ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20240126001293
Original scientific paper

Optimized Test Case Design for Automotive Safety Driving Virtual Training System Using Formal Activity Diagrams

Zhiyuan LI*, Mingju YAO

Abstract: In order to better test the virtual training system for automotive safety driving, ensuring that the system can efficiently and comprehensively cover all application scenarios, this paper defines the syntax and semantics of formal graphical symbols for testing. Based on this, a method for designing test cases for the virtual training system for automotive safety driving using formal activity diagrams is proposed. The example demonstrates that after using this method for test case design, the designed test cases can efficiently test all necessary application scenarios and eliminate invalid scenarios. This method effectively addresses the issues of strong randomness and low testing efficiency in system testing.

Keywords: acitvity diaram; scenario; test case; virtual training for automotive safety driving system

1 INTRODUCTION

The advancement and widespread adoption of computer technology, virtual reality, simulation, and network technologies have provided sophisticated technical underpinnings for the transformation and modernization of automobile safety driving training methods in the digital era [1, 2]. In this context, virtual training facilitated by computer-based automobile safety driving systems leveraging meta-universe technology has emerged as a novel paradigm in the realm of automobile safety driving training and education [3]. The intricate nature of automobile safety driving necessitates a highly elaborate training plot structure within the safety training systems. Given the diverse strategies and operations undertaken by different drivers during training activities within the automobile safety driving system, the training process evolves along distinct plotlines. The efficacy and cost-effectiveness of software testing for virtual training systems in automotive safety driving hinge significantly on the number and quality of test cases employed [4]. The inherent complexity of real-world scenarios often translates into substantial testing expenses during level testing. Inadequate design of software test cases for automobile safety virtual training systems not only squanders resources and time but also amplifies unnecessary workloads, potentially leading to oversight of latent software issues due to incomplete coverage. Hence, the crux of software testing for automobile driving virtual training systems lies in crafting comprehensive test cases that encompass diverse training scenarios within automobile driving virtual training levels, thereby facilitating a thorough examination of the automobile safety driving training system software in accordance with the test cases [5, 6]. The existing test data generation methods for VR driving simulations have certain limitations. They are limited in diversity, making it challenging to generate various driving scenarios to thoroughly test the capabilities of VR driving simulations. They also face scalability issues in handling the large volume of data required for comprehensive testing. Furthermore, they cannot guarantee realism and accuracy, as the generated data may not accurately reflect real-world driving scenarios, impacting the validity of simulation results. The efficiency of data generation is also lacking, leading to prolonged testing cycles and increased resource consumption. Additionally, they lack flexibility in adaptation to the evolving VR technologies and changing simulation requirements.

2 LITERATURE REVIEW

2.1 Training Techniques

Traditionally, training for safe driving of automobiles has been limited by factors such as time, location, safety, and cost, making it difficult to achieve the desired training outcomes. As a result, improving conventional driving safety education methods is essential for enhancing drivers' traffic safety awareness. Virtual reality (VR) technology is more effective than 2D technology in road hazard perception training and evaluation (Thomas 2021) [7]. VR-based training reduced people's risk-taking attitudes at intersections and speeding, demonstrating VR technology's potential in promoting traffic safety (Bari 2020) [8]. Based on VR technology, Liu (2022) constructed a dynamic education model driving safety education system targeting typical illegal driving behaviors [1]. Cutello (2020) had young drivers watch road safety films using VR technology to reduce their risky driving behaviors [11]. Cutello (2020) developed an Android mobile game using Unity3D to enhance the learning effect of driving safety through VR games [9]. Chraif (2013) developed a VR-based system that allows users to identify potential errors made by a virtual driver from a bystander's perspective and enables them to exercise conscious control, thereby improving the user's learning ability [10]. Ma (2020) applied VR technology to construct automobile accident cases in various disaster types to cultivate drivers' safety awareness in accidents [18]. However, the aforementioned studies primarily focus on game-like education for drivers, which increases the fun of learning but has limited application scenarios, insufficient system interactivity, and limited educational content [12]. The educational purpose does not analyze the driver's factors that cause traffic accidents and lacks in-depth education on drivers' safety awareness.

