Human Interaction and Emerging Technologies (IHIET-AI 2024), Vol. 120, 2024, 195–206
https://doi.org/10.54941/ahfe1004569

Exploring Metamorphic Testing for SLFs With User Interactions

Marco Stang, Luca Seidel, Veljko Vučinić, and Eric Sax

Institut fuer Technik der Informationsverarbeitung (ITIV), Karlsruher Institute for Technology (KIT), Karlsruhe, Germany

ABSTRACT

This paper presents a novel approach to testing SLFs (SLFs) in adaptive systems using metamorphic testing (MT). Recognizing the challenge of verifying software that learns from and adapts to user behavior without a definitive oracle, we propose specific metamorphic relations (MRs) as the basis for our testing framework. These MRs are crafted to evaluate the SLFs' capacity to tailor user experiences by adapting to individual behaviors, transitioning user patterns and environmental changes, and managing data anomalies. We demonstrate our testing methodology through a two-stage process, utilizing synthetic data generated by the CAGEN* tool to simulate realistic user interactions and environmental factors. Principal component analysis (PCA) is employed to visualize the effectiveness of SLFs in adhering to the identified MRs. Our findings highlight proficiency in personalization by differentiating between user behaviors. The paper emphasizes the effectiveness of MT in enhancing the development of intelligent, user-centric systems and suggests directions for future research to extend these testing methods to more complex scenarios and diverse SLF architectures.

Keywords: Metamorphic testing, Machine learning, Software testing, Adaptive systems

INTRODUCTION AND MOTIVATION

The field of SLFs is a dynamic facet of modern technology with an increasing presence in various applications (Goodfellow et al., 2016). SLFs utilize data-driven methods such as supervised, unsupervised, and reinforcement learning (Li 2018). In the automotive domain, SLFs are relevant for real-time decision-making, as they adapt to dynamic scenarios by collecting extensive data from sensors. Likewise, SLFs with user interaction assert their influence in the automotive industry, as they observe driver behavior and recognize user-specific interactions with the system. Beyond simply processing data, these functions proactively interact with user inputs, adapting to individual preferences in features like seat heating or gesture control (Stang et al., 2022) to enhance the overall driving experience. The growing integration and interaction of SLFs (Bertolini et al., 2021) underscore the importance of conducting research and refining testing methodologies to ensure their reliability and effectiveness.

*COntext and ACtion GENeration – tool for the generation of synthetic datasets.

Effective test methods for validating SLFs with user interaction are necessary to meet the increasing demand for trustworthy and reliable SLFs. Traditional software testing strategies (Ammann and Offutt, 2016) are inadequate for these functions, as they must contend with unpredictability resulting from their adaptive nature and the diversity of user behaviors. Conventional test cases with fixed specifications cannot predict or verify the behavior of these adaptive algorithms. Adding to the testing challenge, every user introduces unique actions, and traditional approaches fail to capture the vast array of potential interactions. As SLFs continuously evolve with each user, new testing methodologies must adapt to the functions. This paper presents a solution to address the test-oracle problem by leveraging metamorphic testing (Chen et al., 2019) to validate SLFs with user interaction. Metamorphic testing approaches the test oracle problem from a perspective not typically employed by other testing strategies. Instead of focusing on individual test cases, metamorphic tests examine the outcomes of multiple test cases within a testing system and their relationships with each other. Any deviation from these relationships in the function's responses indicates a fault. The detection efficiency of this method depends on the formulation and understanding of these relationships, making it a creative and critical task for testers. The subsequent sections will define SLFs, elaborate on their characteristics, and demonstrate the application of metamorphic testing to ensure their reliability and adaptability. By adopting this innovative approach, we enhance testability to create more robust and precise SLFs, especially those involving user interactions.

SLFS WITH USER INTERACTIONS

This paper presents a method for validating SLFs that involve user interactions. Firstly, the term and the characteristic properties of SLFs are defined, followed by a validation of SLFs that adapt to user interactions.

