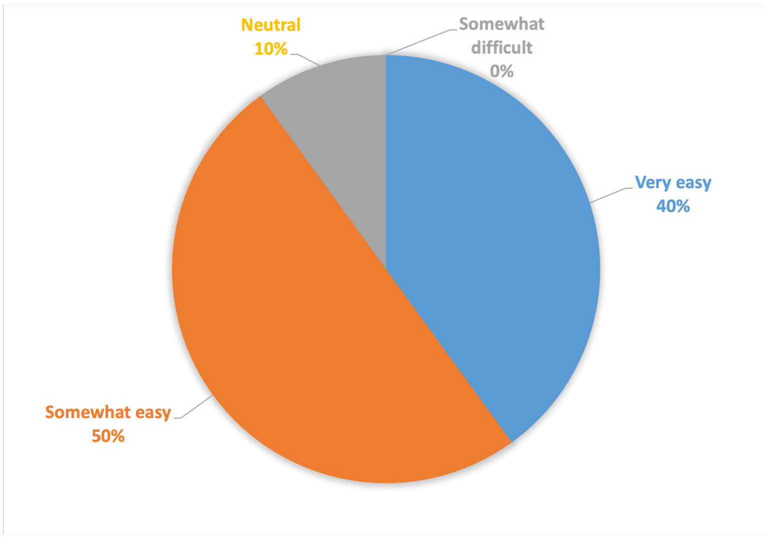
6.2.6 Evaluating the OER’s Effectiveness

In a recent application of the OER, a group of ten testers, primarily students and beginners to MT, were invited to try the OER. The participants were provided with a survey (Table 6.1) to record their experience. The survey asked the participants to assess the OER in three dimensions: content and usability; learning outcome; and overall satisfaction. The key findings are summarized in Figure 6.8.

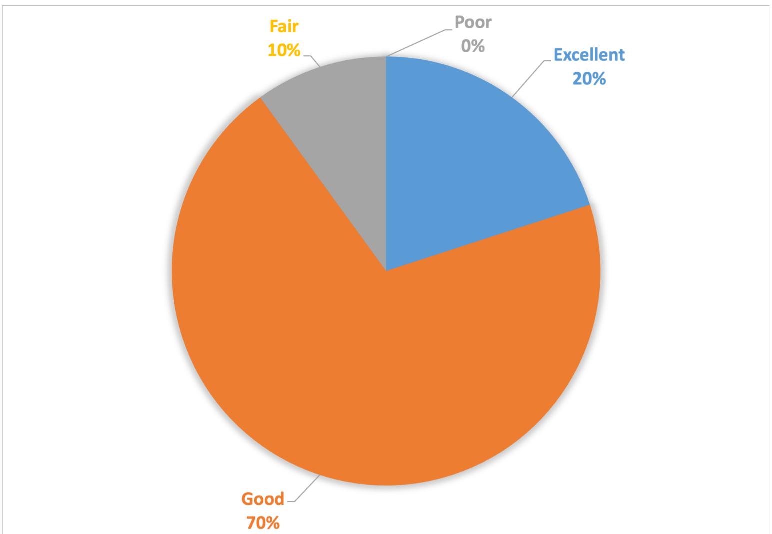
For the first part, Figure 6.8a shows that during a series of testing exercises, the participants reported a notable improvement in their confidence to generate test cases and identify relevant MRs. These beginners, unfamiliar with creating robust test cases, found the narrative, text-based approach of the scenario template particularly beneficial. For instance, by prompting them to articulate “special information that affects outcomes”, the template guided the testers to think crit-



(a) The confidence of users in generating MRs and test scenarios



(b) The ease of navigation and use of the OER



(c) The relevance and clarity of the OER content

Figure 6.8: The findings of the survey

ically about the cause-and-effect relationships inherent in their scenarios. This process not only aided in generating more precise MRs but also in facilitating a deeper comprehension of the SUT.

For the second part (Figure 6.8b), most participants (90%) thought the OER was easy to use and navigate. The flow of the contents allows them to learn the knowledge in a step-by-step manner. Moreover, as shown in Figure 6.8c, the clarity and relevance of the OER contents helped them to reduce the ambiguity during the learning process. As a result, these users were able to construct detailed and concrete test cases, where the results were confirmed by an experienced MT tester with several years of expertise in both MT and ADSs.

Feedback gathered from this group indicated a significant enhancement in their confidence and ability to apply MT techniques. The OER's approach, combining theories and applications, proved instrumental in bridging the gap between theoretical understanding and practical application for these beginners in MT.

In summary, the assessment of the OER with students demonstrated its effectiveness in enhancing users' confidence and ability to generate MRs and test scenarios for ADSs. Given the successful applications of LLMs in MR generation and evaluation (the GPT-MR generator in Section 5.3.1.1 and the GPT-MR evaluator in Section 4.3.3.2), and many existing products offer APIs[2] for integration with other applications, this OER can be enhanced by connecting to these customized models. It would enable users to generate and evaluate MRs more efficiently, thereby enhancing the educational impact of the website.

[2] An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and interact with each other, enabling them to exchange data or functionality [26].

6.3 Enabling Effective MR Generation by Students with Limited Training

In addition to providing learning tools for MT, two studies were conducted on educating students, with little to no prior experience in software testing or ADSs, to generate MRs for a typical ADS function. The OER, alongside a short lecture session, has been provided to students to explore their potential of generating useful MRs within a limited timeframe. The generated MRs were evaluated by several MT experts, and also compared with the ChatGPT-produced ones in Chapter 4. The findings from that chapter indicated that while ChatGPT can create effective MRs, there is still potential for improvement in their correctness and innovation. Similar results were also anticipated for the students, as a prior study [127] demonstrated that inexperienced testers with a small amount of training can identify a sufficient number of MRs. These comparisons would help highlight the strengths and weaknesses in the training of both human testers and AI models, contributing to enhancing MT education.

6.3.1 Conducting a Pilot Study

In this section, a pilot study [248] is presented, which focuses on the potential of novice testers to generate MRs for a critical ADS function: automated parking [214]. The investigation centres on assessing the MRs with the evaluation framework [247] proposed in Section 4.2.1 ($Criteria_{V0}$). Given the limited exposure of participants to MR generation and ADS functionalities, this study also evaluates the educational strategies employed, focusing on their ability to teach essential skills and foster confidence among novice testers. Furthermore, the students-generated MRs were compared to those generated by ChatGPT (GPT-3.5) in Section 4.2, which also assessed MRs for the automated parking function using $Criteria_{V0}$.

6.3.1.1 Recruitment of Participants

This pilot study recruited four male and two female students via email invitations. They were all novices in software testing, MT, and ADSs. All participants were undergraduate students.

6.3.1.2 Study Design

The study was structured to introduce students to the concept and application of MRs within the context of ADSs. The design consisted of a one-hour session divided into two parts: a forty-minute lecture and a twenty-minute hands-on workshop.

The initial lecture aimed to equip students with the necessary knowledge to understand and generate MRs and scenarios related to ADS functions. It started with a broad overview of ADSs using real-life examples, then delved into the challenges of testing these intricate systems, introducing the concept of MT. Following a brief explanation of essential concepts, students were instructed on creating and generating driving scenarios and MRs. The lecture detailed the fundamental elements of driving scenarios and their utilization in the generation of MRs, supported by various examples [175, 246, 250]. Since the lecture only took forty minutes, only the necessary knowledge required to generate MRs for the workshop task was taught, aiming to maximise efficiency.

Following this, the workshop provided a hands-on experience where students were challenged to generate two MRs specifically for an ADS parking function, in which they were provided with a paragraph describing its typical functionality and usage to ensure their understanding of the SUT. Throughout the session, students were encouraged to ask questions related to the lecture content to deepen their understanding. However, to ensure the originality and unbiased nature of the MRs generated, they were restricted from requesting hands-on instructions on tasks or example answers in the workshop. This approach aimed to prevent instructor bias [29] from impacting the quality of the MRs produced.

The choice of the parking function as the focus for MR generation was strategic. The parking function, being a task closely related to the everyday driving experience, was considered to be more accessible and understandable for novice testers compared to more complex ADS functions such as Adaptive Cruise Control[3] (ACC) and Lane Keeping Assist[4] (LKA). Additionally, the relatively closed environment of a parking lot offered a more manageable setting for imagining and designing scenarios.

6.3.1.3 Data Collection

Data collection involved gathering students' generated MRs and their responses on provided answer sheets. These sheets were structured to facilitate organized thought and clarity in the generation of MRs. Each sheet was divided into sections corresponding to the components of an MR, including the source scenario, input relation, follow-up scenario, output relation, and their combinations [255]. The purpose of structuring the answer sheets in this way was to guide the participant's thought process, taking them from comprehending the ADS parking function in a practical setting to conceiving relationships among various scenarios. By dividing the MR generation process into different parts, this structure attempted to reduce the difficulty of MR generation while taking into account the students' diverse academic backgrounds and degrees of skill.

[3]Adaptive Cruise Control (ACC) automatically adjusts a vehicle's speed to maintain a safe distance from the vehicle ahead, enhancing comfort and safety during varying traffic conditions [225].

[4]Lane Keeping Assist (LKA) actively helps to keep the vehicle centred within its lane, providing steering corrections if the vehicle begins to drift unintentionally, thus reducing the risk of lane departure accidents [207].

6.3.1.4 Data Analysis

The analysis of the MRs and scenarios generated by the students employed the $Criteria_{V0}$ (Section 4.2.1). The evaluation was performed by a group of evaluators (primarily the author), who are experts in the area. This pilot study used $Criteria_{V0}$ to compare student-generated MRs with those generated by GPT-3.5 for two main reasons:

1. Despite its limitations, GPT-3.5, available through the free version of ChatGPT, has demonstrated the ability to generate useful MRs [247]. This is comparable to the performance of human beginners who have received only brief instruction because both GPT-3.5 and novice students can produce MRs that are functional but exhibit limitations, potentially due to their limited training and experience (for GPT-3.5, it is the constraints of the model architecture and training data [213]). By comparing the MRs generated by both, the strengths and weaknesses in the capabilities of each one can be identified.

2. Existing analyses of GPT-3.5-generated MRs have used $Criteria|V0$ for assessment. Therefore, to directly compare the student-generated MRs with those generated by GPT-3.5, the same criteria were adopted.

6.3.1.5 Quality of Students-Generated MRs

In the experiment, each of the six participating students was tasked with generating two MRs, resulting in a total of 12 MRs under evaluation. The MRs were assessed using $Criteria_{V0}$, with each criterion rated on a scale of 1 to 5.

Correctness The average score for the correctness criterion was **2.9**, indicating that approximately half of the MRs encountered issues with correctness. The main issues revolved around weak or absent connections between the source and follow-up scenarios, and unclear descriptions of certain elements of driving scenarios, such as the precise actions of the ego vehicle during events. For instance, the following is an example of the human-generated MR which has some problems:

> *"In the source scenario, at a crowded mall parking lot with only two available parking spots (one big and one small). The ego car is small and can fit in both spots. Soon another big car (B) comes in, which can only fit in the big parking spot. And the ego vehicle has not driven into the parking space.*
> *In the follow-up scenario, the ego vehicle had better drive in the smaller spot so another car (B) can fit into the bigger one. The SUT can utilize all the space fully."*

There are two main problems with this MR: First, in the source scenario, the event logic is not clearly shown: The reason why the ego vehicle does not drive into the parking space and how this relates to the parking function is unclear. Second, in the follow-up scenario, the input relation is unclear (i.e., how the follow-up scenario is derived from the source scenario), and the relation between the ego vehicle's behaviour in the source and follow-up scenarios is not specified. Moreover, the description of the ego vehicle's behaviour lacks specificity in demonstrating the system's decision-making processes. Instead, the expressions attribute human-like decision-making to the vehicle (e.g., "the ego vehicle had better drive into the smaller spot"), which diverges from the expected output formats of an ADS, typically characterized by binary or clearly defined actions (e.g., proceed with parking manoeuvre or abort).

Applicability The applicability criterion received an average score of **3.5**, indicating that several MRs demonstrated potential for practical testing applications. One common problem with the generated MRs was that the description closely mirrored existing function specifications, making them more suitable for single-scenario testing [211] rather than emphasizing the relationships among different scenarios. However, a portion of the MRs demonstrated potential for testing applications. An example of MRs in this context involved manipulating the positioning of obstacles around the parking spot. Specifically, it proposed swapping the obstacles' locations on either side of the designated parking space, and the distance between the ego vehicle and the nearby obstacles was similar in both the source and follow-up scenarios.

Novelty The evaluation under the novelty criterion yielded an average score of **2.3**, indicating a general trend toward MRs with limited innovation. This observation aligned with the earlier discussion on applicability, where a portion of MRs were similar to functional specifications, thereby constraining their potential to provide new insights. Despite this trend, a few MRs stood out for their originality, such as an MR exploring the effects of relocating a light source around the parking spot to test the ADS's parking angle consistency. The MR was about placing the light ahead of the target spot in the source scenario and moving the light to the back of the target spot in the follow-up scenario, where the angle of the ego vehicle when it parked into the slot should have been similar in both scenarios. This MR differed from typical obstacle-based tests and was innovative in using the placement of lights to test the angle of the vehicle.

Clarity, Relevance to Safety, and Computational Feasibility These criteria each received an average score of **2.8**. A significant portion of the MRs suffered from ambiguous descriptions and lacked a focus on safety aspects. For instance, below is an example of the MRs generated:

> *"In the source scenario, the ego car is going to drive into the parking lot with a lot of parking spots. In the follow-up scenario, suddenly,*

> *someone has fallen or tripped over a pack of fruit or other commodities left in your parking spot. The ego vehicle stops and waits or drives away to find a new parking spot."*

This MR had shortcomings in three aspects: First, in the source scenario, only the initial behaviour of the ego vehicle was mentioned without detailing the ADS's interaction with the environment or other traffic participants. Second, the follow-up scenario's description of the ego vehicle's action did not concretely outline the ADS's decision in response to the pedestrian hazard. The tester should have specified a single action taken by the vehicle instead of presenting options such as "stops and waits" or "drives away", indicating a misunderstanding of the system. This absence of detail decreased the MR's utility in testing the ADS's safety mechanisms and presented challenges for accurately simulating or implementing the MR in real-world tests. Third, the connection between the behaviours of the ego vehicle in both the source and follow-up scenarios was unclear, making it difficult to determine any violations of MRs during the experiments.

Overall Usefulness The overall usefulness criterion scored an average of **2.6**, reflecting a mixed assessment of the MRs generated by humans. This score reflects the challenges faced in the generation of MRs, with a notable number being either incomplete or inaccurately conceptualized, which consequently limits their practical application in testing environments. Despite these limitations, it is important to acknowledge that within the MRs evaluated, some MRs stood out for their innovative approach and relevance, demonstrating potential value for implementation in testing scenarios.

In summary, the findings indicated that ensuring the correctness of the MRs was difficult for students due to challenges in accurately describing the actions and events of the traffic participants, as well as weak connections between the source and follow-up test scenarios within the same MR. Clarity, safety relevance, and

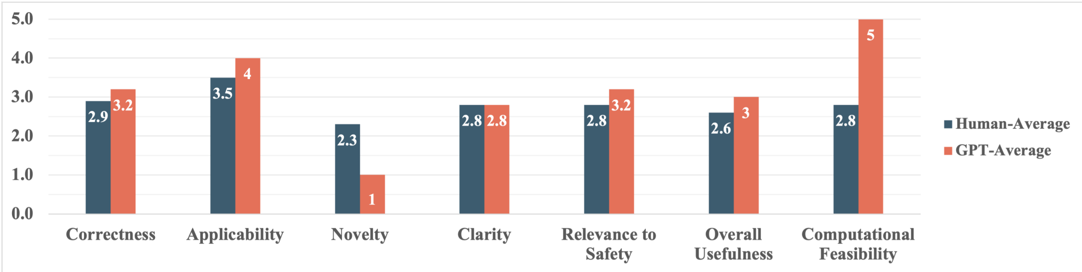


Figure 6.9: Comparison of students-generated and ChatGPT-generated (GPT-3.5) MRs

computational feasibility of the MRs were also areas of weakness, with many MRs failing to provide clear, safety-focused, and realistically testable scenarios. However, some MRs exhibit applicability that could be useful in practical testing. Despite these challenges, some innovative MRs showed promise in uncovering bugs.

6.3.1.6 Comparison of Student-Generated and ChatGPT-Generated MRs

Figure 6.9 presents the comparison results of the MRs in this study with those generated by GPT-3.5 from Section 4.2.

The clarity scores of MRs generated by human beginners and ChatGPT were similar. Both groups produced MRs that were easily understood but closely aligned with function specifications. This could have led to a lack of focus on the logical progression and connection between scenarios. Such connections are essential for MRs to effectively test system responses across different scenarios [47]. In terms of safety relevance, ChatGPT's MRs slightly outperformed those of the humans. This suggests that ChatGPT might be better at incorporating safety considerations into MRs than human novices.

ChatGPT's MRs were better in applicability, computational feasibility, and overall usefulness, possibly due to its extensive training on diverse datasets [222], allowing it to generate more MRs that were well-tuned to existing specifications

Table 6.2: Students' feedback regarding ease of generating MRs

Ease of Generating MRs	**Percentage**
Difficult: Had to overcome some challenges and occasionally sought guidance.	17%
Moderate: encountered occasional challenges but managed to navigate through.	33%
Easy: Found the process fairly straightforward with minimal challenges.	33%
Very Easy: The process was intuitive and faced no significant challenges.	17%

and systems. The slightly better correctness score for ChatGPT could have been because it followed logical and real-world data patterns [222], an area where beginners might struggle without extensive knowledge or experience.

Notably, despite their brief education on the subject, the human participants managed to score higher than ChatGPT in the novelty criterion. This suggests that even at a beginner level, there is a chance for humans to approach problems with an innovative perspective, leading to testing scenarios that an AI may not consider. This is impressive, given the complexity of MR generation and the limited exposure to the subject matter.

The comparative analysis of the MRs indicates that although AI is capable of producing MRs that are more technically precise and relevant, humans demonstrate a remarkable ability for innovation, particularly after receiving only a short introduction to the subject. This highlights the potential for growth and learning among human participants, which could be enhanced through more comprehensive training and learning materials.