2.2 Virtual Simulation Technology

Virtual simulation technology leverages precise physical modeling, efficient numerical simulation, and high-fidelity image rendering to authentically replicate a human-vehicle environment model. This model encompasses vehicles, road infrastructure, weather conditions, lighting settings, traffic dynamics, and a diverse array of onboard sensor models. To address the multifaceted nature of the automotive driving environment and its intricate interactions with onboard environmental sensors, a comprehensive approach is adopted. This approach integrates geometric mapping, physical mapping, pixel mapping, and probability mapping techniques to generate virtual simulation test scenarios characterized by diverse attributes. These scenarios are tailored to meet varying application requirements, ensuring a nuanced and realistic simulation environment for testing and validation purposes. At present, many scholars have done a lot of in-depth research on the generation of digital virtual simulation scenarios. Hettab (2022) expounded on the current testing methods of autonomous driving systems and regarded full-coverage testing and accelerated testing as the focus of the next stage of research [13]. Anju (2019) sorted out and summarized the research progress on virtual testing technology for automotive safety driving based on scenarios, and outlined the principles of typical test scenario random generation and dangerous scenario reinforcement generation methods [14]. Jatana (2017) conducted a comprehensive analysis and summary of scenario-based testing methods, believing that scenario-based testing is a very promising method, and proposed a new scenario-based testing classification method [15]. Li (2013) proposed a simulation scenario construction method for intelligent driving test, and systematically expounded the research progress and status quo of scene construction and traffic modeling methods [17]. Based on the three abstraction levels of functional scene-logical scene-specific scene, proposed a specific scene automatic generation method for decision planning, which can directly generate Open SCENARIO-NARIO format scene files (Sun 2016) [20]. Although there are many comprehensive research reviews on scenario-based testing methods at present, scenario testing does not consider optimization problems. In terms of level design and test cases, there is a lack of systematic and comprehensive summary and analysis of level simulation scenario test case design and optimization.

3 RESEARCH METHODS

3.1 Level Ontology Construction

Ontology was introduced into the engineering domain during the 1990s with the purpose of delineating object types, concepts, attributes, and their interrelationships. Presently, it is defined as "a precise formal specification of a shared conceptual model." In accordance with this definition, ontology exhibits four fundamental characteristics. Firstly, "sharing" denotes that the concepts and knowledge encapsulated in ontology mirror the acknowledged understanding within the domain. Secondly, the term "conceptual model" refers to an abstract feature model derived by generalizing the principles of the real world, independent of specific contextual conditions. Thirdly, "clarity" signifies that each definition within the conceptual model is characterized by explicit constraints. Finally, "formalization" entails that ontology concepts are articulated using standardized language and can be processed automatically by computer systems. Leveraging these four key attributes of ontology, its integration into the scene extraction process ensures the method's generalization and standardization [19]. The scene ontology is articulated using the Unified Modeling Language (UML), comprising four essential components: concepts (C), attributes (A), operations (O), and relationships (R). Concepts represent various entities within the ontology, attributes denote distinct characteristics of these entities, operations signify executable actions associated with the concepts, and relationships delineate the connections between different entities. Attributes encompass both data type attributes and relationship type attributes. Data type attributes encompass enumeration types, numeric types, date types, and others, while relationship type attributes describe the associations between different concepts. It is important to differentiate relationship type attributes as a general attribute category, distinct from relationships that denote interactive links between diverse ontology concepts. Relationships encompass inheritance relationships and composition relationships. In inheritance relationships, the child concept inherits all attributes and operations from the parent concept, with the child concept being able to have multiple child sets but only one parent set. On the other hand, composition relationships represent the association between a whole entity and its parts, where the whole entity has a quantity of 1 and the parts have a lower limit. Notably, * signifies a lower limit of 0, while 1..* indicates a lower limit of 1. The foundational structure of the UML-based ontology is visually depicted in Fig. 1.

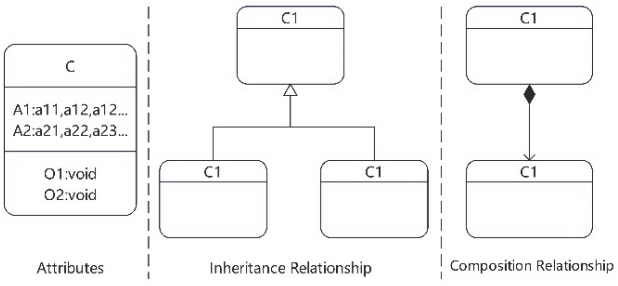

Figure 1 Structure of the ontology based on UML

Build a level ontology using the following six steps.

(1) Determine the domain and scope of the ontology: The ontology defined in this paper is oriented to typical levels commonly seen in automotive safety driving training. Since the currently commonly used natural driving level datasets are mostly safe scene data, this paper takes automotive safety data as an example to construct a level ontology.