Definition: SLFs (SLF): *A SLF is defined as the end product of a machine learning process (Carbonell et al., 1983), which can independently learn and adapt its behavior through the analysis of data or the observation of its performance.*

The properties of SLFs are characterized by (Russell and Norvig 2010)

1. **Data-Driven Learning:** SLFs rely on data-driven learning techniques such as supervised, unsupervised, and reinforcement learning. This method enables the system to learn and gain insights based on data sets.
2. **Adaptability:** The primary trait of a SLF is its ability to adapt and evolve based on new data or experiences. As more data is ingested, the function refines its internal parameters to improve performance.
3. **Optimization:** At their core, SLFs involve optimization processes, adapting parameters to find the best solution to a given problem based on the available data.
4. **Generalization:** SLFs are trained on specific data sets but aim to generalize their learning process to achieve results on unseen data.

   Generalization refers to the ability to make accurate predictions on new, previously unseen examples that were not part of the training or validation datasets.

A specialized subset of SLFs focuses on interaction with the user. These functions do not exclusively learn from data but instead adapt to the user's behavior and preferences and adapt accordingly.

Definition: SLFs with user interactions *are algorithms designed to customize operations by actively learning from user engagements. SLFs go beyond data processing and adapt to the individual preferences and behavior of the user.*

SLFs with use interactions have the following characteristics in addition to the mentioned properties of SLFs:

1. **User-Centric Adaptation:** These functions continually refine their output based on direct or indirect feedback from users. They adapt to individual user preferences and needs, making their responses more tailored over time.
2. **Online Learning:** These functions can continuously learn and update their model as new data becomes available rather than waiting for batch updates. This continuous adaptation allows the system to remain current with the most recent user interactions.
3. **Evolution Over Time:** As user behavior, preferences, and context change over time, these features adapt in the short term and evolve to ensure long-term adaptation to user needs and behavior.
4. **Comfort and Convenience:** SLFs with user interactions strive to improve user comfort and convenience by automating personalized actions and preferences for a seamless, user-centric experience.

In addition to the previously discussed features of SLFs with and without user interactions, it is important to consider differences in the training methods. These methods can be broadly divided into two categories, as outlined by (Ben-David et al., 1997): offline learning, which involves training SLFs without direct user interactions, and online learning, where SLFs continuously gather knowledge through ongoing user interactions (Figure 1).

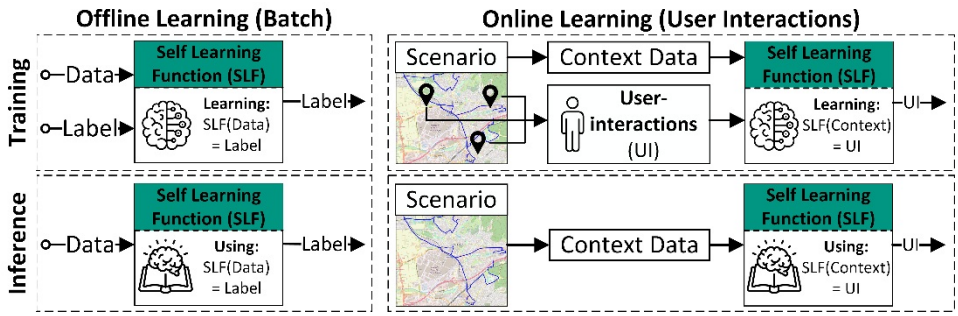

Figure 1: Comparison of offline learning with fixed labels and online learning with scenario data and user interactions.

In Offline Learning, SLFs are trained using a static dataset with specific labels. These labels instruct the SLF during its training phase. Once training is complete, the SLF can identify or categorize new, unseen data when it is

used for practical applications. The dataset used during training is crucial as it forms the base knowledge that the SLF uses to make predictions about new data.