6.3.1.7 Student Feedback on Learning and Practice

After generating the MRs, the students were encouraged to take a short self-evaluation survey to report their experience in learning and generating MRs. The self-evaluation surveys assessed students' ease with which they generated

Table 6.3: Students' feedback regarding the confidence of generated MRs

Confidence in Generated MRs	**Percentage**
Low: Slightly unsure about certain aspects of the MRs.	17%
Moderate: Fairly confident, but see areas of potential improvement.	33%
High: Very confident in most parts, with minor uncertainties.	33%
Very High: Extremely confident; certain about the quality and effectiveness of the MRs.	17%

them and their confidence in their own MRs. The ease of MR generation ranged from "Very Difficult" to "Very Easy", indicating the students' perceived difficulty of the task and their need for assistance (Table 6.2). Confidence levels ranged from "Very Low" to "Very High", reflecting students' trust in the quality and applicability of their MRs (Table 6.3). These surveys aimed to evaluate the effectiveness of the instructional methods and the student's understanding of MR generation.

According to the data in the tables, the diversity in the ease of MR generation suggested that while some students found the methodology accessible, there was a notable variation in how different students perceived the complexity of generating MRs. The fact that 17% found the process "Difficult" points to a need for clearer and more detailed instructions that can cater to a range of learning styles and paces, ensuring that all students receive the support they need.

Confidence levels in the MRs produced indicated that while a substantial number of students felt confident in their work, some students expressed reservations, with 17% indicating "Low" confidence. This split underscores the importance of reinforcing confidence-building in MR generation through practice and feedback. It also highlights the potential for introducing peer review or collaborative learning exercises where students can engage with and learn from each other's approaches and thought processes.

The variation in both ease of MR generation and confidence levels indicates that while the workshop's one-hour timeframe was designed to maximize participation by not being overly time-consuming, it may have influenced the depth of understanding for some students. In an empirical study exploring MT effectiveness [127], it was demonstrated that inexperienced students with a longer learning time (three hours) could deliver a sufficient number of appropriate MRs. The decision to use a shorter timeframe in the research was made to attract more participants and replicate situations where time is limited, such as brief guest lectures, since longer sessions might discourage potential attendees. However, this decision may have implications for the thoroughness of the educational experience, suggesting a trade-off between attracting participants and providing extensive instruction. Meanwhile, the quality of MRs and the ease of the generation process are also dependent on the complexity of the SUT. In this study, the ADS function might have been relatively more complex than the SUTs that the students faced in the empirical study [127], which comprised five JAVA math programs.

These self-evaluations indicated that, although the instruction in this pilot study showed effectiveness, the teaching strategies still needed refinement. In the follow-up study, the author improved instructional clarity by focusing on a detailed scenario-generation process and incorporating practical examples, helping students to overcome challenges and produce higher-quality MRs.

6.3.2 Expanding Research with a Follow-up Study

In this follow-up study, a larger number of participants (fourteen students) were recruited, and the teaching materials were updated by addressing the weaknesses of student-generated MRs revealed in the pilot study, while keeping the same teaching time. The qualities of the new MRs generated by students were then compared with those generated by GPT-4, using $Criteria_{V1}$, which was also used to evaluate the qualities of MRs by GPT-4 (Section 4.3.2). This approach aimed to validate the expected improvements in outcomes, based not only on the enhanced training materials developed from the pilot study's findings but also on earlier results indicating that GPT-4 produces more effective MRs than GPT-3.5.

6.3.2.1 Updated Training Materials

In the updated training session, the content has been refined to better teach students how to generate MRs based on insights from the pilot study. The new training materials not only include the standard introduction to ADSs and MT but also put a stronger emphasis on scenario construction and the general structure of MRs for ADSs. This structure includes key elements such as the source scenario, input relation, follow-up scenario, and output relation. The structure integrity of MRs was the weakness of the students-generated MRs in the pilot study. The scenario template and its accompanying MR generation methodology in the OER (Section 6.2.1) were also introduced in this training session.

Furthermore, the training process was enriched with more detailed examples of MRs for ADSs, aiming to provide participants with clearer, practical illustrations. More interactive elements to engage participants actively were incorporated. These activities were designed to facilitate brainstorming sessions about real-life driving scenarios, which could be beneficial for the MR generation process.

To keep the training session within the allocated time frame, the time spent on introducing ADS concepts was decreased. The pilot study revealed that most participants already possess varying degrees of knowledge about driverless vehicles, likely due to the rapid advancements and widespread commercial advertising of similar ADAS systems in recent years (e.g., Tesla's Full Self-Driving (FSD) [14]). This adjustment allows the training to focus more on the practical aspects of MR generation within the time, ensuring that participants can apply what they learn

Table 6.4: Comparative table of the follow-up and pilot study results

Criteria	Follow-Up Study Score	Pilot Study Score (scaled)	Improvement
Completeness	1.0	N/A	N/A
Correctness	2.1	1.74	20.69%
Generalizability	2.3	2.1	9.52%
Novelty	2.0	1.38	44.93%
Clarity	2.3	1.68	36.90%
Computational Feasibility	1.9	1.68	13.10%
Applicability	2.1	1.68	25.00%

more effectively.

6.3.2.2 Comparison of MR Quality with Pilot Study Findings

The follow-up study used $Criteria_{V1}$ for evaluation, which was different to the pilot study (which used $Criteria_{V0}$). To enable a comparison between the results of the follow-up study and the pilot study, the pilot study marks were scaled: The pilot study employed a 5-point evaluation scale for each criterion, whereas the follow-up study used a 3-point scale. Consequently, the pilot study scores were scaled to correspond with those of the follow-up study (given that $Criteria_{V1}$ is more comprehensive than $Criteria_{V0}$). With the updates of the evaluation frameworks, the generalizability criteria scores from the follow-up study were compared to the applicability criteria scores from the pilot study. Meanwhile, the applicability criteria scores of the follow-up study were evaluated in relation to the safety relevance criteria scores of the pilot study. The comparative results of the MRs from both studies are presented in Table 6.4.

The follow-up study on student-generated MRs showed notable improvements across several evaluation criteria compared to the pilot study. The analysis revealed that the **completeness** of MRs scored a full mark (1.0), indicating that all essential components were present in the evaluated MRs. Although $Criteria_{V0}$ does not have this criterion, as mentioned in the findings (Section 6.3.1.5), some of the MRs lacked the necessary components to form an MR, such as the input or the output relations between the source and follow-up scenarios.

In terms of **correctness**, the follow-up study achieved a score of 2.1 out of 3, reflecting an improvement of 0.36 points (20.69%) from the pilot study's scaled score of 1.74. This enhancement can be attributed to the updated training sessions that emphasized the correct formulation of the relations, which were previously identified as weaknesses. By focusing on scenario construction and detailed examples, the training provided students with a clearer understanding of how to ensure the correctness of their MRs. For instance, Table 6.5 displays two MRs created by students from the pilot study and the follow-up study. Both MRs include the scenario of another vehicle parking ahead of the ego vehicle during the auto-parking stage. By comparing the two MRs, it is evident that the MR from the pilot study lacks clear input and output relations between the source and follow-up scenarios: The transition between the inputs of both scenarios is vague. In other words, the "scenario initialization" element (defined in the scenario template of the OER in Section 6.2.1) is missing in the follow-up scenarios, making the follow-up scenario incomplete. Furthermore, the scenario outputs (i.e., ego behaviours) are not compared between the two scenarios. In contrast, the MR from the follow-up study clearly shows the difference between the source and follow-up scenarios while also comparing the scenario outputs with a defined relationship. Although this MR has other areas for improvement, it includes the correct and necessary components of an MR for ADSs.

Generalizability in the follow-up study was rated at 2.3, slightly outperforming the 2.1 score in the pilot study with a 9.5% improvement. The improved generalizability reflects the emphasis on the general structure of MRs for testing ADSs in the training. Practical examples and interactive brainstorming sessions helped

Table 6.5: Correctness comparison between the pilot study and the follow-up study

MR from the pilot study	**MR from the follow-up study**
"At a crowded mall parking lot with only two parking available parking spots (one big and one small). The ego car is small and can fit in both spots. Soon another big car (B) comes in, which can only fit in the big parking spot. And the ego vehicle has not driven into the parking space. In the follow-up scenario, the ego vehicle had better drive in the smaller spot so another car (B) can fit into the bigger one. The SUT can utilize all the space fully."	*"In the source scenario, spawn ego vehicle in a mall parking lot, driving along a parking lane with one open space on the right. Another vehicle is driving towards ethe go vehicle in the same lane. The other vehicle slows down, let the ego vehicle reach the open space first. The ego vehicle turns right, and parks in the open space. In the follow-up scenario, the other vehicle speeds up, and reaches the open space first. ego vehicle should stop, let the other vehicle finish parking, and then search for open space. The follow-up scenario output is the opposite of the source scenario output."*

participants think more broadly about applying their MRs across different SUTs, enhancing the adaptability of their generated MRs.

Novelty gained a significant improvement, scoring 2.0 in the follow-up study, which represents an increase of 0.62 points (44.93%) from the pilot study's scaled score of 1.38. The inclusion of more practical illustrations and interactive brainstorming sessions in the training helped students generate more innovative and unique MRs. The emphasis on real-life driving scenarios and the structure of MRs facilitated the creation of new input/output relations. Section 6.3.2.4 provides a detailed examination of such innovative MRs.

Clarity also improved, with a score of 2.3 in the follow-up study, up by 0.62 points (36.90%) from the pilot's 1.68. The reduction in time spent on introductory ADS concepts allowed the training to concentrate more on MR usage and structures, making the MRs more comprehensible to individuals with basic knowledge of the field. This approach ensured that students could produce

Table 6.6: Clarity comparison between the pilot study and the follow-up study

MR from the pilot study	**MR from the follow-up study**
"*In the source scenario, the ego car is going to drive into the parking lot with a lot of parking spots. In the follow-up scenario, suddenly, someone has fallen done or tripped over a pack of fruit or other commodities left in your parking spot. The ego vehicle stops and waits or drives away to find a new parking spot.*"	"*In the source scenario, the ego vehicle is trying to park in the slot. A human walks before the ego vehicle. The ego vehicle does not stop and continue parking. In the follow-up scenario, the human walk after the ego vehicle. The ego vehicle should stop.*"

clearer and more accessible MRs. Table 6.6 shows a comparison of MRs from the pilot study and the follow-up study. Both MRs describe similar scenarios involving human interaction with the ego vehicle during the parking stage. In the MR from the pilot study, the follow-up scenario's description of the ego vehicle's action/decision is unclear. For example, it presents an ambiguous choice between "stops and waits" and "drives away", which fails to specify a final decision made by the ADS. In contrast, the MR from the follow-up study provides a clearer and more concise description of the scenarios, along with well-defined input and output relations, which also reflects the effectiveness of the updated training materials to let students have a deeper understanding of the ADS mechanisms. This clarity ensures that the MR is easily understandable and accurately represents the intended behaviour of the ADS.

Computational feasibility had a smaller improvement relative to the pilot study, scoring 1.9 compared to the pilot study's scaled score of 1.68, an increase of 0.22 points (13.10%). The updated training emphasized practical aspects involving generating source test cases and MRs, which made these concepts more accessible and easier to implement for the students.

Finally, the **applicability** of the MRs in the follow-up study, when compared to the pilot study's "relevance to safety" criterion, improved by 0.42 points (25.00%), from a scaled score of 1.68 to 2.1. The emphasis on scenario construction and the specific examples used during training, which focused on real-world driving situations and practical applications, ensured that the MRs were more relevant and useful for a larger group of SUTs.

In summary, the follow-up study indicates significant improvements in all areas due to enhanced training methods. Emphasizing practical examples and tackling weak points helped to improve the quality and applicability of the student-generated MRs.

6.3.2.3 Comparison of MR Quality between Human Beginners and GPT-4

As outlined in Section 4.3.4, ChatGPT, especially GPT-4, can produce MRs that are complete, relevant, and generalized for complex systems while there is potential for improvement in originality and accuracy. This aligns with the results from student-generated MRs. Since both the human and ChatGPT produced MRs target the same type of ADS function, the following comparison was conducted on the quality of the MRs between the two, where Table 6.7 displays the scores for human-generated and GPT-generated MRs across each criterion, with the bold value indicating the higher score.

In terms of completeness (1.0), both human- and GPT-4-generated MRs contained all essential components for an MR. The same mark was also found in correctness (2.1), indicating that the input and output relations correspond well with the specifications of the SUT. This comparison suggested that the enhancements in training have raised the accuracy of human-generated MRs to a level similar to those generated by GPT-4, although not all MRs were completely accurate. Clarity was another area where both human- and GPT-4-generated MRs scored equally (2.3). Both sets of MRs were understandable to a broad audience,

Table 6.7: Evaluation comparison table of MRs between students and GPT-4

Evaluation Criteria	**Human MRs (Follow-up Study)**	**GPT-4 MRs**
Completeness	1.0	1.0
Correctness	2.1	2.1
Generalizability	2.3	**2.7**
Novelty	**2.0**	1.8
Clarity	2.3	2.3
Computational Feasibility	1.9	**2.0**
Applicability	2.1	**2.9**
Total Score	13.7	**14.6**

including individuals with a basic understanding and domain experts. This clarity ensured that the MRs could be effectively communicated and used by various stakeholders involved in the testing process.

Notably, the gap in **novelty** between GPT-4 (1.8) and human-generated MRs (2.0) is smaller than in the previous pilot study (1.0 vs 2.3), while the novelty of the MRs in the follow-up study has also increased compared to the previous study. This aligns with the observations found in Section 4.3.1 that GPT-4's ability to generate innovative MRs has greatly improved compared to GPT-3.5, making its MRs almost as novel as those created by human beginners.

Additionally, GPT-4 exhibits obvious strengths in several areas. GPT-4-generated MRs were more **generalizable** (2.7) compared to human-generated MRs (2.3). This suggested that GPT-4 could produce MRs that were applicable to a broader range of SUTs, demonstrating greater adaptability and versatility.

In terms of **computational feasibility**, GPT-4 maintained a slight advantage (2.0) over human-generated MRs (1.9). The results indicated that GPT-4-generated MRs were easier to automate in test case generation and validation processes.

Applicability was another area where GPT-4 outperformed, scoring 2.9 compared to 2.1 for human-generated MRs. GPT-4-generated MRs showed a stronger relevance to the key features of the SUT, indicating a deep understanding and alignment with the unique characteristics and behaviours of the SUT, making these MRs effective for targeted testing scenarios.

In summary, while both human- and GPT-4-generated MRs scored the same in completeness, correctness, and clarity, GPT-4-generated MRs have advantages in generalizability, computational feasibility, and applicability. Although these MRs were less novel than human-generated ones, the gap has significantly narrowed compared to GPT-3.5 in the pilot study. The training improvements have also brought the correctness of human-generated MRs closer to the GPT, compared to the results in the pilot study.

Despite these improvements, GPT still struggles with generating highly creative and original MRs that require deep insights or unique ideas. To address this, the training of LLMs should include a wider range of diverse MRs and their applications across various fields. Future research could also explore techniques and frameworks that blend structured guidance with human feedback to enhance the novelty of the generated MRs. Furthermore, teaching students skills such as prompt engineering and MR selection will help them collaborate effectively with AI to merge human insight with the strengths of LLMs.

6.3.2.4 Unique Features of the Human-Generated MRs

Among the MRs generated in the follow-up study, students demonstrated a remarkable ability to incorporate realistic, everyday scenarios into their MRs, which is a unique advantage that has not been found in AI-generated MRs. This ability reflects their understanding of common driving situations, adding variety to the simulation testing. The personal experiences and observations allowed students to create more natural and comprehensive scenarios that better simulate the unpredictability and variety of real-world driving conditions. It not only enhances the novelty and applicability of the MRs but also ensures that the ADSs are evaluated against a broader range of challenges, which leads to more reliable and safer systems. The following contents highlight some of the key innovative features of the students-generated MRs.

1. **Passenger Interaction**:

   - **Innovative element within the scenario**: Passengers get off from another vehicle next to the parking space.
   - **Rational**: This scenario is rarely seen in typical testing scenarios for parking, which usually focus on the behaviours of the ego vehicle. By including the behaviour of passengers from another vehicle, this MR introduces additional real-world dynamics and safety considerations, requiring the ego vehicle to respond to unpredictable human actions from other traffic participants, thus improving the robustness of the testing.
   - **Example**:

     > *"In the source scenario, spawn the ego vehicle in a mall parking lot, with one open space on the right. ego vehicle is driving along the parking lane. The ego vehicle turns right, and parks in the empty slot. In the follow-up scenarios, keep other conditions unchanged. The passengers in the car right next to the empty space are currently getting off. The ego vehicle should wait till all passengers leave the open space, then turn right, and park in the empty space. The source scenario output is a subset of the follow-up scenarios."*

2. **Snatch Parking Space**

- **Innovative element within the scenario**: The ego vehicle attempts to park, but another car parks in the slot first.
- **Rational**: This element demonstrates competitive and aggressive parking behaviours often seen in real-life situations. Such behaviours are less commonly seen in existing scenarios, which tend to focus on more predictable and cooperative interactions. By introducing these real-world competitive behaviours, human-generated MRs highlight the need for the ADS to handle complex, real-time decision-making and adapt to more adversarial driving conditions.
- **Example**:

  "In the source scenario, spawn ego vehicle in a mall parking lot, driving along a parking lane with one open space on the right. Another vehicle is driving towards the ego vehicle in the same lane. The other vehicle slows down, let the ego vehicle reach the open space first. The ego vehicle turns right, and parks in the open space. In the follow-up scenario, the other vehicle speeds up and reaches the open space first. The ego vehicle should stop, let other vehicles finish parking, and then search for open space. The follow-up scenario output is the opposite of the source scenario output."

3. **Decision-Making in Parking**

- **Innovative element within the scenario**: The ego vehicle decides between a nearby tight parking spot and a distant, spacious one.
- **Rational**: This element highlights the ability of the ego vehicle to evaluate and prioritize parking options, which simulates a common decision-making process drivers face. It adds complexity by requiring the vehicle to balance convenience and practicality, enhancing the MR's depth and applicability.

- **Example**:

  "In the source scenario, there is a tight parking space between two cars. The ego vehicle will detect the tight parking area and steer itself into the space. In the follow-up scenario, there is another parking area. The second parking area is much easier for the ego vehicle to park, because there is nothing around the second place. But the location is far away from the car, such as 10 meters. The ego vehicle should still choose the first parking area it detected."

4. **Dynamic Parking Conditions**

   - **Innovative element within the scenario**: An elevated car parking slot that suddenly moves.
   - **Rational**: This element introduces complexity with moving parts in a parking environment, similar to mechanical failures or adjustments. Elevated car parking slots are less common to find in real-life and testing, making this scenario unique. It tests the ego vehicle's ability to adapt to unexpected changes in its environment, a critical capability for real-world autonomous systems. This ensures that the vehicle can safely handle dynamic and potentially hazardous conditions, thereby improving the reliability and safety of the system.
   - **Example**:

     "In the source scenario, the ego vehicle is going to park in a slot in an elevated car parking slot. Suddenly the slot begins to move, and the ego vehicle stops. In the follow-up scenario, the moving slot has been fixed, and the ego vehicle continues to park in the slot."

5. **Spatial Awareness Challenges**

- **Innovative element within the scenario**: Car doors open into the parking slot, blocking the ego vehicle.
- **Rational**: This element emphasizes the need for spatial awareness and adaptability when faced with dynamic obstacles. Such type of scenarios is rarely found in existing scenarios, which typically view other vehicles as whole, static entities without focusing on moveable elements like doors. This limitation often arises because simulating individual components adds complexity and variability, making the scenarios harder to control and replicate. By including such elements, human-generated MRs test the ego vehicle's ability to navigate around dynamically changing obstacles, reflecting real-world parking lot challenges where surrounding vehicles can create unexpected obstructions.
- **Example**:

  > *"In the source scenario, the ego vehicle is going to park in an empty slot. Suddenly the car next to the slot has opened its door on the side close to the target slot, and invaded the target slot. The ego vehicle stops parking accordingly. In the follow-up scenario, change the door to another side of the car, and the ego vehicle successfully parks in the slot."*

In summary, the ability of students to incorporate realistic, dynamic, and complex scenarios into their MRs reflects their intuitive understanding of real-world situations. These innovative elements enhance the relevance and applicability of their MRs, which demonstrates the human advantage in designing and transforming a broader range of real-life variables and interactions, adding significant value to the testing process.

Chapter 7

Discussion and Conclusion

Figure 7.1 illustrates the relationships between contributions and their role in achieving the thesis's ROs. This thesis examines the challenges and solutions related to the oracle problem in ADS testing through MT, focusing on both ADSs and simulators, as addressed in RO1 (*Examine the effectiveness of MT for ADS testing, involving ADS and AD simulators*). The initial application of ME on the Baidu Apollo ADS enhanced the understanding of the ADS mechanisms and MT familiarity, demonstrating ME's value in educating SQA professionals. It also inspired MT experiments on the object-detection algorithm of the perception-camera module, where adjusting brightness in scenarios revealed conflicting obstacle-detection results, addressing the oracle problem in validating complex ADS modules. Furthermore, acknowledging the importance of reliable simulation data in ADS testing, the research extended MT to evaluate AD simulators from NIO and CARLA. The experiments uncovered issues in the built-in functions and performance limitations, highlighting MT's effectiveness in validating simulation platforms to ensure ADS development accuracy.

To address the challenges of constructing test cases and generating MRs (RO2: *Tackle the challenge of MR generation, especially for beginners and testers in new systems*), the thesis introduces a structured test case template with MR genera-

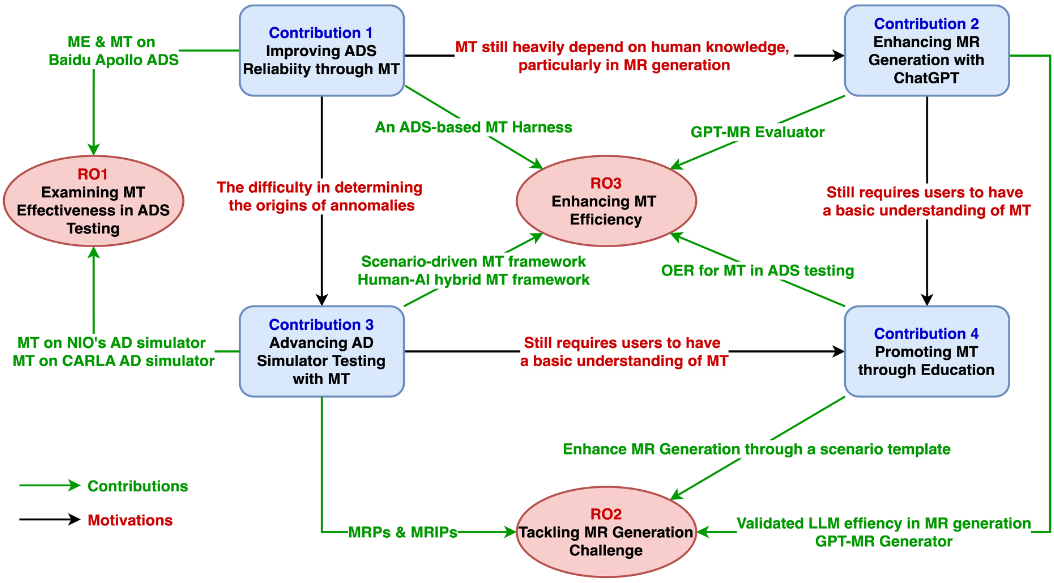


Figure 7.1: Connections among contributions and ROs

tion methodology to simplify scenario creation for beginners. The introduction of the MRPs and MRIPs provided diverse templates for various testing scenarios, reducing MR generation complexity and allowing testers to focus on identifying and resolving system defects. Additionally, the integration of LLMs like GPT-3.5 and GPT-4 further reduced barriers to MT. These models were proven capable of effectively generating high-quality MRs. This approach simplified MR generation, allowing less experienced testers to participate more easily and efficiently. Comparisons with human-generated MRs showed comparable performance, proving LLMs as valuable tools.

To enhance MT efficiency in ADS testing (RO3: *Enhance MT efficiency by automating and regulating the process of ADS testing, while also lowering the entry barriers for beginners using MT*), the thesis introduces various frameworks and tools. An ADS-based MT harness streamlined testing procedures, minimizing manual effort and improving test management. A scenario-based MT framework was developed to integrate ME and MT processes, covering the entire testing cycle from test case generation to result validation. Both frameworks were validated through industry case studies. A human-AI hybrid MT framework was also proposed to combine human expertise with AI-driven automation by integrating LLMs for MR generation with human oversight for MR selection and refinement. It also allowed test automation by integrating multiple modules designed specifically for the assessment of the simulator performance. Finally, the development of an OER significantly promoted MT and simplified ADS testing. It provided clear instructions and tools to streamline scenario creation and MR generation, lowering entry barriers. This fostered a better understanding of MT principles and encouraged broader adoption in practical applications.

In summary, this research has made meaningful advancements in the domain of MT for ADS testing, and has successfully met the ROs outlined in the Introduction Chapter. By addressing key challenges in MR generation and MT application, and by integrating innovative technologies such as LLMs, the thesis lays a foundation for more robust and efficient testing practices. The methodologies and tools developed not only enhance testing efficiencies but also provide a roadmap for future research aimed at ensuring the safety and reliability of complex autonomous systems.

7.1 Addressing the Research Questions

The following contents summarize the work performed in this PhD project to address the RQs outlined in the Introduction of the thesis (Section 1.2).

RQ1. How can MT be effectively applied in ADS testing to uncover anomalies and enhance system understanding?

In the Baidu Apollo ADS experiments, MT was successfully applied to uncover anomalies and enhance system understanding. In the ME practice on the perception and localization modules of the Baidu Apollo ADS (Section 3.2), the insights gained from ME not only contribute to a deeper understanding of the ADS but also suggest the potential use of ME experiences in teaching and training SQA professionals. Furthermore, the findings from ME serve as a foundation for the subsequent MT experiments, in which the author applied MT on the object-detection algorithm of the perception-camera module (Section 3.3.3). The experiments revealed conflicting obstacle-detection results by raising the brightness of a specific part of the driving scenarios, both in individual and sequential frames, demonstrating the ability of MT to address the oracle problem when validating the perception module of ADSs.

During the experiments of testing ADS, it is sometimes difficult to determine the origin of the anomalies: Whether it comes from the ADS or the simulator. Therefore, the author realized that ensuring the validity of AD simulators was also important in the overall testing process of ADSs, and it was an area that has been relatively underexplored in the MT testing domain. Experiments on the NIO AD simulator (Section 5.2.3) demonstrated that MT was effective in identifying various issues within the simulator, thus addressing the oracle problem in AD simulator testing. Subsequent experiments on the CARLA simulator (Section 5.3), implementing the MRPs proposed in NIO simulator testing, identified four major bugs linked to the simulator's built-in functions and performance. These findings underscore the importance of validating simulators to ensure reliable ADS testing outcomes, highlighting that MT can be systematically applied to uncover hidden issues and improve the robustness and accuracy of AD simulators.

In summary, by covering both ADSs and AD simulators, the two most significant areas in ADS testing [110], the experimental results successfully demonstrate the effectiveness of MT in this domain.

RQ2. How can the difficulty of generating MRs be reduced to assist beginners and testers in understanding and testing new systems?

MR generation is a critical challenge in MT, specifically for beginners and new testers to the system [123]. To address this issue, the author proposed a series of approaches to regulate and streamline the MR generation process for testing ADSs and simulators, including guidelines, patterns, and integration with new technologies.

The MR generation methodology, along with a test case generation template for ADSs included in the OER (Section 6.2.1), can both simplify the initial steps in scenario creation and reduce the difficulty of generating MRs, making it more efficient and understandable, especially for beginners who are unfamiliar with MT and testing ADSs. The methodology includes four equations that explain the relationships between the MR and test scenarios, outlining the logic for generating MRs without needing prior knowledge of the MR generation process. Additionally, three MRPs and MRIPs were proposed for the purpose of testing the AD simulator and ADSs (Section 5.2.1). These MRPs/MRIPs were able to generate a substantial number of MRs and reveal a number of defects during testing. An MR generation instruction on the usage of MRPs is also provided to reduce the difficulty of using MRPs.

With the fast development of LLMs, this thesis examines their effectiveness in reducing the difficulty and complexity of generating MRs, especially for beginners and new testers. The exploration with ChatGPT, specifically GPT-3.5 and GPT-4, demonstrates it could significantly simplify the MR generation process for MT (Sections 4.2 and 4.3). Evaluations reveal that these AI-driven methods produce high-quality MRs tailored to ADS functions, with a customized GPT-MR evaluator performing comparably to human evaluators, showing AI's potential to enhance MT practices. Comparisons with MR generation by students (Section 6.3) further underscore that LLMs can effectively support those struggling with MR generation, making them valuable tools for helping novices improve their understanding of the system.

RQ3. How can MT usage in ADS testing be simplified to increase efficiency and lower adoption barriers?

During the testing of the Baidu Apollo ADS, the author proposed an ADS-based MT harness to facilitate the testing of ADSs (Section 3.3.2), which would increase efficiency and help testers better organize the testing procedure. An industry case study to illustrate its usage in the actual production phases is also presented. This test harness can not only benefit testers by increasing their productivity when testing ADSs, but the case study can also raise awareness of the additional human efforts wasted during the testing-preparation phase.

In addition, to regulate and simplify the testing process of MT for AD simulators, the author proposed a scenario-based MT framework that integrates both ME and MT processes (Section 5.2.2). The framework encompasses the entire testing cycle, from test-case generation to result validation. Its self-evolving design allows testers to iteratively refine test cases and MRs while enhancing their understanding of the system until system defects are revealed. This approach can significantly reduce the time and effort needed to prepare for testing a new system. The author also presents an industry case study to illustrate the framework's strengths and limitations and suggests areas for further enhancement.

To leverage the effectiveness of LLMs in streamlining MT, a human-AI hybrid MT framework was proposed (Section 5.3.1). This framework combines human input with AI automation to generate and refine MRs, which involves a GPT-MR generator, a customized GPT designed with the MRPs outlined in this thesis, capable of producing MRs based on user specifications. MT experts select and refine these MRs, which are then fed into a test harness that automates the testing process, including test case generation and execution. The program interface allows users to input parameters that trigger specific functions, such as generating follow-up test scenarios and executing multiple scenarios sequentially. It can also create vehicle control sequences for managing vehicle movements in simulations. The harness consists of five internal modules, each dedicated to executing and creating scenarios for different MRs.

Finally, the author developed an OER (Section 6.2) to help lower the learning and application barriers of MT for students, testers, and SQA professionals who are new to MT and testing ADSs. It contains instructions and tools to simplify the steps in scenario creation, making MR generation more efficient and understandable for beginners. It not only bridges the knowledge gap but also fosters a deeper understanding of MT principles and their application in real-world scenarios.

7.2 Recommendations for Future Work

The following recommendations may guide future work based on this thesis:

1. **Enhance the applications of MT on other ADSs and AD-related platforms**: This thesis involves MT on the ADS and AD simulators (Chapters 3 and 5), as in the context of ADS testing. However, there are other systems contributing to ADS validity that remain underexplored by MT. For instance, communication systems that encompass Vehicle-to-Everything (V2X) technology [238]. These systems allow ADS-equipped vehicles to communicate with other road users and infrastructures. MT could test the reliability of ADS in handling communication-related scenarios by introducing variations in V2X data, which would test the system's adaptability and robustness. For instance, testing conflicting V2X signals from different sources, like two vehicles reporting differing road conditions, could test the ADS's capacity to prioritize or merge data accurately.

2. **Enhance the framework proposed in this study:** This thesis has proposed various frameworks and guidelines designed to enhance MT efficiency and simplify the generation of MRs, as detailed in Section 7.1. Although these approaches targeted ADS testing, future work could adapt them to other ADSs and software. This would enhance testing efficiency by minimizing manual efforts, allowing users to identify limitations and opportunities for refinement. Some frameworks may even be combined to provide a more general and streamlined user experience. For instance, integrating the ADS-based test harness into the human-AI hybrid MT framework could extend its applicability to both ADS and AD simulators. With AI-driven MR generation capabilities, this integration would make the framework more powerful in enhancing testing efficiency and accuracy.

3. **Enhance MRPs/MRIPs to cover a wider range of MRs and improve applicability to various SUTs:** The proposed MRPs/MRIPs are a significant contribution of this thesis (Section 5.2.1). While initially designed for ADS testing, the exploration in Section 5.3.6.3 has shown the potential of the LLM configured with these MRPs to generate MRs for other application domains. Therefore, future work could enhance their adaptability and generalizability by testing the MRs derived from these MRPs/MRIPs on SUTs across different domains and refining them based on the results [132].

4. **Improve the MR generation strategy on the using the LLMs:** In Section 4.2.2.3, efforts of using prompt-engineering to improve the quality of ChatGPT's answers have been introduced. Although it is proven effective, this approach is relatively brief and requires users to have a foundational understanding of MT and some experience. Therefore, future work could focus on developing a systematic guideline for interacting with LLMs using minimal prompts to enhance MT application. This thesis has introduced two customized GPT models, a GPT-MR evaluator (Section 4.3.3.2) and a

GPT-MR generator (Section 5.3.1.1). Future work could focus on integrating these models to automatically generate and select the effective MRs. For instance, developing an expert system that covers the entire MT cycle could significantly simplify the learning process and enhance testing efficiency.

5. **Connect LLMs with the OER introduced in this thesis:** In alignment with OER principles (open access, collaboration and sharing) [206], the source code of the website in Section 6.2 is available as open-source on GitHub. Given the powerful capabilities of LLMs to help users grasp new concepts quickly [213], and many existing products offer APIs [26] for integration with other applications, future work could work on connecting the website to the GPT models introduced in this thesis. This would enable users to efficiently generate and assess their MRs, enhancing the educational impact of the website.

6. **Conduct a systematic quantitative analysis comparing GPT-generated and human-derived MRs:** This thesis has demonstrated the effectiveness of GPT-driven MR generation (Section 5.3), yet quantitative comparisons between GPT-generated and human-derived MRs remain underexplored. Future work could systematically evaluate these MRs using metrics such as fault-detection rates, improvements in test coverage, and the ability to reveal previously undetected faults. Conducting such an analysis would provide users with a more comprehensive understanding of both MR generation techniques, enhancing guidance in teaching MR generation, improving strategies for training LLMs, and exploring possibilities to combine the strengths of AI-driven and human-driven approaches.

References

[1] Shamsu Abdullahi, Abubakar Zakari, Haruna Abdu, Amina Nura, Musa Ahmed Zayyad, Salisu Suleiman, Alhassan Adamu, and Abdulfatahu Samaila Mashasha. “Software Testing: Review on Tools, Techniques and Challenges”. In: *International Journal of Advanced Research in Technology and Innovation* 2.2 (2020), pp. 11–18. ISSN: 2682-8324.

[2] *About NIO*. Online. Available: `https://www.nio.com/about`. [Accessed: Dec. 26, 2023]. NIO Inc., Jan. 2023.

[3] Jubril Gbolahan Adigun, Linus Eisele, and Michael Felderer. “Metamorphic Testing in Autonomous System Simulations”. In: *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2022, pp. 330–337. DOI: `10.1109/SEAA56994.2022.00059`.

[4] John Ahlgren, Maria Berezin, Kinga Bojarczuk, Elena Dulskyte, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Maria Lomeli, Erik Meijer, Silvia Sapora, and Justin Spahr-Summers. “Testing Web Enabled Simulation at Scale Using Metamorphic Testing”. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 2021, pp. 140–149. DOI: `10.1109/ICSE-SEIP52600.2021.00023`.

[5] Abhimanyu S Ahuja. “The impact of artificial intelligence in medicine on the future role of the physician”. In: *PeerJ* 7 (2019), e7702. DOI: `10.7717/peerj.7702`.

[6] Faisal S Alsubaei. “Reliability and Security Analysis of Artificial Intelligence-Based Self-Driving Technologies in Saudi Arabia: A Case Study of Openpilot”. In: *Journal of advanced transportation* 2022.1 (2022), p. 2085225.

[7] Haiyang Ao and Ya Pan. “SMET: Scenario-based Metamorphic Testing for Autonomous Driving Models”. In: *arXiv preprint arXiv:2104.01093* (2021).

[8] *Apollo*. Online. Available: `https://github.com/ApolloAuto/apollo`. [Accessed: Jun. 20, 2024]. ApolloAuto, 2024.

[9] ApolloAuto. *Apollo*. `https://github.com/ApolloAuto/apollo/blob/master/docs/cyber/CyberRT_Developer_Tools.md`. 2020.

[10] ApolloAuto. *Apollo Planning Module Introduction*. Online. Available: `https://developer.apollo.auto/document_cn.html?target=/Apollo-Homepage-Document/Apollo_Doc_CN_6_0/`. [Accessed: Jun. 27, 2022]. ApolloAuto, May 2022.

[11] ApolloAuto. *Apollo Minibus: Autonomous Driving Solutions*. Online. Available: `https://www.apollo.auto/minibus`. [Accessed: Oct. 20, 2020]. Baidu, 2024.