In contrast, Online Learning is a more dynamic and continuously adaptive approach. Instead of relying on pre-assigned labels, it learns from ongoing user interactions, which provide feedback. This learning process is shaped by real-time scenarios, which are constantly changing. As a result, the SLF continually updates its understanding and predictions based on the latest user feedback and the interaction context. This method allows the SLF to constantly improve and adjust its responses to suit the current situation and user needs. Offline learning sets a solid foundation for SLFs to make future predictions, while Online Learning emphasizes adaptability, with the SLF constantly learning and adjusting to new information and user interactions as they happen.

TRADITIONAL TESTING METHODOLOGY

In software quality assurance, developing testing strategies is of central importance for managing the intricate challenges of software development. Within this spectrum, requirement-based testing (Castro et al., 2001) is recognized as a principal and historically entrenched methodology. This approach is predicated on the essential notion that software must conform to its predetermined specifications. It underscores a thorough validation directly anchored to the documented requirements. Among the various techniques employed within requirement-based testing is the V-Model (Bröhl, 1993), which serves as a structured and sequential development methodology. This model stands out for its distinctive feature of mirroring the development stages (requirements definition, system design, etc.) with corresponding testing phases (system testing, acceptance testing, etc.) on the opposite arm of the 'V', thereby integrating development and testing efforts in a cohesive and parallel manner. This methodological testing framework incorporates a suite of specific testing practices, such as unit testing, which focuses on the minor parts of the code; integration testing, which evaluates the interactions between different code segments; regression testing, which ensures that recent changes do not disrupt existing features; and scenario-based testing, which examines software performance under hypothetical use cases. At its core, requirement-based testing (Hetzel 1988), particularly when implementing the V-Model, revolves around the question, "Does the software perform its intended functions?" It involves crafting test cases that directly correspond with the software's documented requirements, thus guaranteeing that the application meets its delineated functional and performance criteria. The alignment of the test cases with the software requirements is crucial for checking whether the software fulfills the promised functions.

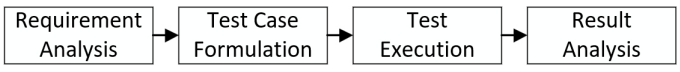

Figure 2: Traditional workflow of requirement-based testing.

In the V-model of software testing, a crucial part of the structured approach (Ammann and Offutt, 2016) shown in Figure 2, the process starts with test engineers conducting a detailed study of what the software is expected to do. They meticulously examine the list of software requirements to comprehend them fully. Following this, test engineers develop specific test cases tailored to these requirements. Each test case is constructed to verify whether the software functions in alignment with its expected behavior. Once the test cases are prepared, the actual testing phase begins. Test engineers run the software and closely observe whether it performs as intended. After the testing, test engineers assess the results. They scrutinize whether the software fulfills the initial requirements. These issues are recorded if there are any deviations or if the software fails to perform as required. This feedback is crucial as it enhances the software, ensuring it meets the established standards and functions correctly.

LIMITATIONS OF TRADITIONAL TESTING METHODS FOR SLF

SLFs continuously adapt and evolve by responding to incoming user interactions and data from their environment. Therefore, they do not exhibit static behavior that is identical for every user but is dynamically shaped, dependent on the individual user and scenario. The validation of SLFs through traditional software testing methods based on requirement-based approaches faces challenges when dealing with the unique characteristics of SLFs. These challenges include (Koopman and Wagner, 2016; Bach et al., 2017):

1. **Dynamic Behaviour versus Static Requirements:** Traditional requirements-based testing is based on the idea that software actions are consistent and determined by a fixed set of specifications. However, SLFs are inherently dynamic and designed to modify their behavior in response to new information. Therefore, a test scenario that passes today may not necessarily pass tomorrow, not due to a defect, but because the SLF has updated its behavior based on its learning process.
2. **Infinite Scenarios**: Requirement-based methods often define a limited set of test scenarios based on documented requirements. With SLFs, especially in real-time learning contexts, the potential range of inputs and scenarios is vast, if not infinite. Each user interaction introduces its unique behavior, context, and sequence of actions, making it nearly impossible to anticipate and document every potential scenario.
3. **Context Sensitivity:** User interactions with systems are not isolated events. They are set in a specific context that includes factors outside the immediate interaction, such as the user's mood, previous interactions, location, and time of day. These external factors can significantly influence the user's input and expectations of the system's results. Traditional testing methods that evaluate functionality in isolation overlook these contextual features, potentially missing key interaction patterns or user expectations.

STATE OF THE ART OF METAMORPHIC TESTING

With their dynamic and evolving nature, SLFs pose significant challenges to traditional requirement-based testing methods. These conventional methods

falter in the face of the inherent dynamics of systems that adapt and grow based on continuous learning from data. Metamorphic testing (MT), a property-based testing approach, offers a more robust solution to these challenges. Unlike traditional methods that focus on predefined expected outputs for specific inputs, MT is based on the inherent properties and invariants of the system, defined as metamorphic relations (MR). In contrast to contract-based testing (Guissouma et al.), where a system's specific expected output behavior under various conditions is defined, MRs determine how a set of outputs should change in response to modifications in the input. MRs offer an advantage over contract-based testing for test cases where a test oracle, a definitive source for assessing output correctness, is either unavailable or impractical. Initially, MRs were primarily perceived as simple transformation properties, like "the rotating of an image should not change the classification". Such relationships become the basis for MT. However, as the MT approach matured, the community started recognizing the versatility and depth of MRs. They are not limited to simple transformations but can encompass more complex relationships like those found in machine learning, where data perturbations should not radically change predictions. Over time, research (Chen et al., 2019) has emphasized the importance of identifying "useful" and "effective" MRs. An effective MR should have the following characteristics (Segura et al., 2016):

- **Relevance:** MRs should be directly associated with the software's functionality and the domain of its application.
- **Fault revelation:** MRs should be sensitive to faults, meaning that if there is a fault in the software, violating the MR should be probable.
- **Independence:** The MRs selected should be independent of each other to maximize fault detection. Overlapping or redundant MRs might lead to wasted testing effort without additional fault revelation.

IDENTIFICATION OF USER-SPECIFIC METAMORPHIC RELATIONS

This section examines the properties of SLFs concerning user interactions, leading to the formulation of relevant metamorphic relationships. The importance of determining these properties cannot be overestimated, as they serve as the basis for a differentiated analysis of the system under test (SuT). This analysis aims to evaluate the responsiveness and adaptability.

Ensuring Distinctiveness: The ability to differentiate is essential because it enables the SuT to offer personalized interactions and services. Each user of the SLF may have unique preferences, habits, and needs that the SuT should recognize and automatically consider. This testing property can verify whether the SLF can differentiate between users with different behaviors and extract individual routines for each user. The characteristic of "Ensuring Distinctiveness" can be divided into the following sub-aspects: On the one hand, it includes learning different user behaviors (MR 1.1), and on the other hand, it involves assigning the known patterns to a specific user (MR 1.2).

Recognizing the Transition of User Behavior: A dynamic user environment requires the SuT to recognize changes in user behavior. This property

encompasses two pivotal elements: assimilation of novel behavioral patterns and discontinuation of established ones. The former relates to the system's prowess in identifying and accommodating new user behaviors (MR 2.1), while the latter deals with its competency in phasing out behaviors that are no longer exhibited by the user (MR 2.2). These elements ensure that the system remains in sync with evolving user interactions.

Adapting to Environmental Shifts: While some behavioral transitions emanate directly from the user, others are precipitated by external environmental changes. Such external factors invariably influence user behavior, necessitating alterations in the system's routines. This property is segmented into three key facets: adjusting to time variations (MR 3.1), acknowledging seasonal transitions and adjusting system behavior accordingly, such as modulation of temperature controls in tandem with climatic changes (MR 3.2), and discerning the user's directional intent, leading to pertinent recommendations and adjustments during their journey (MR 3.3).