[12] Mahmuda Asrafi, Huai Liu, and Fei-Ching Kuo. “On Testing Effectiveness of Metamorphic Relations: A Case Study”. In: *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement*. 2011, pp. 147–156. DOI: `10.1109/SSIRI.2011.21`.

[13] Association for Standardization of Automation and Measuring Systems. *ASAM OpenSCENARIO: User Guide*. Online. Available: `https://www.asam.net/standards/detail/openscenario/`. [Accessed: Jun. 2, 2023]. 2023.

[14] *Autopilot and Full Self-Driving Capability — Tesla Support Other Europe.* Online. Available: https://www.tesla.com/en_eu/support/autopilot. [Accessed: May 11, 2023]. Tesla, 2023.

[15] *Autoware Concepts.* Online. Available: https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-concepts/. [Accessed: Jun. 20, 2024]. Autoware Foundation, 2024.

[16] *Autoware Overview.* Online. Available: https://autoware.org/autoware-overview/. [Accessed: Jun. 20, 2024]. Autoware, 2024.

[17] Jon Ayerdi, Asier Iriarte, Pablo Valle, Ibai Roman, Miren Illarramendi, and Aitor Arrieta. "Metamorphic Runtime Monitoring of Autonomous Driving Systems". In: *arXiv preprint arXiv:2310.07414* (2023).

[18] Jon Ayerdi, Valerio Terragni, Aitor Arrieta, Paolo Tonella, Goiuria Sagardui, and Maite Arratibel. "Generating metamorphic relations for cyber-physical systems with genetic programming: an industrial case study". In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021, pp. 1264–1274. ISBN: 9781450385626. DOI: 10.1145/3468264.3473920.

[19] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixão, Filipe Mutz, Lucas de Paula Veronese, Thiago Oliveira-Santos, and Alberto F. De Souza. "Self-driving cars: A survey". In: *Expert Systems with Applications* (2020), p. 113816.

[20] Luciano Baresi and Damian A Tamburri. "Architecting Artificial Intelligence for Autonomous Cars: The OpenPilot Framework". In: *European Conference on Software Architecture*. Springer. 2023, pp. 189–204.

[21] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. “The oracle problem in software testing: A survey”. In: *IEEE transactions on software engineering* 41.5 (2014), pp. 507–525.

[22] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, and H. W. Schmidt. “The Impact of Source Test Case Selection on the Effectiveness of Metamorphic Testing”. In: *Proc. 1st Int. Workshop Metamorphic Testing (MET ’16)*. Austin, Texas: Association for Computing Machinery, 2016, pp. 5–11. ISBN: 9781450341639. DOI: 10.1145/2896971.2896977.

[23] Babak Bashari Rad, Harrison Bhatti, and Mohammad Ahmadi. “An Introduction to Docker and Analysis of its Performance”. In: *IJCSNS International Journal of Computer Science and Network Security* 173 (Mar. 2017), p. 8.

[24] Gagandeep Batra and Jyotsna Sengupta. “An efficient metamorphic testing technique using genetic algorithm”. In: *Information Intelligence, Systems, Technology and Management: 5th International Conference, ICISTM 2011, Gurgaon, India, March 10-12, 2011. Proceedings 5.* Springer. 2011, pp. 180–188.

[25] Arianna Blasi, Alessandra Gorla, Michael D. Ernst, Mauro Pezzè, and Antonio Carzaniga. “MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation”. In: *Journal of Systems and Software* 181.C (Nov. 2021). ISSN: 0164-1212. DOI: 10.1016/j.jss.2021.111041.

[26] Joshua Bloch. “How to design a good API and why it matters”. In: *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications.* 2006, pp. 506–507.

[27] Barry Boehm and Richard Turner. “Management Challenges to Implementing Agile Processes in Traditional Development Organizations”. In:

IEEE Software 22.5 (2005), pp. 30–39. ISSN: 0740-7459. DOI: 10.1109/MS.2005.129.

[28] Peter Bossaerts and Carsten Murawski. “Computational complexity and human decision-making”. In: *Trends in cognitive sciences* 21.12 (2017), pp. 917–929.

[29] Guy A Boysen, David L Vogel, Marissa A Cope, and Asale Hubbard. “Incidents of bias in college classrooms: Instructor and student perceptions.” In: *Journal of Diversity in Higher Education* 2.4 (2009), p. 219.

[30] Joshua Brown, Zhi Quan Zhou, and Yang-Wai Chow. “Metamorphic testing of navigation software: A pilot study with Google Maps”. In: *Proceedings of the 51st Hawaii International Conference on System Sciences.* 2018, pp. 5687–5696. DOI: 10.24251/HICSS.2018.713.

[31] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. *Sparks of artificial general intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[32] Darcy Bullock, Brian Johnson, Richard B Wells, Michael Kyte, and Zhen Li. “Hardware-in-the-loop simulation”. In: *Transportation Research Part C: Emerging Technologies* 12.1 (2004), pp. 73–89.

[33] Vince D. Calhoun and Godfrey Pearlson. “A selective review of simulated driving studies: Combining naturalistic and hybrid paradigms, analysis approaches, and future directions”. In: *NeuroImage* 59.1 (2012). Neuroergonomics: The human brain in action and at work, pp. 25–35. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2011.06.037.

[34] Gerardo Canfora, Massimiliano Di Penta, and Luigi Cerulo. “Achievements and challenges in software reverse engineering”. In: *Communications of the ACM* 54.4 (2011), pp. 142–151.

[35] Yuxiang Cao, Zhi Quan Zhou, and Tsong Yueh Chen. "On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions". In: *2013 13th International Conference on Quality Software*. IEEE. 2013, pp. 153–162.

[36] *Case Studies*. Online. Available: `https://autoware.org/case-studies/`. [Accessed: Jun. 20, 2024]. Autoware, 2024.

[37] Tom Caswell, Shelley Henson, Marion Jensen, and David Wiley. "Open Educational Resources: Enabling universal education". In: *International Review of Research in Open and Distributed Learning* 9.1 (2008), pp. 1–11.

[38] Kwok Ping Chan, Tsong Yueh Chen, Fei-Ching Kuo, and Dave Towey. "A revisit of adaptive random testing by restriction". In: *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004*. IEEE. 2004, pp. 78–85.

[39] Wing Kwong Chan, Tsong Yueh Chen, Heng Lu, TH Tse, and Stephen S Yau. "Integration testing of context-sensitive middleware-based applications: a metamorphic approach". In: *International Journal of Software Engineering and Knowledge Engineering* 16.05 (2006), pp. 677–703.

[40] changsh726. *Apollo cyber-record file parser (Python based)*. GitHub. Available: `https://github.com/ApolloAuto/apollo/blob/266afbf68d83fa6fac7a812ff8a950223f5ab2c0/modules/tools/record_parse_save/README.md`. [Accessed: 2020]. 2020.

[41] Qianwen Chao, Huikun Bi, Weizi Li, Tianlu Mao, Zhaoqi Wang, and Ming Lin. "A Survey on Visual Traffic Simulation: Models, Evaluations, and Applications in Autonomous Driving". In: *Computer Graphics Forum* 39 (July 2019). DOI: `10.1111/cgf.13803`.

[42] Leilei Chen, Lizhi Cai, Jiang Liu, Zhenyu Liu, Shiyan Wei, and Pan Liu. "An optimized method for generating cases of metamorphic testing". In:

2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012). 2012, pp. 439–443.

[43] Li Chen, Tutian Tang, Zhitian Cai, Yang Li, Penghao Wu, Hongyang Li, Jianping Shi, Junchi Yan, and Yu Qiao. *Level 2 Autonomous Driving on a Single Device: Diving into the Devils of Openpilot*. 2022. arXiv: 2206.08176.

[44] Lingjiao Chen, Matei Zaharia, and James Zou. "How is ChatGPT's behavior changing over time?" In: *Harvard Data Science Review* 6.2 (2024).

[45] T. Y. Chen, S. C. Cheung, and S. M. Yiu. *Metamorphic Testing: A New Approach for Generating Next Test Cases*. Technical Report HKUST-CS98-01. Hong Kong: Department of Computer Science, The Hong Kong University of Science and Technology, 1998.

[46] Tsong Yueh Chen, DH Huang, TH Tse, and Zhi Quan Zhou. "Case Studies on the Selection of Useful Relations in Metamorphic Testing". In: *Proc. of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004)*. Citeseer, 2004, pp. 569–583.

[47] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. "Metamorphic testing: A review of challenges and opportunities". In: *ACM Computing Surveys (CSUR)* 51.1 (2018), 4:1–4:27.

[48] Tsong Yueh Chen, Fei-Ching Kuo, Wenjuan Ma, Willy Susilo, Dave Towey, Jeffrey Voas, and Zhi Quan Zhou. "Metamorphic testing for cybersecurity". In: *Computer* 49.6 (2016), pp. 48–55.

[49] Tsong Yueh Chen, Fei-Ching Kuo, Robert G Merkel, and TH Tse. "Adaptive random testing: The art of test case diversity". In: *Journal of Systems and Software* 83.1 (2010), pp. 60–66.

[50] Tsong Yueh Chen, Pak-Lok Poon, and TH Tse. “A choice relation framework for supporting category-partition test case generation”. In: *IEEE transactions on software engineering* 29.7 (2003), pp. 577–593.

[51] Tsong Yueh Chen, Pak-Lok Poon, and Xiaoyuan Xie. “METRIC: METamorphic Relation Identification based on the Category-choice framework”. In: *Journal of Systems and Software* 116 (2016), pp. 177–190. ISSN: 0164-1212. DOI: 10.1016/j.jss.2015.07.037.

[52] Xiang Chen, Meranda Salem, Tuhin Das, and Xiaoqun Chen. “Real time software-in-the-loop simulation for control performance validation”. In: *Simulation* 84.8-9 (2008), pp. 457–471.

[53] *Conversation with Autoware Foundation*. Online. Available: https://web.moda.gov.tw/en/press/background-information/4027. [Accessed: Jun. 20, 2024]. Ministry of Digital Affairs, Taiwan, 2024.

[54] Creative Commons. *Share your work - Creative Commons 2023*. Online. Available: https://creativecommons.org/share-your-work/. [Accessed: Dec. 14, 2023]. Sept. 2023.

[55] Therese M Cumming and Megan C Rose. “Exploring universal design for learning as an accessibility tool in higher education: A review of the current literature”. In: *The Australian Educational Researcher* 49.5 (2022), pp. 1025–1043.

[56] Ermira Daka and Gordon Fraser. “A Survey on Unit Testing Practices and Problems”. In: *2014 IEEE 25th International Symposium on Software Reliability Engineering*. 2014, pp. 201–211. DOI: 10.1109/ISSRE.2014.11.

[57] Martin D Davis and Elaine J Weyuker. “Pseudo-oracles for non-testable programs”. In: *Proceedings of the ACM’81 Conference*. 1981, pp. 254–257.

[58] Yao Deng, Guannan Lou, Xi Zheng, Tianyi Zhang, Miryung Kim, Huai Liu, Chen Wang, and Tsong Yueh Chen. “BMT: Behaviour driven development-based metamorphic testing for autonomous driving models”.

In: *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. IEEE. 2021, pp. 32–36.

[59] Yao Deng, Xi Zheng, Tianyi Zhang, Huai Liu, Guannan Lou, Miryung Kim, and Tsong Yueh Chen. "A Declarative Metamorphic Testing Framework for Autonomous Driving". In: *IEEE Transactions on Software Engineering* 49.4 (2023), pp. 1964–1982. DOI: 10.1109/TSE.2022.3206427.

[60] Yao Deng, Xi Zheng, Tianyi Zhang, Guannan Lou, Huai Liu, and Miryung Kim. "RMT: Rule-based metamorphic testing for autonomous driving models". In: *arXiv* (2021), pp. 1–12.

[61] Vaishali M Deshmukh, Rajalakshmi B, Gopi BS Krishna, and Gourav Rudrawar. "An Overview of Deep Learning Techniques for Autonomous Driving Vehicles". In: *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2022, pp. 979–983. DOI: 10.1109/ICSSIT53264.2022.9716433.

[62] Nitish M. Devadiga. "Software Engineering Education: Converging with the Startup Industry". In: *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. 2017, pp. 192–196. DOI: 10.1109/CSEET.2017.38.

[63] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[64] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. "Euclidean distance matrices: essential theory, algorithms, and applications". In: *IEEE Signal Processing Magazine* 32.6 (2015), pp. 12–30.

[65] Guowei Dong, Baowen Xu, Lin Chen, Changhai Nie, and Lulu Wang. "Case studies on testing with compositional metamorphic relations". In: *Journal of Southeast University (English Edition)* 24.4 (2008), pp. 437–443.

[66] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[67] *DuerOS*. Online. Available: `https://apollo.auto/platform/dueros/dueros.html`. [Accessed: Oct. 20, 2020]. Baidu, 2024.

[68] Gaurav Duggal and Bharti Suri. "Understanding regression testing techniques". In: *Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology*. Citeseer. 2008.

[69] Peggy A. Ertmer. "Addressing first- and second-order barriers to change: Strategies for technology integration". In: *Educational Technology Research and Development* 47.4 (1999), pp. 47–61. DOI: `10.1007/BF02299597`.

[70] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. "The many faces of publish/subscribe". In: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[71] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. "Baidu Apollo EM Motion Planner". In: *arXiv preprint arXiv:1807.08048* (2018).

[72] Fahime Farahi and Hadi Sadoghi Yazdi. "Probabilistic Kalman filter for moving object tracking". In: *Signal Processing: Image Communication* 82 (2020), p. 115751. ISSN: 0923-5965. DOI: `10.1016/j.image.2019.115751`.

[73] Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. "Mathematical capabilities of ChatGPT". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23)*. New Orleans, LA, USA: Curran Associates Inc., 2024, pp. 1–46.

[74] Gagandeep and Gurdeepak Singh. “An automated metamorphic testing technique for designing effective metamorphic relations”. In: *Contemporary Computing: 5th International Conference, IC3 2012, Noida, India, August 6-8, 2012. Proceedings 5*. Springer. 2012, pp. 152–163.

[75] Marie-Claude Gaudel. “Software testing based on formal specification”. In: *Testing Techniques in Software Engineering. PSSE 2007. Lecture Notes in Computer Science*. Ed. by Paulo Borba, Ana Cavalcanti, Augusto Sampaio, and Jim Woodcook. Berlin, Heidelberg: Springer, 2010, pp. 215–242. ISBN: 978-3-642-14335-9. DOI: `10.1007/978-3-642-14335-9_7`.

[76] Liesel Gentelli. “Using industry professionals in undergraduate teaching: Effects on student learning”. In: *Journal of University Teaching & Learning Practice* 12.4 (2015). DOI: `10.53761/1.12.4.4`.

[77] Louie Giray. “Prompt engineering with ChatGPT: a guide for academic writers”. In: *Annals of biomedical engineering* 51.12 (2023), pp. 2629–2633.

[78] glopezdiest. *atomic-bahavior.py [Source code]*. GitHub. Available: `https://github.com/carla-simulator/scenario_runner/blob/f1a94516db57612f22d5c42e597e350e781b9224/srunner/scenariomanager/scenarioatomics/atomic_behaviors.py`. [Accessed: Jun. 7, 2023]. 2023.

[79] glopezdiest. *Get ScenarioRunner*. GitHub. Available: `https://github.com/carla-simulator/scenario_runner/blob/master/Docs/getting_scenariorunner.md`. [Accessed: Jun. 7, 2023]. 2023.

[80] Alex Groce, Todd Kulesza, Chaoqiang Zhang, Shalini Shamasunder, Margaret Burnett, Weng-Keen Wong, Simone Stumpf, Shubhomoy Das, Amber Shinsel, Forrest Bice, et al. “You are the only possible oracle: Effective test selection for end users of interactive machine learning systems”. In: *IEEE Transactions on Software Engineering* 40.3 (2013), pp. 307–323.

[81] An Guo, Yang Feng, and Zhenyu Chen. "LiRTest: Augmenting LiDAR point clouds for automated testing of autonomous driving systems". In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022)*. Virtual, South Korea: ACM, 2022, pp. 480–492. ISBN: 9781450393799. DOI: 10.1145/3533767.3534397.

[82] Richard Hamlet and J Maciniak. "Random testing. Encyclopedia of software Engineering". In: *Wiley: New York* (1994), pp. 970–978.

[83] Jia Cheng Han and Zhi Quan Zhou. "Metamorphic Fuzz Testing of Autonomous Vehicles". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*. Seoul, Republic of Korea: ACM, 2020, pp. 380–385. ISBN: 9781450379632. DOI: 10.1145/3387940.3392252.

[84] Mark Harman, Yue Jia, and Yuanyuan Zhang. "Achievements, Open Problems and Challenges for Search Based Software Testing". In: *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. 2015, pp. 1–12. DOI: 10.1109/ICST.2015.7102580.

[85] Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "A comprehensive survey of trends in oracles for software testing". In: *University of Sheffield, Tech. Rep. CS-13-01* (2013).

[86] Jie Hong, Jie Zhang, Qi Qiu, Angang Ma, Meng Li, Shiyu Yan, and Helin Gong. "A Dynamic Recognition Method of Metamorphic Relation Identification". In: *2022 13th International Conference on Reliability, Maintainability, and Safety (ICRMS)*. IEEE. 2022, pp. 81–86.

[87] Peifeng Hu, Zhenyu Zhang, W. K. Chan, and T. H. Tse. "An Empirical Comparison Between Direct and Indirect Test Result Checking Approaches". In: *Proc. 3rd Int. Workshop Software Quality Assurance (SOQUA '06)*. Portland, Oregon: ACM, 2006, pp. 6–13. ISBN: 1595935843. DOI: 10.1145/1188895.1188901.

[88] Peng Huang, Marco Ceccagnoli, Chris Forman, and DJ Wu. "Appropriability mechanisms and the platform partnership decision: Evidence from enterprise software". In: *Management Science* 59.1 (2013), pp. 102–121.

[89] Zhan-wei Hui and Song Huang. "MD-ART: a test case generation method without test oracle problem". In: *Proceedings of the 1st International Workshop on Specification, Comprehension, Testing, and Debugging of Concurrent Programs* (2016). DOI: 10.1145/2975954.2975959.

[90] Zhan-wei Hui, Xiaojuan Wang, Song Huang, and Sen Yang. "MT-ART: A Test Case Generation Method Based on Adaptive Random Testing and Metamorphic Relation". In: *IEEE Transactions on Reliability* 70 (2021), pp. 1397–1421. DOI: 10.1109/tr.2021.3106389.

[91] *Introducing ChatGPT*. Online. Available: https://openai.com/blog/chatgpt. [Accessed: Jun. 4, 2023]. OpenAI, Nov. 2022.

[92] *Introducing GPTs*. Online. Available: https://openai.com/index/introducing-gpts/. [Accessed: Jul. 30, 2024]. OpenAI, Apr. 2023.

[93] *Introducing Meta Llama 3: The most capable openly available LLM to date*. Online. Available: https://ai.meta.com/blog/meta-llama-3/. [Accessed: Apr. 25, 2024]. Meta, June 2024.

[94] Muhammad Iqbal. "Metamorphic Testing of Advanced Driver-Assistance Systems: Implementing Euro NCAP Standards on OpenStreetMap". In: *2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)*. IEEE. 2023, pp. 1–8.

[95] Muhammad Iqbal, Jia Cheng Han, Zhi Quan Zhou, Dave Towey, and Tsong Yueh Chen. "Metamorphic testing of Advanced Driver-Assistance System (ADAS) simulation platforms: Lane Keeping Assist System (LKAS) case studies". English. In: *Information and Software Technology* 155 (Mar. 2023). ISSN: 0950-5849. DOI: 10.1016/j.infsof.2022.107104.

[96] Mahmudul Islam, Farhan Khan, Sabrina Alam, and Mahady Hasan. "Artificial Intelligence in Software Testing: A Systematic Review". In: *TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)*. 2023, pp. 524–529. DOI: 10.1109/TENCON58879.2023.10322349.

[97] Jackbart94. *Traffic Manager*. Online. Available: https://carla.readthedocs.io/en/latest/adv_traffic_manager/. [Accessed: Jun. 4, 2023]. 2023.

[98] Mohammad Hossein Jarrahi. "Artificial Intelligence and the future of work: Human-AI symbiosis in organizational decision making". In: *Business Horizons* 61.4 (2018), pp. 577–586. ISSN: 0007-6813. DOI: 10.1016/j.bushor.2018.03.007.

[99] Yue Jia and Mark Harman. "An Analysis and Survey of the Development of Mutation Testing". In: *IEEE Transactions on Software Engineering* 37.5 (2011), pp. 649–678. DOI: 10.1109/TSE.2010.62.

[100] Mingyue Jiang, Tsong Yueh Chen, Fei-Ching Kuo, and Zuohua Ding. "Testing central processing unit scheduling algorithms using metamorphic testing". In: *2013 IEEE 4th International Conference on Software Engineering and Service Science*. IEEE. 2013, pp. 530–536.

[101] Upulee Kanewala. "Techniques for automatic detection of metamorphic relations". In: *2014 IEEE seventh international conference on software testing, verification and validation workshops*. IEEE. 2014, pp. 237–238.

[102] Upulee Kanewala, James M Bieman, and Asa Ben-Hur. "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels". In: *Software testing, verification and reliability* 26.3 (2016), pp. 245–269.

[103] Upulee Kanewala and James M. Bieman. "Using machine learning techniques to detect metamorphic relations for programs without test oracles".

In: *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. 2013, pp. 1–10. DOI: 10.1109/ISSRE.2013.6698899.

[104] Lei Kang. *Baidu Apollo launches 6th-gen robotaxi with 60% lower cost.* Accessed on 15.03.2024. May 2024. URL: https://cnevpost.com/2024/05/15/baidu-apollo-launches-6th-gen-robotaxi/#:~:text=Baidu's%20self%2Ddriving%20ride%20hailing,to%20be%20profitable%20by%202025.&text=Chinese%20tech%20giant%20Baidu's%20smart,the%20business%20to%20become%20profitable..

[105] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. "Autoware on board: Enabling autonomous vehicles with embedded systems". In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. 2018, pp. 287–296. DOI: 10.1109/ICCPS.2018.00035.

[106] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[107] Muhammad Nadeem Khokhar, Muhammad Bilal Bashir, and Muhammad Fiaz. "Metamorphic Testing of AI-based Applications: A Critical Review". In: *International Journal of Advanced Computer Science and Applications* 11 (2020). DOI: 10.14569/ijacsa.2020.0110498.

[108] Tae Kyun Kim. "T test as a parametric statistic". In: *Korean journal of anesthesiology* 68.6 (2015), pp. 540–546.

[109] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. "Large language models are zero-shot reasoners". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*. New Orleans, LA, USA: Curran Associates Inc., 2024, pp. 1–15. ISBN: 9781713871088.

[110] Yavuz Koroglu and Franz Wotawa. "Towards a Review on Simulated ADAS/AD Testing". In: *2023 IEEE/ACM International Conference on Automation of Software Test (AST)*. 2023, pp. 112–122. DOI: 10.1109/AST58925.2023.00015.

[111] Toma Kosar, Pablo E. Martınez López, Pablo A. Barrientos, and Marjan Mernik. "A preliminary study on various implementation approaches of domain-specific language". In: *Information and software technology* 50.5 (Apr. 2008), pp. 390–405. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2007.04.002.

[112] Axel van Lamsweerde. "Formal specification: a roadmap". In: *Proceedings of the Conference on the Future of Software Engineering*. 2000, pp. 147–159.

[113] Vu Le, Mehrdad Afshari, and Zhendong Su. "Compiler validation via equivalence modulo inputs". In: *ACM SIGPLAN Notices* 49.6 (2014), pp. 216–226.

[114] David Lee and Mihalis Yannakakis. "Principles and methods of testing finite state machines-a survey". In: *Proceedings of the IEEE* 84.8 (1996), pp. 1090–1123. DOI: 10.1109/5.533956.

[115] Tahar Lejmi and Ihsan Sabuncuoglu. "Effect of load, processing time and due date variation on the effectiveness of scheduling rules". In: *International journal of production research* 40.4 (2002), pp. 945–974.

[116] Josh Lerner and Jean Tirole. "The open source movement: Key research questions". In: *European economic review* 45.4-6 (2001), pp. 819–826.

[117] Lawrence Lessig. "The creative commons". In: *Copyright Law*. Routledge, 2017, pp. 335–347.

[118] Hareton KN Leung and Lee White. "Insights into regression testing (software testing)". In: *Proceedings. Conference on Software Maintenance-1989*. IEEE. 1989, pp. 60–69.

[119] Sam Levin. *Uber crash shows 'catastrophic failure' of self-driving technology, experts say*. Online. Available: https://www.theguardian.com/technology/2018/mar/22/self-driving-car-uber-death-woman-failure-fatal-crash-arizona. [Accessed: Sep. 13, 2020]. Guardian News and Media, Mar. 2018.

[120] Daofei Li and Ao Liu. "Personalized highway pilot assist considering leading vehicle's lateral bahaviours". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 237.5 (2023), pp. 1146–1159.

[121] Jia Li, Yongmin Li, Ge Li, Zhi Jin, Yiyang Hao, and Xing Hu. "SKCODER: A Sketch-Based Approach for Automatic Code Generation". In: *Proceedings of the 45th International Conference on Software Engineering (ICSE '23)*. Melbourne, Victoria, Australia: IEEE Press, 2023, pp. 2124–2135. ISBN: 9781665457019. DOI: 10.1109/ICSE48619.2023.00179.

[122] Jingyao Li, Lei Liu, and Peng Zhang. "Tabular-expression-based method for constructing metamorphic relations". In: *Software: Practice and Experience* 50.8 (2020), pp. 1345–1380.

[123] Rui Li, Huai Liu, Pak-Lok Poon, Dave Towey, Chang-Ai Sun, Zheng Zheng, Zhi Quan Zhou, and Tsong Yueh Chen. *Metamorphic Relation Generation: State of the Art and Visions for Future Research*. 2024. arXiv: 2406.05397.

[124] Wei Li, Chengwei Pan, Rong Zhang, Jiaping Ren, Yuexin Ma, Jin Fang, Feilong Yan, Qichuan Geng, Xinyu Huang, Huajun Gong, Weiwei Xu, Guoping Wang, Dinesh Manocha, and Ruigang Yang. "AADS: Augmented autonomous driving simulation using data-driven algorithms". In: *Science Robotics* 4.28 (Mar. 2019). DOI: 10.1126/scirobotics.aaw0863.

[125] Mikael Lindvall, Dharmalingam Ganesan, Ragnar Árdal, and Robert E Wiegand. "Metamorphic model-based testing applied on NASA DAT–An

experience report”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. IEEE. 2015, pp. 129–138.

[126] Mikael Lindvall, Adam Porter, Gudjon Magnusson, and Christoph Schulze. “Metamorphic Model-Based Testing of Autonomous Systems”. In: *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*. 2017, pp. 35–41. DOI: 10.1109/MET.2017.6.

[127] Huai Liu, Fei-Ching Kuo, Dave Towey, and Tsong Yueh Chen. “How Effectively Does Metamorphic Testing Alleviate the Oracle Problem?” In: *IEEE Transactions on Software Engineering* 40.1 (2014), pp. 4–22. DOI: 10.1109/TSE.2013.46.

[128] Huai Liu, Xuan Liu, and Tsong Yueh Chen. “A new method for constructing metamorphic relations”. In: *12th International Conference on Quality Software*. IEEE. 2012, pp. 59–68.

[129] Jiawei Liu, Yuheng Huang, Zhijie Wang, Lei Ma, Chunrong Fang, Mingzheng Gu, Xufan Zhang, and Zhenyu Chen. “Generation-based Differential Fuzzing for Deep Learning Libraries”. In: *ACM Transactions on Software Engineering and Methodology* 33.2 (2023), pp. 1–28.

[130] Kui Liu, Dongsun Kim, Tegawendé F. Bissyandé, S. Yoo, and Y. Le Traon. “Mining Fix Patterns for FindBugs Violations”. In: *IEEE Transactions on Software Engineering* 47 (2017), pp. 165–188. DOI: 10.1109/TSE.2018.2884955.

[131] Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. “Computing Systems for Autonomous Driving: State of the Art and Challenges”. In: *IEEE Internet of Things Journal* 8.8 (2021), pp. 6469–6486. DOI: 10.1109/JIOT.2020.3043716.

[132] Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. “Testing of autonomous driving systems: where are we and where should we go?” In: *Proceedings of the 30th ACM Joint European Software Engi-*

neering Conference and Symposium on the Foundations of Software Engineering. 2022, pp. 31–43.

[133] Quang Hung Luu, Huai Liu, Tsong Yueh Chen, and Hai L. Vu. “A Sequential Metamorphic Testing Framework for Understanding Autonomous Vehicle’s Decisions”. In: *IEEE Transactions on Intelligent Vehicles* (Feb. 2024). ISSN: 2379-8904. DOI: 10.1109/TIV.2024.3370740.

[134] Xuewei Lv, Song Huang, Zhan-wei Hui, and Haijin Ji. “Test cases generation for multiple paths based on PSO algorithm with metamorphic relations”. In: *IET Software* 12 (2018), pp. 306–317. DOI: 10.1049/iet-sen.2017.0260.

[135] Zahid Mahmood, Ossama Haneef, Nazeer Muhammad, and Shahid Khattak. “Towards a fully automated car parking system”. In: *IET Intelligent Transport Systems* 13.2 (2019), pp. 293–302.

[136] MattRowe18. *Actors and blueprints*. Online. Available: https://carla.readthedocs.io/en/docs-preview/core_actors/. [Accessed: Jun. 4, 2023]. 2023.

[137] Johannes Mayer and Ralph Guderlei. “An Empirical Study on the Selection of Good Metamorphic Relations”. In: *30th Annual International Computer Software and Applications Conference (COMPSAC’06)*. Vol. 1. 2006, pp. 475–484.

[138] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. “From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 2383–2390.

[139] Selina Meyer, David Elsweiler, Bernd Ludwig, Marcos Fernandez-Pichel, and David E. Losada. “Do We Still Need Human Assessors? Prompt-Based GPT-3 User Simulation in Conversational AI”. In: *Proceedings of the 4th Conference on Conversational User Interfaces (CUI ’22)*. Glas-

gow, United Kingdom: ACM, 2022, pp. 1–6. ISBN: 9781450397391. DOI: 10.1145/3543829.3544529.

[140] Kunal Mishra, Gail Kaiser, and Swapneel Sheth. "Effectiveness of Teaching Metamorphic Testing, Part II". In: *Columbia University Academic Commons* (2013). DOI: 10.7916/D8902C8Q.

[141] Nadia Mullen, Judith Charlton, Anna Devlin, and Michel Bedard. "Simulator validity: Behaviours observed on the simulator and on the road". English. In: *Handbook of Driving Simulation for Engineering, Medicine and Psychology.* Ed. by Donald L Fisher, Matthew Rizzo, Jeff K Caird, and John D Lee. 1st. Australia: CRC Press, 2011, pp. 1–18. ISBN: 9781138074583.

[142] Markus U Müller, Nikoo Ekhtiari, Rodrigo M Almeida, and Christoph Rieke. "Super-resolution of multispectral satellite images using convolutional neural networks". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2020 (Aug. 2020), pp. 33–40. DOI: 10.5194/isprs-annals-v-1-2020-33-2020.

[143] Aravind Nair, Karl Meinke, and Sigrid Eldh. "Leveraging mutants for automatic prediction of metamorphic relations using machine learning". In: *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation.* 2019, pp. 1–6.

[144] Srinivas Nidhra. "Black Box and White Box Testing Techniques - A Literature Review". In: *International Journal of Embedded Systems and Applications* 2.2 (2012), pp. 29–50. DOI: 10.5121/ijesa.2012.2204.

[145] Srinivas Nidhra and Jagruthi Dondeti. "Black box and white box testing techniques-a literature review". In: *International Journal of Embedded Systems and Applications (IJESA)* 2.2 (2012), pp. 29–50.

[146] *NIO Assisted and Intelligent Driving: Safe Intelligent Driving*. Online. Available: https://www.nio.com/nad. [Accessed: May 11, 2023]. Nio.com, 2023.

[147] Rihards Novickis, Aleksandrs Levinskis, Roberts Kadikis, Vitalijs Fescenko, and Kaspars Ozols. "Functional Architecture for Autonomous Driving and its Implementation". In: *2020 17th Biennial Baltic Electronics Conference (BEC)*. 2020, pp. 1–6. DOI: 10.1109/BEC49624.2020.9276943.

[148] Alberto Núñez and Robert M Hierons. "A methodology for validating cloud models using metamorphic testing". In: *Ann. Telecommun.* 70.3-4 (2015), pp. 127–135.

[149] Megan Olsen and Mohammad Raunak. "Increasing Validity of Simulation Models Through Metamorphic Testing". In: *IEEE Transactions on Reliability* 68.1 (2019), pp. 91–108. DOI: 10.1109/TR.2018.2850315.

[150] OpenAI. *Hello GPT-4o*. Online. Available: https://openai.com/index/hello-gpt-4o/. [Accessed: May 25, 2024]. OpenAI, May 2024.

[151] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch,

Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael,

Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. "GPT-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[152] Thomas J. Ostrand and Marc J. Balcer. "The category-partition method for specifying and generating fuctional tests". In: *Communications of the ACM* 31.6 (1988), pp. 676–686.

[153] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. "Training language models to follow instructions with human

feedback”. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS ’22)*. New Orleans, LA, USA: Curran Associates Inc., 2024, pp. 1–15. ISBN: 9781713871088.

[154] Ya Pan, Haiyang Ao, and Yong Fan. “Metamorphic Testing for Autonomous Driving Systems in Fog based on Quantitative Measurement”. In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2021, pp. 30–37. DOI: 10.1109/QRS-C55045.2021.00015.

[155] Jon Perez-Cerrolaza, Jaume Abella, Markus Borg, Carlo Donzella, Jesús Cerquides, Francisco J. Cazorla, Cristofer Englund, Markus Tauber, George Nikolakopoulos, and Jose Luis Flores. “Artificial Intelligence for Safety-Critical Systems in Industrial and Transportation Domains: A Survey”. In: *ACM Computing Surveys* 56.7 (2024). ISSN: 0360-0300. DOI: 10.1145/3626314.

[156] Tobias Pieper and Roman Obermaisser. “Distributed co-simulation for software-in-the-loop testing of networked railway systems”. In: *2018 7th Mediterranean Conference on Embedded Computing (MECO)* (2018), pp. 1–5. DOI: 10.1109/MECO.2018.8406023.

[157] Pak-Lok Poon, Sau-Fun Tang, TH Tse, and Tsong Yueh Chen. “CHOC’LATE: A Framework for Specification-Based Testing”. In: *Communications of the ACM* 53.4 (Apr. 2010), pp. 113–118. ISSN: 0001-0782. DOI: 10.1145/1721654.1721687.

[158] pvgladkov. *atomic-criteria.py [Source code]*. GitHub. Available: https://github.com/carla-simulator/scenario_runner/blob/f1a94516db57612f22d5c42e597e350e781b9224/srunner/scenariomanager/scenarioatomics/atomic_criteria.py. [Accessed: Jun. 7, 2023]. 2023.

[159] Kun Qiu, Zheng Zheng, Tsong Yueh Chen, and Pak-Lok Poon. “Theoretical and empirical analyses of the effectiveness of metamorphic relation composition”. In: *IEEE Transactions on software engineering* 48.3 (2020), pp. 1001–1017.

[160] Karishma Rahman, Indika Kahanda, and Upulee Kanewala. “MRpredT: Using text mining for metamorphic relation prediction”. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, pp. 420–424.

[161] Vysyaraju Manikanta Raju, Vrinda Gupta, and Shailesh Lomate. “Performance of Open Autonomous Vehicle Platforms: Autoware and Apollo”. In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. 2019, pp. 1–5. DOI: 10.1109/I2CT45611.2019.9033734.

[162] Gerald Rau and Yu-Shan Shih. “Evaluation of Cohen’s kappa and other measures of inter-rater agreement for genre analysis and other nominal data”. In: *Journal of English for Academic Purposes* 53 (2021), p. 101026. DOI: 10.1016/J.JEAP.2021.101026.

[163] Mohammad S Raunak and Megan M Olsen. “Metamorphic testing on the continuum of verification and validation of simulation models”. In: *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. IEEE. 2021, pp. 47–52.

[164] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[165] Hongping Ren, Hui Gao, He Chen, and Guangzhen Liu. “A Survey of Autonomous Driving Scenarios and Scenario Databases”. In: *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*. 2022, pp. 754–762. DOI: 10.1109/DSA56465.2022.00107.

[166] Suzanne Robertson and James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. 3rd. Boston, MA, USA: Addison-Wesley, 2012, pp. 5–6.

[167] Stewart Robinson. "Three sources of simulation inaccuracy (and how to overcome them)". In: *WSC'99. 1999 Winter Simulation Conference Proceedings. 'Simulation - A Bridge to the Future' (Cat. No.99CH37038)*. Vol. 2. 1999, 1701–1708 vol.2. DOI: 10.1109/WSC.1999.816913.