Upholding Robustness and Detecting Outliers: Central to the efficacy of these functions is their robustness - their ability to counteract inaccuracies and disruptions, particularly from sensor-induced perturbations. Since sensor data can be adulterated with noise, the algorithm's resilience in sifting through this noise to procure accurate data is paramount (MR 4.1). Furthermore, in the vast sea of data, outliers - data points that deviate markedly from the anticipated distribution - can emerge. A proficient SLF identifies these outliers and devises strategies to mitigate their potential distortions, ensuring the model's integrity and accuracy (MR 4.2).

Table 1. Summary of test properties and corresponding metamorphic relations.

Properties	Metamorphic Relation	ID
Ensuring Distinctiveness	Adapt to multiple user behaviors	MR 1.1
	Differentiate between users	MR 1.2
Recognizing the Transition of User Behavior	Adding new behavior user patterns	MR 2.1
	Removing known behavior user patterns	MR 2.2
Adapting to Environmental Shifts	Modification of the time characteristic	MR 3.1
	Recognition of seasonal effects	MR 3.2
	Recognition of the direction of travel	MR 3.3
Upholding Robustness and Detecting Outliers	Functionality with measurement noise	MR 4.1
	Ignore outliers	MR 4.2

IN-DEPTH DESCRIPTION OF TESTSETUP FOR MR 2.1

According to the MR definition, a suitable testing strategy for assessing compliance or non-compliance with metamorphic relationships is evaluated. Figure 3 illustrates the two testing stages: the source test case and the follow-up test case. The outputs of these test cases are employed to assess the differentiation capabilities of two distinct SLFs. The investigation focuses on two SLFs engineered to absorb and process information based on user interactions. These SLFs were developed to capture user behavior patterns

systematically but require an in-advance identification of the user as an input feature, as proposed by Vučinić et al. (Vučinić, 2023).

We establish a Source Test Case as the reference point for the foundational assessment. This scenario records the baseline behavior, represented by Person A navigating through a simulation involving interactions with two different user interactions (UIs), such as opening a window or adjusting the heating system.

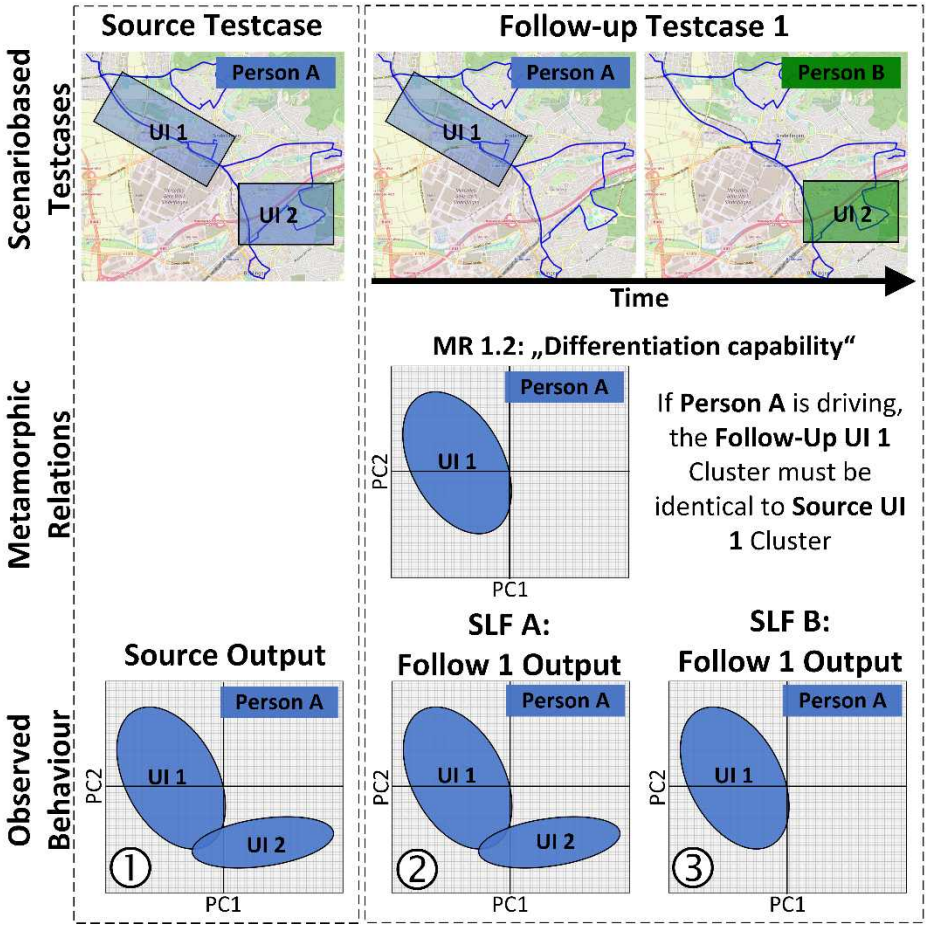

Figure 3: Illustration of the testsetup, consisting of source testcase and follow-up testcase 1.

The follow-up Test Case 1 is a variation of the Source Test Case: Contrary to it, the two UIs are executed by two individuals, not by one, as in the Source Test Case. The Metamorphic Relation MR 1.2 mandates that if Person A is driving, the follow-up UI 1 cluster must be identical to the Source UI 1 Cluster. The MR is validated by comparing the Source Output and the SLF Output of the Follow-Up data set.

In Figure 3, this comparison reveals that SLF Version A satisfies MR 1.2, as the output from SLF Version A is identical to that of the Source Test Case,

even though MR1.2 proclaims a difference. In contrast, SLF Version B shows the behavior described in MR1.2 and successfully satisfies the metamorphic test.

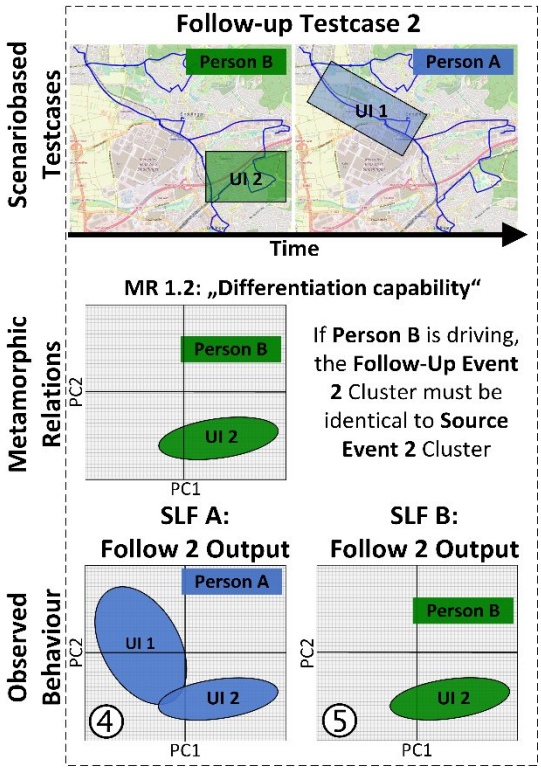

Figure 4: Follow-up Testcase 2 with interchanged user interactions.

Subsequently, Follow-up Test Case 2 (Figure 4) modifies Test Case 1 by swapping the UI responsibilities between the two individuals, assigning Person A to UI 1 and Person B to UI 2. The MR insists on consistency; if Person B is the driver, then the output of UI 2 in the follow-up should match the UI 2 output in the Source Test Case. Upon reviewing Figure 4, it is evident that SLF Version A's performance does not adhere to the prescribed MR, while SLF Version B exhibits shows compliance with MR 1.2

PROTOTYPICAL IMPLEMENTATION

The previously described test setup was evaluated using synthetically generated data with CAGEN (Stang et al., 2021). The metamorphic test case was devised for a comfort feature: a seat heating function with various intensity levels (Levels 1-2). In the specific case of the seat heating function, UI 1 and UI 2 correspond to the labels Seat Heating 1 and 2, respectively. The grey cluster aggregates the data points where the seat heating was not activated and is not shown in Figure 3 and Figure 4. To improve mapping, the numbers in Figure 5 correspond to those from Figure 3 and Figure 4.