[168] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Martins Mozeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. "LGSVL simulator: A high fidelity simulator for autonomous driving". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[169] *ROS2 Documentation*. Online. Available: https://index.ros.org/doc/ros2/. [Accessed: Oct. 10, 2020]. ROS Infrastructure, Oct. 2020.

[170] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. "Prioritizing test cases for regression testing". In: *IEEE Transactions on software engineering* 27.10 (2001), pp. 929–948.

[171] Daniel Rubio. "Beginning Django: Web Application Development and Deployment with Python". In: Berkeley, CA: Apress, 2017. Chap. 1, pp. 1–29. ISBN: 978-1-4842-2787-9. DOI: 10.1007/978-1-4842-2787-9_1.

[172] Jürgen Rudolph, Samson Tan, and Shannon Tan. "ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?" In: *Journal of Applied Learning and Teaching* 6.1 (2023).

[173] Aziz Salah, Rachida Dssouli, and Guy Lapalme. "Compiling Real-time Scenarios into a Timed Automaton". In: *Formal Techniques for Networked*

and Distributed Systems. Boston, MA: Springer US, 2001, pp. 135–150. ISBN: 978-0-306-47003-5.

[174] Sergio Segura, Amador Durán, Javier Troya, and Antonio Ruiz-Cortés. "Metamorphic Relation Patterns for Query-Based Systems". In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*. 2019, pp. 24–31. DOI: 10.1109/MET.2019.00012.

[175] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. "A survey on metamorphic testing". In: *IEEE Transactions on Software Engineering* 42.9 (2016), pp. 805–824. DOI: 10.1109/TSE.2016.2532875.

[176] Sergio Segura, José A. Parejo, Javier Troya, and Antonio Ruiz-Cortés. "Metamorphic Testing of RESTful Web APIs". In: *IEEE Transactions on Software Engineering* 44.11 (2018), pp. 1083–1099. DOI: 10.1109/TSE.2017.2764464.

[177] Sergio Segura, Amador Durán Toro, J. Troya, and Antonio Ruiz-Cortés. "A Template-Based Approach to Describing Metamorphic Relations". In: *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)* (2017), pp. 3–9. DOI: 10.1109/MET.2017.3.

[178] Sergio Segura, Dave Towey, Zhi Quan Zhou, and Tsong Yueh Chen. "Metamorphic Testing: Testing the Untestable". In: *IEEE Software* 37.3 (2020), pp. 46–53. DOI: 10.1109/MS.2018.2875968.

[179] Olena Volodymyrivna Semenikhina, MG Drushlyak, Yu A Bondarenko, Svitlana Mykolaivna Kondratiuk, and Iryna Mykolaivna Ionova. "Open Educational Resources as a Trend of Modern Education". In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2019, pp. 779–782. DOI: 10.23919/MIPRO.2019.8756837.

[180] Anuj Kumar Sewani. "Why we use Django rather than Flask in Asset Management System". In: *International Journal for Research in Applied*

Science and Engineering Technology (2021). DOI: 10.22214/ijraset.2021.35756.

[181] Seung Yeob Shin, Fabrizio Pastore, Domenico Bianculli, and Alexandra Baicoianu. "Towards Generating Executable Metamorphic Relations Using Large Language Models". In: *arXiv preprint arXiv:2401.17019* (2024).

[182] Adamya Shyam and Nitin Mukesh. "A Django Based Educational Resource Sharing Website: Shreic". In: *Journal of Scientific Research* 64 (2020), pp. 138–152. DOI: 10.37398/jsr.2020.640134.

[183] Ana-Claudia Sima and Tarcisio Mendes de Farias. "On the Potential of Artificial Intelligence Chatbots for Data Exploration of Federated Bioinformatics Knowledge Graphs". In: *Proc. of the 6th Workshop on Semantic Web Solutions for Large-Scale Biomedical Data Analytics (SeWeBMeDa 2023), co-located with ESWC 2023*. Hersonissos, Greece, 2023, pp. 29–36.

[184] William Simão de Deus and Ellen Francine Barbosa. "A Systematic Mapping of the Classification of Open Educational Resources for Computer Science Education in Digital Sources". In: *IEEE Transactions on Education* 65.3 (2022), pp. 450–460. DOI: 10.1109/TE.2021.3128019.

[185] RANKA Stanković, Daniela Carlucci, OLIVERA Kitanović, Nikola Vulović, and BOJAN Zlatić. "LRMI markup of OER content within the BAEKTEL project". In: *Proc. 6th International Conference on e-Learning (eLearning-2015)*. 2015, pp. 98–103.

[186] Neville A Stanton. "On the reliability and validity of, and training in, ergonomics methods: a challenge revisited". In: *Theoretical Issues in Ergonomics Science* 17.4 (2016), pp. 345–353.

[187] Chang-Ai Sun, Hepeng Dai, Ning Geng, Huai Liu, Tsong Yueh Chen, Peng Wu, Yan Cai, and Jinqiu Wang. "An Interleaving Guided Metamorphic Testing Approach for Concurrent Programs". In: *ACM Transactions on Software Engineering and Methodology* 33.1 (2023), pp. 1–21.

[188] Chang-ai Sun, Hepeng Dai, Huai Liu, and T. Chen. “Feedback-Directed Metamorphic Testing”. In: *ACM Transactions on Software Engineering and Methodology* 32 (2022), pp. 1–34. DOI: 10.1145/3533314.

[189] Chang-Ai Sun, An Fu, Pak-Lok Poon, Xiaoyuan Xie, Huai Liu, and Tsong Yueh Chen. “METRIC$^+$: A Metamorphic Relation Identification Technique Based on Input Plus Output Domains”. In: *IEEE Transactions on Software Engineering* 47.9 (2019), pp. 1764–1785.

[190] Chang-ai Sun, Hui Jin, SiYi Wu, An Fu, ZuoYi Wang, and Wing Kwong Chan. “Identifying metamorphic relations: A data mutation directed approach”. In: *Software: Practice and Experience* 54.3 (2024), pp. 394–418.

[191] Chang-ai Sun, Yiqiang Liu, Zuoyi Wang, and WK Chan. “μMT: a data mutation directed metamorphic relation acquisition methodology”. In: *Proceedings of the 1st International Workshop on Metamorphic Testing*. 2016, pp. 12–18.

[192] Chang-ai Sun, Guan Wang, Baohong Mu, Huai Liu, ZhaoShun Wang, and Tsong Yueh Chen. “Metamorphic testing for web services: Framework and a case study”. In: *2011 IEEE International Conference on Web Services*. IEEE. 2011, pp. 283–290.

[193] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. “Scalability in Perception for Autonomous Driving: Waymo Open Dataset”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2443–2451. DOI: 10.1109/CVPR42600.2020.00252.

[194] The SVL Simulator Team. *Sunsetting SVL Simulator*. Online. Available: `https://github.com/lgsvl/simulator`. [Accessed: Jun. 20, 2024]. SVL Simulator, 2022.

[195] Vivek Thoutam. "A Study On Python Web Application Framework". In: *Journal of Electronics,Computer Networking and Applied Mathematics* (2021). DOI: `10.55529/jecnam.11.48.55`.

[196] Yongqiang Tian, Shiqing Ma, Ming Wen, Yepang Liu, Shing-Chi Cheung, and Xiangyu Zhang. "To what extent do DNN-based image classification models make unreliable inferences?" In: *Empirical Software Engineering* 26.5 (2021), pp. 1–40.

[197] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. "DeepTest: Automated testing of deep-neural-network-driven autonomous cars". In: *Proceedings of the 40th International Conference on Software Engineering*. 2018, pp. 303–314.

[198] Dave Towey and Tsong Yueh Chen. "Teaching software testing skills: Metamorphic testing as vehicle for creativity and effectiveness in software testing". In: *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE. 2015, pp. 161–162.

[199] Dave Towey, Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, and Zhi Quan Zhou. "Metamorphic testing: A new student engagement approach for a new software testing paradigm". In: *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE. 2016, pp. 218–225.

[200] Dave Towey and Matthew Pike. "Not a silver bullet, but a silver lining: Metamorphic marking administration". In: *2021 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2021, pp. 928–935.

[201] Christian Troost, Robert Huber, Andrew R Bell, Hedwig van Delden, Tatiana Filatova, Quang Bao Le, Melvin Lippe, Leila Niamir, J Gareth Pol-

hill, Zhanli Sun, et al. “How to keep it adequate: A protocol for ensuring validity in agent-based simulation”. In: *Environmental Modelling & Software* 159 (2023), p. 105559.

[202] Javier Troya, Sergio Segura, and Antonio Ruiz-Cortés. “Automated inference of likely metamorphic relations for model transformations”. In: *Journal of Systems and Software* 136 (2018), pp. 188–208.

[203] Christos Tsigkanos, Pooja Rani, Sebastian Müller, and Timo Kehrer. “Large Language Models: The Next Frontier for Variable Discovery within Metamorphic Testing?” In: *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2023, pp. 678–682. DOI: 10.1109/SANER56733.2023.00070.

[204] Udacity. *Self-Driving Car Steering Models Evaluation*. https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation. Accessed: 2024-09-11. 2023.

[205] Riley Underwood, Quang-Hung Luu, and Huai Liu. “A metamorphic testing framework and toolkit for modular automated driving systems”. In: *2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)*. IEEE. 2023, pp. 17–24.

[206] UNESCO. “Forum on the impact of Open Courseware for higher education in developing countries”. In: *Final report* (2002).

[207] Roni Utriainen, Markus Pöllänen, and Heikki Liimatainen. “The Safety Potential of Lane Keeping Assistance and Possible Actions to Improve the Potential”. In: *IEEE Transactions on Intelligent Vehicles* 5.4 (2020), pp. 556–564. DOI: 10.1109/TIV.2020.2991962.

[208] Isha Verma, Deepak Kumar, and Rajni Goel. “Implementation and Comparison of Artificial Intelligence Techniques in Software Testing”. In: *2023 6th International Conference on Information Systems and Computer Net-*

works (ISCON). 2023, pp. 1–9. DOI: 10 . 1109 / ISCON57294 . 2023 . 10112041.

[209] Susana M Vieira, Uzay Kaymak, and João MC Sousa. "Cohen's kappa coefficient as a performance measure for feature selection". In: *International Conference on Fuzzy Systems* (2010), pp. 1–8. DOI: 10.1109/FUZZY.2010.5584447.

[210] Massaki Wada, Kang Sup Yoon, and Hideki Hashimoto. "Development of advanced parking assistance system". In: *IEEE Transactions on Industrial Electronics* 50.1 (2003), pp. 4–17. DOI: 10.1109/TIE.2002.807690.

[211] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. "AdvSim: Generating Safety-Critical Scenarios for Self-Driving Vehicles". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9904–9913. DOI: 10.1109/CVPR46437.2021.00978.

[212] Juan Wang, Yongyi Yang, and Bin Xia. "A Simplified Cohen's Kappa for Use in Binary Classification Data Annotation Tasks". In: *IEEE Access* 7 (2019), pp. 164386–164397. DOI: 10.1109/ACCESS.2019.2953104.

[213] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. "Software Testing with Large Language Models: Survey, Landscape, and Vision". In: *IEEE Transactions on Software Engineering* (2024), pp. 1–27. DOI: 10.1109/TSE.2024.3368208.

[214] Wenbo Wang, Y Song, Juan Zhang, and H Deng. "Automatic parking of vehicles: A review of literatures". In: *International Journal of Automotive Technology* 15 (2014), pp. 967–978.

[215] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. "Image quality assessment: From error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[216] M. Warrens. "New interpretations of Cohen's kappa". In: *Journal of Mathematics* 2014 (2014), p. 203907. DOI: 10.1155/2014/203907.

[217] Elaine J. Weyuker. "On Testing Non-Testable Programs". In: *The Computer Journal* 25.4 (1982), pp. 465–470. ISSN: 0010-4620. DOI: 10.1093/comjnl/25.4.465.

[218] Rafal Włodarski, Aneta Poniszewska-Marańda, and Jean-Remy Falleri. "Impact of software development processes on the outcomes of student computing projects: A tale of two universities". In: *Information and Software Technology* 144 (2022), p. 106787. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2021.106787.

[219] Chaohua Wu, Liqun Sun, and Zhi Quan Zhou. "The impact of a dot: Case studies of a noise metamorphic relation pattern". In: 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET). IEEE, 2019, pp. 17–23.

[220] Peng Wu. "Iterative metamorphic testing". In: *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*. Vol. 1. IEEE. 2005, pp. 19–24.

[221] Peng Wu, Xiao Chen Shi, Jiang Jun Tang, Hui Min Lin, and TY Chen. "Metamorphic testing and special case testing: A case study". In: *Journal of Software* 16.7 (2005), pp. 1210–1220.

[222] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. "A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development". In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136. DOI: 10.1109/JAS.2023.123618.

[223] Rachael A Wynne, Vanessa Beanland, and Paul M Salmon. "Systematic review of driving simulator validation studies". In: *Safety Science* 117 (2019), pp. 138–151.

[224] Zhenglong Xiang, Hongrun Wu, and Fei Yu. “A genetic algorithm-based approach for composite metamorphic relations construction”. In: *Information* 10.12 (2019), p. 392.

[225] Lingyun Xiao and Feng Gao. “A comprehensive review of the development of adaptive cruise control systems”. In: *Vehicle System Dynamics* 48.10 (2010), pp. 1167–1192. DOI: 10.1080/00423110903365910.

[226] Chao Xiong, Zhiyi Zhang, Yuqian Zhou, Chen Liu, and Zhiqiu Huang. “IEMT: Inequality-Based Metamorphic Testing for Autonomous Driving Models”. In: *2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. IEEE. 2022, pp. 283–287.

[227] *XPILOT*. Online. Available: https://heyxpeng.com/intelligent/xpilot?parentId=4. [Accessed: June 11, 2023]. Xpeng Motors, 2023.

[228] Jiaxuan Xu, Qi Luo, Kecheng Xu, Xiangquan Xiao, Siyang Yu, Jiangtao Hu, Jinghao Miao, and Jingao Wang. “An Automated Learning-Based Procedure for Large-scale Vehicle Dynamics Modeling on Baidu Apollo Platform”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 5049–5056. DOI: 10.1109/IROS40897.2019.8968102.

[229] Shiyu Yan, Xiaohua Yang, Zhongjian Lu, Meng Li, Helin Gong, and Jie Liu. “Identification Algorithm Framework and Structural Model on Input Pattern of Metamorphic Relations”. In: 2022 9th International Conference on Dependable Systems and Their Applications (DSA). IEEE, 2022, pp. 374–382.

[230] Lan Yang, Yipeng Yang, Guoyuan Wu, Xiangmo Zhao, Shan Fang, Xishun Liao, Runmin Wang, and Mengxiao Zhang. “A systematic review of autonomous emergency braking system: impact factor, technology, and

performance evaluation". In: *Journal of advanced transportation* 2022.1 (2022), p. 1188089.

[231] Sen Yang, Dave Towey, and Zhi Quan Zhou. "Metamorphic exploration of an unsupervised clustering program". In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*. IEEE. 2019, pp. 48–54.

[232] Sen Yang, Dave Towey, Zhi Quan Zhou, and Tsong Yueh Chen. "Preparing Software Quality Assurance Professionals: Metamorphic Exploration for Machine Learning". In: *IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE 2019)*. 2019, pp. 874–880.

[233] Zhen Yang, Song Huang, Changyou Zheng, Xingya Wang, Yang Wang, and Chunyan Xia. "MetaLiDAR: Automated metamorphic testing of LiDAR-based autonomous driving systems". In: *Journal of Software: Evolution and Process* (2023), e2644.

[234] Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. "A comprehensive capability analysis of GPT-3 and GPT-3.5 series models". In: *arXiv preprint arXiv:2303.10420* (2023).

[235] Yuyu Yin, Jiajie Ruan, Youhuizi Li, Yu Li, and Zhijin Pan. "Syntax-based metamorphic relation prediction via the bagging framework". In: *Expert Systems* 39.6 (2022), e12902.

[236] Zhihao Ying, Dave Towey, Anthony Graham Bellotti, Tsong Yueh Chen, and Zhi Quan Zhou. "SFIDMT-ART: A metamorphic group generation method based on Adaptive Random Testing applied to source and follow-up input domains". In: *Information and Software Technology* 175 (2024), p. 107528.

[237] Zhihao Ying, Dave Towey, Tsong Yueh Chen, and Zhi Quan Zhou. "MT-PART: Metamorphic-Testing-Based Adaptive Random Testing Through Partitioning". In: *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2024, pp. 1184–1193. DOI: 10.1109/COMPSAC61105.2024.00158.

[238] Sumendra Yogarayan, Siti Fatimah Abdul Razak, Afizan Azman, and Mohd Fikri Azli Abdullah. "Vehicle to everything (v2x) communications technology for smart mobility in Malaysia: a comprehensive review". In: *Journal of Southwest Jiaotong University* 56.4 (2021).

[239] Shin Yoo and Mark Harman. "Regression testing minimization, selection and prioritization: A survey". In: *Software testing, verification and reliability* 22.2 (2012), pp. 67–120.

[240] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. "A survey of autonomous driving: Common practices and emerging technologies". In: *IEEE Access* 8 (2020), pp. 58443–58469.

[241] Bo Zhang, Hongyu Zhang, Junjie Chen, Dan Hao, and Pablo Moscato. "AutoMR: Automatic Discovery and Cleansing of Numerical Metamorphic Relations". In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2019, pp. 246–246. DOI: 10.1109/ICSME.2019.00036.

[242] Jie Zhang, Junjie Chen, Dan Hao, Yingfei Xiong, Bing Xie, Lu Zhang, and Hong Mei. "Search-Based Inference of Polynomial Metamorphic Relations". In: *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. ASE '14. Vasteras, Sweden: ACM, 2014, pp. 701–712. ISBN: 9781450330138. DOI: 10.1145/2642937.2642994.

[243] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. "DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems". In: *2018 33rd*

IEEE/ACM International Conference on Automated Software Engineering (ASE). 2018, pp. 132–142. DOI: 10.1145/3238147.3238187.

[244] Min-Ling Zhang. "Ml-rbf: RBF Neural Networks for Multi-Label Learning". In: *Neural Processing Letters* 29.2 (Apr. 2009), pp. 61–74. DOI: 10.1007/s11063-009-9095-3.

[245] Pengcheng Zhang, Xuewu Zhou, Patrizio Pelliccione, and Hareton Leung. "RBF-MLMR: A Multi-Label Metamorphic Relation Prediction Approach Using RBF Neural Network". In: *IEEE Access* 5 (2017), pp. 21791–21805. DOI: 10.1109/ACCESS.2017.2758790.