The original model outputs for the source test case are depicted in a scatter plot matrix, illustrating the principal component analysis (PCA) projections, particularly the first two principal components. PC1 is the direction in a dataset that captures the maximum variance, representing the most significant pattern of data variation. PC2 is the second most significant direction, orthogonal to PC1, capturing the next highest level of variance in the data. The validation compares the PCA visualizations of the Source Test Case (1) against the Follow-Up Test Cases for SL Versions A and B (2,3,4,5). For the Validation of SLF Version A, comparing the Source Test Case (1) with the output of Test Case 1 (2) and Test Case 2 (4) reveals that SLF Version A does not adapt to different individuals according to MR1. 2, which proclaims, if only one person is driving there should be only one cluster. Instead, the SLF continues to associate both clusters and their behaviors to one single user.

In contrast, SLF Version B learned to recognize that the UI is operated by different individuals. This property can be observed by comparing the Source Test Case (1) with the output of Test Case 1 (3) and Test Case 2 (5). Only one cluster is discernible in (3) and (5), which also partly corresponds with the Source Test Case. However, it should be noted that the cluster from Seat Heating Lvl 1 in (3) does not entirely match the cluster from Seat Heating Lvl 1 in (1). The reason for this discrepancy warrants further investigation.

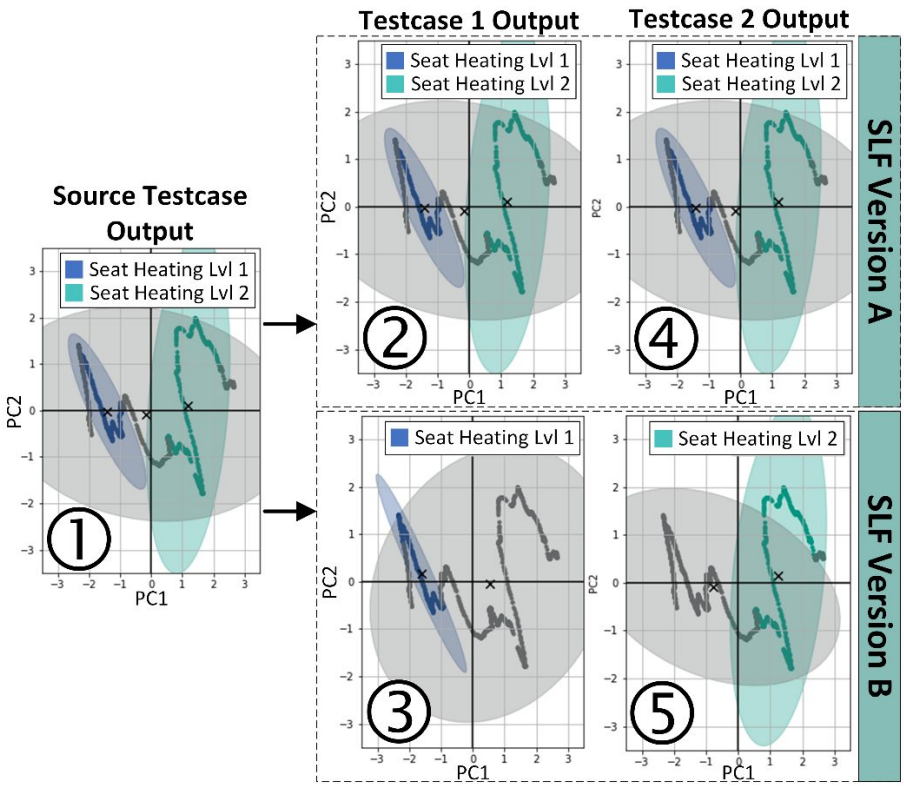

Figure 5: Output analysis for a self-learning seat heating function.