[246] Yifan Zhang, Matthew Pike, Dave Towey, Jia Cheng Han, and Zhi Quan Zhou. "Preparing Future SQA Professionals: An Experience Report of Metamorphic Exploration of an Autonomous Driving System". In: *2022 IEEE Global Engineering Education Conference (EDUCON)*. 2022, pp. 2121–2126. DOI: 10.1109/EDUCON52537.2022.9766791.

[247] Yifan Zhang, Dave Towey, and Matthew Pike. "Automated Metamorphic-Relation Generation with ChatGPT: An Experience Report". In: *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2023, pp. 1780–1785. DOI: 10.1109/COMPSAC57700.2023.00275.

[248] Yifan Zhang, Dave Towey, and Matthew Pike. "Enabling Effective Metamorphic-Relation Generation by Novice Testers: A Pilot Study". In: *2024 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2024, pp. 2398–2403. DOI: 10.1109/COMPSAC61105.2024.00384.

[249] Yifan Zhang, Dave Towey, Matthew Pike, Jia Cheng Han, Zhi Quan Zhou, Chenghao Yin, Qian Wang, and Chen Xie. "Scenario-Driven Metamorphic Testing for Autonomous Driving Simulators". In: *Software Testing, Verification and Reliability* 34.7 (2024), e1892. DOI: 10.1002/stvr.1892.

[250] Yifan Zhang, Dave Towey, Matthew Pike, Jia Cheng Han, George Zhou, Chenghao Yin, Qian Wang, and Chen Xie. "Metamorphic Testing Harness for the Baidu Apollo Perception-Camera Module". In: 2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET). IEEE, 2023, pp. 9–16. DOI: 10.1109/MET59151.2023.00009.

[251] Yifan Zhang, Dave Towey, Matthew Pike, Zhi Quan Zhou, and Tsong Yueh Chen. "Enhancing ADS Testing: An Open Educational Resource for Metamorphic Testing". In: *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2024, pp. 138–147. DOI: 10.1109/COMPSAC61105.2024.00029.

[252] Yuxin Zhang, Cheng Wang, Ruilin Yu, Luyao Wang, Wei Quan, Yang Gao, and Pengfei Li. "The AD4CHE Dataset and Its Application in Typical Congestion Scenarios of Traffic Jam Pilot Systems". In: *IEEE Transactions on Intelligent Vehicles* 8.5 (2023), pp. 3312–3323. DOI: 10.1109/TIV.2023.3260902.

[253] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. "Explainability for large language models: A survey". In: *ACM Transactions on Intelligent Systems and Technology* 15.2 (2024), pp. 1–38.

[254] Zhi Quan Zhou and Liqun Sun. "Metamorphic testing of driverless cars". In: *Communications of the ACM* 62.3 (2019), pp. 61–67.

[255] Zhi Quan Zhou, Liqun Sun, Tsong Yueh Chen, and Dave Towey. "Metamorphic Relations for Enhancing System Understanding and Use". In: *IEEE Transactions on Software Engineering* 46.10 (2020), pp. 1120–1154. DOI: 10.1109/TSE.2018.2876433.

[256] Zhi Quan Zhou, Shaowen Xiang, and Tsong Yueh Chen. "Metamorphic Testing for Software Quality Assessment: A Study of Search Engines".

In: *IEEE Transactions on Software Engineering* 42.3 (2016), pp. 264–284. DOI: 10.1109/TSE.2015.2478001.

[257] Zhi Quan Zhou, Junting Zhu, Tsong Yueh Chen, and Dave Towey. “In-Place Metamorphic Testing and Exploration”. In: *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*. 2022, pp. 1–6. DOI: 10.1145/3524846.3527334.

[258] Mikel Zorrilla, Njål Borch, François Daoust, Alexander Erk, Julián Flórez, and Alberto Lafuente. “A web-based distributed architecture for multi-device adaptation in media applications”. In: *Personal and Ubiquitous Computing* 19 (2015), pp. 803–820.

Appendices

Appendix A

Example Conversations to Enhance Prompts with ChatGPT

1. **User: Give me five metamorphic relations (MRs) for testing the parking module of autonomous driving systems (ADSs)**

   ChatGPT: Sure, here are five MRs for testing the parking module of autonomous driving systems:

   - **MR1: Change in starting position** - This MR involves changing the starting position of the car before parking. The expected behaviour is that the car should be able to detect the new position and park accordingly.
   - **MR2: Change in parking spot size** - This MR involves changing the size of the parking spot. The expected behaviour is that the car should be able to adjust its parking manoeuvre to fit in the new spot size.
   - **MR3: Change in parking lot layout** - This MR involves changing the layout of the parking lot. The expected behaviour is that the car should be able to detect the changes and adjust its parking manoeuvre

accordingly.

- **MR4: Change in obstacle position** - This MR involves changing the position of obstacles in the parking lot. The expected behaviour is that the car should be able to detect the new position of the obstacles and park accordingly.

- **MR5: Change in weather conditions** - This MR involves changing the weather conditions, such as rain or snow. The expected behaviour is that the car should be able to adjust its parking manoeuvre to account for the slippery surface or reduced visibility.

2. **User: You should include the input relations and output relations for each MR.**

   ChatGPT: Sure! Here are the input relations, output relations, and expected behaviours for each of the five MRs I listed previously for testing the parking module of autonomous driving systems:

   - **MR1: Change in starting position**
     - **Input relation:** Starting position of the car
     - **Output relation:** Final parking position of the car
     - **Expected behaviour:** The car should detect the new starting position and park accordingly, resulting in a final parking position that is appropriate for the new starting position.
   - **MR2: Change in parking spot size**
     - **Input relation:** Size of the parking spot
     - **Output relation:** Final parking position of the car
     - **Expected behaviour:** The car should adjust its parking manoeuvre to fit in the new spot size, resulting in a final parking position that is appropriate for the new spot size.
   - **MR3: Change in parking lot layout**

  - **Input relation:** Layout of the parking lot
  - **Output relation:** Final parking position of the car
  - **Expected behaviour:** The car should detect the changes in the parking lot layout and adjust its parking manoeuvre accordingly, resulting in a final parking position that is appropriate for the new layout.

- **MR4: Change in obstacle position**
  - **Input relation:** Position of obstacles in the parking lot
  - **Output relation:** Final parking position of the car
  - **Expected behaviour:** The car should detect the new position of the obstacles and park accordingly, resulting in a final parking position that is appropriate for the new obstacle position.

- **MR5: Change in weather conditions**
  - **Input relation:** Weather conditions
  - **Output relation:** Final parking position of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the slippery surface or reduced visibility due to the changed weather conditions, resulting in a final parking position that is appropriate for the new conditions.

3. **User: The input relation and output relation for each MR are vague. State the changes among inputs and the relations among outputs more clearly.**

   ChatGPT: Sure! Here are the more specific input relations and output relations for each MR for testing the parking module of autonomous driving systems:

   - **MR1: Change in starting position**

- **Input relation:** The initial position (x,y) of the car before parking
  - **Output relation:** The final parking position (x',y') of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the change in the initial position and park accordingly, resulting in a final parking position that is appropriate for the new starting position.
- **MR2: Change in parking spot size**
  - **Input relation:** The size of the original parking spot (length, width)
  - **Output relation:** The final parking position (x',y') of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to fit in the new spot size, resulting in a final parking position that is appropriate for the new spot size.
- **MR3: Change in parking lot layout**
  - **Input relation:** The original and new parking lot layouts
  - **Output relation:** The final parking position (x',y') of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the changes in the parking lot layout and park accordingly, resulting in a final parking position that is appropriate for the new layout.
- **MR4: Change in obstacle position**
  - **Input relation:** The original and new position (x,y) of obstacles in the parking lot
  - **Output relation:** The final parking position (x',y') of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the new position of the obstacles and park

accordingly, resulting in a final parking position that is appropriate for the new obstacle position.

- **MR5: Change in weather conditions**
  - **Input relation:** The original and new weather conditions, such as rain or snow
  - **Output relation:** The final parking position (x',y') of the car
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the slippery surface or reduced visibility due to the changed weather conditions, resulting in a final parking position that is appropriate for the new conditions.

4. **User: the expression "resulting in a final parking position that is appropriate for the new conditions" is vague. Make it clearer for each MR**

   ChatGPT: Certainly! Here are the updated expected behaviours for each MR that make the final parking position more specific:

   - **MR1: Change in starting position** - The car should adjust its parking manoeuvre to account for the change in the initial position and park accordingly, resulting in a final parking position that is appropriate for the new starting position. For example, if the starting position is closer to the edge of the parking lot, the final parking position should be adjusted to avoid hitting any objects or pedestrians.
   - **MR2: Change in parking spot size** - The car should adjust its parking manoeuvre to fit in the new spot size, resulting in a final parking position that is appropriate for the new spot size. For example, if the parking spot is larger, the car should park closer to the centre of the spot to avoid leaving too much space on either side.
   - **MR3: Change in parking lot layout** - The car should adjust its parking manoeuvre to account for the changes in the parking lot lay-

out and park accordingly, resulting in a final parking position that is appropriate for the new layout. For example, if the parking lot layout changes to include diagonal parking spaces, the car should adjust its parking angle accordingly.

- **MR4: Change in obstacle position** - The car should adjust its parking manoeuvre to account for the new position of the obstacles and park accordingly, resulting in a final parking position that is appropriate for the new obstacle position. For example, if an obstacle is now positioned closer to the parking space, the car should park farther away from the obstacle to avoid hitting it.
- **MR5: Change in weather conditions** - The car should adjust its parking manoeuvre to account for the slippery surface or reduced visibility due to the changed weather conditions, resulting in a final parking position that is appropriate for the new conditions. For example, if the surface is slippery due to rain, the car should park more slowly and cautiously to avoid slipping or sliding.

5. **User: You miss the input and output relations**

   ChatGPT: I apologize for the confusion. Here are the updated input and output relations for each MR, along with the clearer expected behaviours:

   - **MR1: Change in starting position**
     - **Input relation:** Original starting position (x,y) and new starting position (x',y')
     - **Output relation:** Final parking position (x",y")
     - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the change in the initial position and park accordingly, resulting in a final parking position (x",y") that is appropriate for the new starting position. For example, if the starting position is closer to the edge of the parking lot, the final

parking position should be adjusted to avoid hitting any objects or pedestrians.

- **MR2: Change in parking spot size**
  - **Input relation:** Original parking spot size (length, width) and new parking spot size (length', width')
  - **Output relation:** Final parking position (x",y")
  - **Expected behaviour:** The car should adjust its parking manoeuvre to fit in the new spot size, resulting in a final parking position (x",y") that is appropriate for the new spot size. For example, if the parking spot is larger, the car should park closer to the centre of the spot to avoid leaving too much space on either side.
- **MR3: Change in parking lot layout**
  - **Input relation:** Original and new parking lot layouts
  - **Output relation:** Final parking position (x",y")
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the changes in the parking lot layout and park accordingly, resulting in a final parking position (x",y") that is appropriate for the new layout. For example, if the parking lot layout changes to include diagonal parking spaces, the car should adjust its parking angle accordingly.
- **MR4: Change in obstacle position**
  - **Input relation:** Original and new position (x,y) of obstacles in the parking lot
  - **Output relation:** Final parking position (x",y")
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the new position of the obstacles and park accordingly, resulting in a final parking position (x",y") that is

appropriate for the new obstacle position. For example, if an obstacle is now positioned closer to the parking space, the car should park farther away from the obstacle to avoid hitting it.

- **MR5: Change in weather conditions**
  - **Input relation:** Original and new weather conditions, such as rain or snow
  - **Output relation:** Final parking position (x”,y”)
  - **Expected behaviour:** The car should adjust its parking manoeuvre to account for the slippery surface or reduced visibility due to the changed weather conditions, resulting in a final parking position (x”,y”) that is appropriate for the new conditions. For example, if the surface is slippery due to rain, the car should park more slowly and cautiously to avoid slipping or sliding.

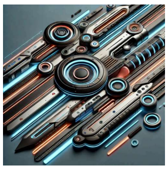
Appendix B

Metamorphic Relations Derived by the GPT-MR Generator for Testing the CARLA Simulator

You said:

I want to test the spawn function of the simulator. Specifically, I want to test its capability of preventing the actors from spawning at the location that would cause collisions. I want you to generate MRs for me.

ChatGPT said:



To test the spawn function of the ADS simulator, focusing on preventing actors from spawning at locations that would cause collisions, we can generate Metamorphic Relations (MRs) using the provided Metamorphic Relation Patterns (MRPs).

Metamorphic Relation Pattern: Impact Amplification

MR1: Amplified Actor Density

- **Scenario**: Start with a scenario where multiple actors are spawned in a region with no initial collisions.
- **Amplified Scenario**: Increase the density of actors by spawning additional actors in the same region.
- **Expected Outcome**: The simulator should prevent actors from spawning in locations that would result in collisions. The system should either not spawn the additional actors or adjust their spawn points to avoid collisions.
- **Rationale**: This tests the simulator's ability to handle higher densities and ensure collision-free spawning.

MR2: Amplified Complexity

- **Scenario**: Begin with a scenario where actors are spawned around static obstacles (e.g., buildings, barriers) without causing collisions.
- **Amplified Scenario**: Increase the complexity by adding more static obstacles in close proximity to the original spawn points.
- **Expected Outcome**: The simulator should continue to prevent actors from spawning at locations that would cause collisions, despite the increased obstacle density.
- **Rationale**: This ensures the spawn function can handle more complex environments and still avoid collisions.

Metamorphic Relation Pattern: Marginal Ensemble Revision

MR3: Slight Positional Shifts

- **Scenario**: Define a baseline scenario where actors are spawned without causing collisions.
- **Marginal Revision Scenario**: Slightly shift the spawn points of actors by small margins (e.g., a few centimeters or inches).

- **Expected Outcome**: The system should maintain collision-free spawning, as these minor positional changes should not lead to collisions.
- **Rationale**: This tests the precision and robustness of the spawn function in handling small adjustments without resulting in collisions.

MR4: Minor Orientation Changes

- **Scenario**: Start with a scenario where actors are spawned without causing collisions.
- **Marginal Revision Scenario**: Adjust the orientations of the actors by small angles (e.g., 5-10 degrees).
- **Expected Outcome**: The simulator should still prevent any collisions upon spawning, as minor orientation changes should not affect the collision avoidance mechanism.
- **Rationale**: This verifies that the spawn function consistently avoids collisions despite minor changes in actor orientation.

By applying these MRs, you can thoroughly test the spawn function of the ADS simulator to ensure it effectively prevents actors from spawning at locations that would cause collisions, even under varying conditions and complexities.

You said:

I want to test the Autopilot mode of the CARLA simulator. I want to test its capability through a source scenario that EGO vehicle prepares to pass a junction when the traffic light is red.

ChatGPT said:

To test the Autopilot mode of the CARLA simulator, particularly focusing on the EGO vehicle's behavior when preparing to pass a junction with a red traffic light, we can generate Metamorphic Relations (MRs) using the provided Metamorphic Relation Patterns (MRPs).

Metamorphic Relation Pattern: Impact Amplification

MR1: Amplified Traffic Density

- **Source Scenario**: The EGO vehicle approaches a junction with a red traffic light, with light traffic (e.g., a few vehicles) in the vicinity.
- **Amplified Scenario**: Increase the traffic density by adding more vehicles around the junction.
- **Expected Outcome**: The EGO vehicle should still recognize the red traffic light and stop, regardless of the increased traffic density.
- **Rationale**: This tests the Autopilot's capability to handle increased traffic complexity while maintaining compliance with traffic signals.

MR2: Amplified Traffic Signal Complexity

- **Source Scenario**: The EGO vehicle approaches a simple junction with a standard red traffic light.
- **Amplified Scenario**: Introduce additional traffic control signals (e.g., pedestrian crossings with their own signals, or additional lanes with separate signals) at the junction.
- **Expected Outcome**: The EGO vehicle should correctly interpret the red traffic light for its lane and stop, despite the increased complexity of traffic signals.
- **Rationale**: This ensures the Autopilot can handle more complex traffic signal scenarios without misinterpreting signals.

Metamorphic Relation Pattern: Marginal Ensemble Revision

MR3: Slight Positional Shifts of EGO Vehicle

- **Source Scenario**: The EGO vehicle is positioned at a specific distance from the junction with a red traffic light.
- **Marginal Revision Scenario**: Slightly adjust the starting position of the EGO vehicle (e.g., a few meters forward or backward).
- **Expected Outcome**: The EGO vehicle should still recognize the red traffic light and stop, regardless of its slightly altered starting position.

- **Rationale**: This tests the robustness of the Autopilot in maintaining correct behavior with minor positional variations.

MR4: Minor Timing Adjustments

- **Source Scenario**: The EGO vehicle approaches the junction as the traffic light is red.
- **Marginal Revision Scenario**: Introduce slight timing variations to the traffic light cycle (e.g., the red light duration is slightly extended or shortened).
- **Expected Outcome**: The EGO vehicle should continue to comply with the traffic light, stopping when it is red, despite minor timing changes.
- **Rationale**: This verifies that the Autopilot consistently reacts appropriately to red traffic lights even with small changes in the signal timing.

MR5: Minor Environmental Changes

- **Source Scenario**: The EGO vehicle approaches a junction with a red traffic light under standard environmental conditions.
- **Marginal Revision Scenario**: Introduce minor environmental changes such as slight variations in weather (e.g., light rain or fog) or time of day (e.g., dawn or dusk).
- **Expected Outcome**: The EGO vehicle should still recognize and stop for the red traffic light, regardless of these minor environmental changes.
- **Rationale**: This tests the Autopilot's ability to handle slight environmental variations while maintaining correct behavior at traffic signals.

By applying these MRs, you can comprehensively test the Autopilot mode of the CARLA simulator, ensuring that the EGO vehicle correctly interprets and responds to red traffic lights at junctions under varying conditions and complexities.

You said:

I want to test the built-in simulation criteria in the simulator. These evaluation criteria are used to analyze if a scenario was completed successfully or failed.

ChatGPT said:

To test the built-in simulation criteria in the CARLA simulator, specifically focusing on evaluating if a scenario was completed successfully or failed, we can generate Metamorphic Relations (MRs) using the provided Metamorphic Relation Patterns (MRPs). These MRs will help validate the robustness and accuracy of the evaluation criteria under different conditions.