CONCLUSION

This paper presents an approach to evaluate SLFs with user interactions by utilizing metamorphic testing. Our methodology used the CAGEN tool to generate synthetic data, enabling the creation of precise and controlled test scenarios that mirror real-world scenarios. The results showcased the effectiveness of MRs in distinguishing the performance of different SLFs. We observed the SLFs' capabilities to adapt to user-specific interactions and discern and adjust to changing environmental factors. Notably, in individualization, SLF Version B surpassed SLF Version A, displaying enhanced personalization following the conditions of MR 1.2. The study underlines the promise of metamorphic testing as a critical resource for improving SLFs, confirming their effectiveness, and identifying further development opportunities.

In conclusion, metamorphic testing in the context of SLFs, with a specific emphasis on user interactions, emerges as a promising and efficient strategy. For the future, we are advocating extending this testing method to a broader range of scenarios and SLF models to strengthen the resilience and adaptability of these systems.

ACKNOWLEDGMENT

We thank the student assistant, Martin Duong, for his work and commitment to this project.

REFERENCES

Ammann, P. and Offutt, J. (2016) Introduction to Software Testing: Cambridge University Press.

Bach, J. et al. (2017) 'Data-driven development a complementing approach for automotive systems engineering', in 2017 IEEE International Systems Engineering, pp. 1–6. doi: 10.1109/SysEng.2017.8088295.

Ben-David, S., Kushilevitz, E. and Mansour, Y. (1997) 'Online Learning versus Offline Learning', Machine Learning, 29(1), pp. 45–63. doi: 10.1023/A:1007465907571.

Bertolini, M. et al. (2021) 'Machine Learning for industrial applications: A comprehensive literature review', Expert Systems with Applications, 175, p. 114820.

Bröhl, A.-P. (1993) Das V-Modell: Der Standard für die Softwareentwicklung mit Praxisleitfaden: Oldenbourg.

Carbonell, Jaime G. et al. (1983). An Overview on Machine Learning. In: Machine Learning. Elsevier, 3–23.

Castro, J., Kolp, M. and Mylopoulos, J. (2001) 'A Requirements-Driven Development Methodology', Advanced Information Systems Engineering. doi: 10.1007/3-540-45341-5_8.

Chen, T. Y. et al. (2019) Metamorphic Testing, ACM Computing Surveys, 51(1), pp. 1–27. doi: 10.1145/3143561.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning: MIT Press.

Guissouma, H. et al., ICARUS - Incremental Design and Verification of Software Updates in Safety-Critical Product Lines. In: 2021 47th Euromicro Conference, 371–378.

Hetzel, B. (1988) The complete guide to software testing: QED Information Sciences, Inc.

Koopman, P. and Wagner, M. (2016) 'Challenges in Autonomous Vehicle Testing and Validation', SAE International Journal of Transportation Safety, 4(1), pp. 15–24. doi: 10.4271/2016-01-0128.

Russell, S. J. and Norvig, P. (2010) Artificial intelligence a modern approach: London.

Segura, S. et al. (2016) 'A Survey on Metamorphic Testing', IEEE Transactions on Software Engineering, 42(9), pp. 805–824. doi: 10.1109/tse.2016.2532875.

Stang, M. et al. (2022) 'Development of a self-learning automotive comfort function: an adaptive gesture control with few-shot-learning', in 2022 International Conference on Connected, pp. 1–8. doi: 10.1109/ICCVE52871.2022.9742989.

Stang, M., Marquez, M. G. and Sax, E. (2021) 'CAGEN - Context-Action Generation for Testing SLFs', in Human Interaction 2021, pp. 12–19. doi: 10.1007/978-3-030-74009-2_2.

Vucinic, V., Seidel, L., Stang, M. and Sax, E. 2023. 'USID - Unsupervised Identification of the Driver for Vehicle Comfort Functions', IHIET-AI 2024. Manuscript submitted for publication.