Metamorphic Relation Pattern: Impact Amplification

MR1: Amplified Traffic Complexity

- **Source Scenario**: A basic scenario where the EGO vehicle completes a route with minimal traffic and passes through checkpoints without incidents.
- **Amplified Scenario**: Increase the traffic complexity by adding more vehicles, pedestrians, and obstacles along the route.
- **Expected Outcome**: The evaluation criteria should accurately assess whether the scenario was completed successfully or failed, considering the increased traffic complexity. Successful completion should still be recognized if the EGO vehicle navigates through without collisions or rule violations.
- **Rationale**: This tests the criteria's ability to handle and accurately evaluate more complex traffic scenarios.

MR2: Amplified Environmental Challenges

- **Source Scenario**: A scenario where the EGO vehicle drives through clear weather conditions and straightforward road conditions.
- **Amplified Scenario**: Introduce challenging environmental conditions such as heavy rain, fog, or night-time driving.
- **Expected Outcome**: The evaluation criteria should still correctly determine if the scenario was completed successfully or failed, considering the amplified environmental challenges. The criteria should recognize successful completion if the EGO vehicle adapts to the challenging conditions without incidents.
- **Rationale**: This ensures the evaluation criteria are robust and can accurately assess performance under different environmental conditions.

Metamorphic Relation Pattern: Marginal Ensemble Revision

MR3: Slight Positional Adjustments

- **Source Scenario**: A scenario where the EGO vehicle starts and ends at specific checkpoints and completes the route without incidents.
- **Marginal Revision Scenario**: Slightly adjust the starting and ending positions of the EGO vehicle by a few meters.
- **Expected Outcome**: The evaluation criteria should still correctly assess the scenario as successful or failed, based on the EGO vehicle's performance, despite the slight positional adjustments.
- **Rationale**: This tests the evaluation criteria's ability to handle minor positional changes without affecting the assessment of scenario success.

MR4: Minor Route Variations

- **Source Scenario**: A scenario where the EGO vehicle follows a predefined route and passes all checkpoints successfully.
- **Marginal Revision Scenario**: Introduce minor variations in the route, such as slight detours or alternative paths that still lead to the same checkpoints.
- **Expected Outcome**: The evaluation criteria should recognize the scenario as successful if the EGO vehicle follows the varied route correctly and reaches all checkpoints without incidents.
- **Rationale**: This verifies that the evaluation criteria are flexible and can accurately assess different but valid route variations.

MR5: Subtle Behavioral Changes

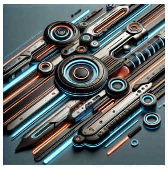
- **Source Scenario**: A scenario where the EGO vehicle drives at a constant speed and adheres to all traffic rules, completing the route successfully.
- **Marginal Revision Scenario**: Introduce subtle changes in the EGO vehicle's behavior, such as slight variations in speed or minor adjustments in lane positioning, without violating traffic rules.
- **Expected Outcome**: The evaluation criteria should still assess the scenario as successful if the EGO vehicle completes the route correctly, despite the subtle behavioral changes.
- **Rationale**: This ensures the evaluation criteria are robust and can accurately evaluate the EGO vehicle's performance even with minor behavioral variations.

By applying these MRs, you can thoroughly test the built-in simulation criteria in the CARLA simulator to ensure they accurately and consistently evaluate whether scenarios are completed successfully or failed under varying conditions and complexities.

You said:

I want to test the consistency of simulation outputs of the same scenario

ChatGPT said:



To test the consistency of simulation outputs for the same scenario in the CARLA simulator, we can generate Metamorphic Relations (MRs) using the provided Metamorphic Relation Patterns (MRPs). These MRs will help verify that the simulator produces consistent results when the same scenario is run multiple times under similar conditions.

Metamorphic Relation Pattern: Impact Amplification

MR1: Repeated Runs with Amplified Factors

- **Source Scenario**: A baseline scenario where the EGO vehicle drives through a predefined route with standard traffic and environmental conditions.
- **Amplified Scenario**: Increase the number of runs for the same scenario under identical conditions (e.g., 10 repeated runs).
- **Expected Outcome**: The simulation outputs should be consistent across all runs, showing similar EGO vehicle behaviors, routes, and interactions with other entities.
- **Rationale**: This tests the simulator's ability to produce consistent outputs under repeated runs with amplified repetition factors.

Metamorphic Relation Pattern: Marginal Ensemble Revision

MR2: Slight Variations in Initial Conditions

- **Source Scenario**: A scenario where the EGO vehicle starts from a specific position and drives through a route with predefined traffic and environmental conditions.
- **Marginal Revision Scenario**: Introduce slight variations in the initial conditions, such as small changes in the starting position (e.g., a few centimeters) or slight timing differences (e.g., starting the scenario a few milliseconds later).
- **Expected Outcome**: The simulation outputs should remain consistent, with the EGO vehicle exhibiting similar behaviors and outcomes despite the slight initial variations.
- **Rationale**: This tests the robustness of the simulator in maintaining consistent outputs with minor changes in initial conditions.

MR3: Minor Environmental Changes

- **Source Scenario**: A scenario where the EGO vehicle drives through a route with standard environmental conditions (e.g., clear weather, daylight).
- **Marginal Revision Scenario**: Introduce minor changes in the environment, such as slight variations in weather (e.g., light fog) or time of day (e.g., early morning light).

- **Expected Outcome**: The simulation outputs should be consistent, showing similar EGO vehicle behaviors and outcomes despite the minor environmental changes.
- **Rationale**: This ensures the simulator's consistency in producing outputs under slightly different environmental conditions.

MR4: Small Adjustments in Traffic Density

- **Source Scenario**: A scenario where the EGO vehicle drives through a route with a specific number of other vehicles and pedestrians.
- **Marginal Revision Scenario**: Introduce small adjustments in the traffic density, such as adding or removing one vehicle or pedestrian.
- **Expected Outcome**: The simulation outputs should remain consistent, with the EGO vehicle's behavior and interactions being similar despite the small changes in traffic density.
- **Rationale**: This verifies the simulator's ability to maintain consistent outputs with minor variations in traffic conditions.

MR5: Subtle Behavioral Tweaks

- **Source Scenario**: A scenario where the EGO vehicle follows a specific set of driving behaviors (e.g., speed, lane positioning).
- **Marginal Revision Scenario**: Introduce subtle tweaks to the EGO vehicle's behavior, such as slight variations in speed or lane positioning.
- **Expected Outcome**: The simulation outputs should be consistent, showing similar overall behaviors and outcomes for the EGO vehicle despite the subtle behavioral tweaks.
- **Rationale**: This tests the consistency of the simulator in producing outputs when there are minor variations in the EGO vehicle's behavior.

By applying these MRs, you can effectively test the consistency of the simulation outputs for the same scenario in the CARLA simulator, ensuring that the results are reliable and repeatable under varying but controlled conditions.

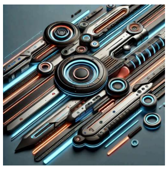
Appendix C

Metamorphic Relations Derived by the GPT-MR Generator for Systems in Various Domains

You said:

I want to create MRs to test a search engine

ChatGPT said:



To generate Metamorphic Relations (MRs) for testing a search engine, we can adapt the two Metamorphic Relation Patterns (MRPs) provided:

1. MRP_ImpactAmplification for Search Engine Testing:

This MRP focuses on amplifying influential factors that should significantly affect the search results. In a search engine, the following can be considered influential factors:

- **Query Length:** Extending or shortening the search query.
- **Query Specificity:** Adding more specific terms or broader terms to the query.
- **Filters (e.g., time, location):** Applying or modifying filters that should have a noticeable impact on the results.

MR1: Amplify query specificity

- **Original Input:** Query: "autonomous driving cars"
- **Transformed Input:** Amplify specificity by adding details: "autonomous driving cars with LIDAR sensors"
- **Expected Outcome:** The results should be more specific, aligning closely with the additional term "LIDAR sensors." The search engine should now prioritize results related to LIDAR sensors in autonomous cars, compared to the general query.

MR2: Amplify filtering criteria

- **Original Input:** Query: "electric cars"
- **Transformed Input:** Add time filter (e.g., "last 6 months")
- **Expected Outcome:** The search results should only include results published in the past 6 months. The results should now focus on more recent and relevant information about electric cars.

MR3: Amplify query length

- **Original Input:** Query: "best ADS simulators"
- **Transformed Input:** Extend the query: "best ADS simulators for highway driving"
- **Expected Outcome:** The search results should now focus specifically on ADS simulators designed for highway driving, providing more relevant results with the extended query.

2. MRP_MarginalEnsembleRevision for Search Engine Testing:

This MRP tests non-influential factors that should not significantly impact search results. In a search engine context, these factors could include:

- **Minor variations in query case or punctuation.**
- **Word reordering that does not change the meaning of the query.**

MR4: Alter case in query

- **Original Input:** Query: "Autonomous driving"
- **Transformed Input:** Change case: "autonomous DRIVING"
- **Expected Outcome:** The search results should remain the same, as query case should not impact the outcome.

MR5: Slight reorder of query terms

- **Original Input:** Query: "electric car prices"
- **Transformed Input:** Reorder query: "car electric prices"
- **Expected Outcome:** The search results should remain unchanged, as the reordering does not alter the intended meaning of the query.

MR6: Add inconsequential punctuation

- **Original Input:** Query: "self driving cars"
- **Transformed Input:** Add punctuation: "self-driving cars"
- **Expected Outcome:** The results should remain consistent, as the added punctuation does not change the meaning of the search query.

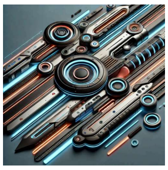
These MRs can help validate that the search engine responds correctly to both significant and minor changes in query input, ensuring robustness in the search functionality.

Messages beyond this point are only visible to you

You said:

I want to create MRs to test the route function of a navigation software

ChatGPT said:



To generate Metamorphic Relations (MRs) for testing the route function of a navigation software, we will use the two Metamorphic Relation Patterns (MRPs) you provided: **Impact Amplification** and **Marginal Ensemble Revision**.

In this context, we can identify **influential factors** (such as starting point, destination, traffic conditions, road closures) and **non-influential factors** (such as minor GPS errors or slight variations in initial conditions that shouldn't change the route).

Here are some generated MRs:

---

1. MR 1: Impact Amplification (Traffic Conditions)

Scenario:

- The navigation software calculates the shortest route from Point A to Point B in normal traffic conditions.

Metamorphic Relation:

- Amplify traffic conditions by adding severe traffic congestion on specific sections of the route. The system should recalculate a different, potentially longer but faster route. This change verifies that the system can adapt routes based on significant traffic variations.

---

2. MR 2: Marginal Ensemble Revision (Minor Position Shift)

Scenario:

- The navigation software generates a route from Point A to Point B.

Metamorphic Relation:

- Slightly shift the starting point by a few meters (e.g., 5 meters in any direction). The system should still generate the same route, as this minor change should not influence the overall route. This MR verifies that the system can handle small GPS inaccuracies without deviating from the expected route.

---

3. MR 3: Impact Amplification (Road Closures)

Scenario:

- The system calculates a route from Point A to Point B under normal road conditions.

Metamorphic Relation:

- Amplify the influential factor by closing one or more roads on the original route. The system should recalculate an alternative route to avoid the closed roads. This MR tests the system’s ability to react to road closures and suggest feasible alternatives.

---

4. MR 4: Marginal Ensemble Revision (Route Timing)

Scenario:

- The navigation software calculates a route that takes 30 minutes to travel from Point A to Point B with moderate traffic conditions.

Metamorphic Relation:

- Marginally adjust the speed limits on roads or minorly alter the timing of traffic lights without changing the overall traffic patterns. The resulting route should still be the same, with minor differences in the estimated travel time. This MR ensures that the route function is robust to small changes in speed limits or timing assumptions.

---

5. MR 5: Impact Amplification (Alternate Routes with Toll Roads)

Scenario:

- The system calculates a route from Point A to Point B, preferring a route without tolls.

Metamorphic Relation:

- Amplify the availability of toll roads by adding multiple fast toll roads along the route. The system should offer a new route that includes the toll roads if they significantly shorten the travel time. This MR tests the system’s ability to weigh time versus cost trade-offs.

Summary:

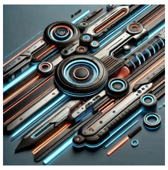
- **Impact Amplification MRs** focus on major input factors like traffic, road closures, and toll roads, testing if the system recalculates optimal routes in response to these changes.
- **Marginal Ensemble Revision MRs** deal with small, negligible input changes like minor shifts in GPS position or speed limits, ensuring these have no significant impact on the route generated.

These MRs can be applied iteratively to test the robustness of the route calculation function of the navigation software.

You said:

I want to create MRs to test an email app

ChatGPT said:



To generate Metamorphic Relations (MRs) for testing an email app, I'll adapt the provided Metamorphic Relation Patterns (MRPs) to suit the functionalities of such an application. These MRs will focus on verifying how the email app responds to influential factors, such as sending/receiving emails, and non-influential factors, such as minor layout or content changes, without affecting functionality.

Here's how I will apply each MRP to create MRs for the email app:

1. MRP_ImpactAmplification

Objective: Amplify influential factors related to email sending/receiving functionality and observe amplified or consistent effects.

MR1: Amplified Bulk Email Sending

- *Scenario*: The user sends 10 emails individually to various recipients.
- *Modification (Amplification)*: The user sends 100 emails instead.
- *Expected Behavior*: The system should still send all 100 emails, and the email queue should be processed successfully without crashes or delays.
- *Rationale*: Testing whether an increase in the number of emails amplifies processing time or reveals performance issues.

MR2: Increased Email Attachment Size

- *Scenario*: The user sends an email with a 1 MB attachment.
- *Modification (Amplification)*: The user sends the same email with a 50 MB attachment.
- *Expected Behavior*: The system should either send the email if it is under the allowable limit or return an appropriate error if it exceeds it.
- *Rationale*: Verifies whether increasing the attachment size affects the app's ability to handle large files or provides correct feedback.

2. MRP_MarginalEnsembleRevision

Objective: Introduce minor changes in non-influential factors to verify the consistency of results.

MR3: Small Change in Email Subject

- *Scenario*: The user sends an email with the subject line "Meeting at 10 AM".

- *Modification (Marginal Revision)*: The subject is changed to "Meeting at 10 A.M." (minor punctuation revision).
- *Expected Behavior*: The email should still be sent successfully, and the system should not treat the revised subject as significantly different in functionality.
- *Rationale*: Ensures that minor textual changes do not affect the ability to send or classify the email.

MR4: Slight Change in Recipient Order

- *Scenario*: The user sends an email to three recipients: `alice@example.com`, `bob@example.com`, `charlie@example.com`.
- *Modification (Marginal Revision)*: The order of recipients is changed to `charlie@example.com`, `alice@example.com`, `bob@example.com`.
- *Expected Behavior*: The email should still be sent to all recipients without any change in outcome.
- *Rationale*: Verifies that the order of recipients does not affect delivery, ensuring consistent functionality.

3. Amplifying Existing MRs

Objective: Apply MRP_ImpactAmplification again on previously generated MRs to test boundary or extreme behavior.

MR5: Further Amplification on Bulk Email Sending

- *Scenario (from MR1)*: The user sends 100 emails.
- *Modification (Further Amplification)*: The user sends 1000 emails.
- *Expected Behavior*: The system should handle this large volume or provide clear feedback if a limit is reached, verifying the upper bounds of the app's capacity.
- *Rationale*: Tests system robustness and scalability by pushing email volume to an extreme.

MR6: Further Amplification on Recipient List

- *Scenario (from MR4)*: The user sends an email to three recipients.
- *Modification (Further Amplification)*: The user sends the email to 100 recipients.
- *Expected Behavior*: The system should handle the large recipient list efficiently without performance degradation.
- *Rationale*: Tests the system's ability to handle emails with many recipients and validates performance under high load.

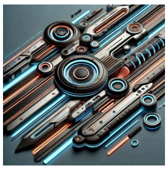
These MRs aim to explore both the boundaries of influential factors (e.g., large email volumes, big attachments) and the stability under minor, non-influential changes (e.g., subject tweaks, recipient order). These tests will help ensure the email app functions correctly under varied conditions.

Messages beyond this point are only visible to you

You said:

I want MRs for an ordering system

ChatGPT said:



For an ordering system, I'll apply the two Metamorphic Relation Patterns (MRPs) – **MRP_ImpactAmplification** and **MRP_MarginalEnsembleRevision** – to create meaningful MRs.

Influential Factors

In an ordering system, influential factors could include:

- **Order quantity** (influences total cost, inventory levels)
- **Item availability** (impacts ability to fulfill order)
- **Customer priority** (affects processing speed and prioritization)

Non-Influential Factors

Non-influential factors may include:

- **Time of order (within non-peak hours)**
- **Minor variations in user input format (e.g., different capitalization, spaces)**
- **Small changes in delivery time window (if non-critical)**

Metamorphic Relations (MRs)

1. MRP_ImpactAmplification

MR1: Amplify Order Quantity

- **Original Scenario:** Place an order with 10 units of an item.
- **Amplified Scenario:** Increase the order to 100 units of the same item.
- **Expected Outcome:** The system should show an amplified increase in total price and inventory update. If inventory is insufficient, the system should prompt a warning or partial order.

Rationale: This MR ensures that changes in order quantity lead to proportional or consistent changes in total cost and inventory levels, helping verify if the system appropriately handles large orders.

MR2: Amplify Customer Priority

- **Original Scenario:** Place an order with normal customer priority.
- **Amplified Scenario:** Change the customer priority to "urgent" or "VIP."

- **Expected Outcome:** The system should prioritize this order, potentially reflecting faster processing times or higher resource allocation.

Rationale: Amplifying the customer priority should result in observable behavior changes, confirming whether the system responds appropriately to different priority levels.

---

2. MRP_MarginalEnsembleRevision

MR3: Minor Variations in Order Time

- **Original Scenario:** Place an order at 3:00 PM.
- **Marginally Revised Scenario:** Place the same order at 3:05 PM.
- **Expected Outcome:** The system's processing and output (confirmation, price, item availability) should remain unchanged.

Rationale: A slight change in order time within non-peak hours should not affect the outcome, verifying that non-influential time changes do not alter the result.

MR4: Minor Format Changes in Customer Information

- **Original Scenario:** Input customer name as "John Doe."
- **Marginally Revised Scenario:** Input customer name as "JOHN DOE" (all caps).
- **Expected Outcome:** The system should process the order without error, and the outcome (confirmation, order details) should remain unchanged.

Rationale: Minor variations in capitalization or formatting should not impact the system’s ability to correctly process and store the customer information.

These MRs use the key principles of impact amplification and marginal ensemble revision to test the robustness of the ordering system in handling both critical and non-critical factors.

Messages beyond this point are only visible to you