(2) Synthesize existing ontology concepts: The ontology and its attributes and relationships constructed in this paper refer to existing definitions related to the safe driving test process based on scenarios, such as the 6 - layer scene model proposed by PEGASUS and the scene ontology related concepts proposed by Geyer1 et al.

(3) Define the concepts and attributes in the ontology: Considering the object-oriented and safe driving data constraints of the ontology, some of the ontology concepts, attributes, and operations defined in this paper are shown in Tab. 1.

Table 1 Partial concepts, attributes, and operations of the level ontology

Concept	Road	Vehicle	Weather
Attributes and operations	Road type, Shape Parameters, Road surface characteristics	Vehicle number, Vehicle type, Initial velocity, Distance relationship, Lateral position	Weather type

(4) Define concepts and hierarchical structures: In the process of ontology construction, considering that the ontology structure will be expanded as the types of available data increase in the future, this paper chooses to construct the level ontology structure by starting from the root concept and extending it down to the leaf concept layer by layer.

(5) Define concept properties: The relationship attributes in the level ontology mainly include the relative relationship between the target vehicle and the traffic vehicle, the relative relationship between the target vehicle and the road, the relative relationship between the traffic vehicle and the road, and the connectivity between lanes. The data type attributes mainly include the initial velocity of the vehicle, the lateral position of the vehicle, and the longitudinal position of the vehicle.

(6) Define property restrictions: Property restrictions are the key to logical scene construction. In logical scenes, they can be understood as the parameter space of different scene elements. Therefore, property restrictions are obtained through safe driving data analysis.

Through the above steps, the ontology structure of common test level scenes for automotive safety based on UML is finally established, as shown in Fig. 2.

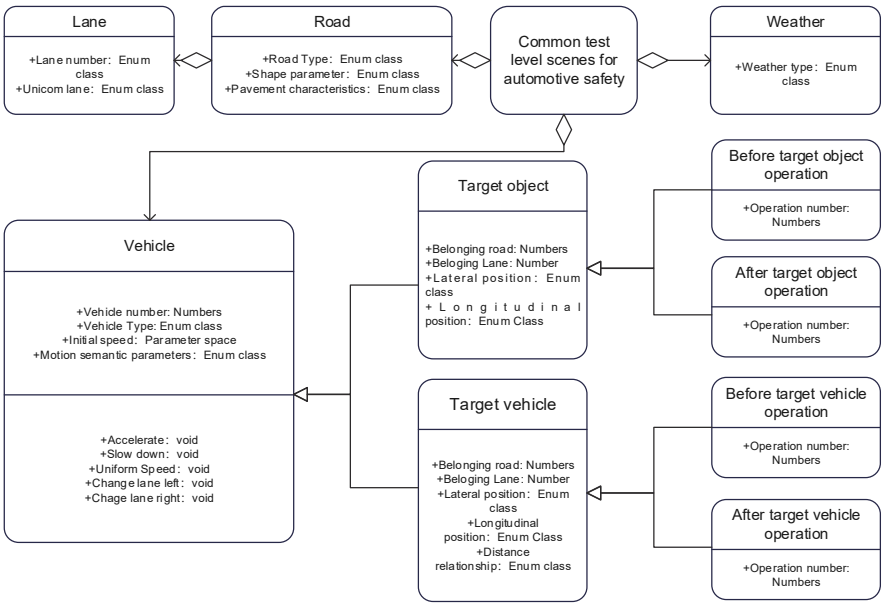

Figure 2 UML's ontology structure of common test level scenarios for automotive safety

3.2 Syntax and Definition of Car Safety Driving Virtual Training Activity Diagram

The formal activity diagram is a modeling tool commonly used in software engineering and system design to describe the flow of activities and interactions within a system. The concept of formal activity diagrams originated from the Unified Modeling Language (UML) and is a graphical representation method within UML. Formal activity diagrams provide an intuitive way to illustrate the relationships between activities and events in a system, aiding in the analysis and design of complex systems. Through formal activity diagrams, designers can gain a clearer understanding of the interactions and processes between different parts of the system, facilitating communication and collaboration during the system design and development process [16]. The formal activity diagram uses graphical symbols to describe the training activities in the automotive safety virtual training system and the relationship between activities. The formal graphical symbols of the automotive safety virtual training activity diagram are shown in Fig. 3.

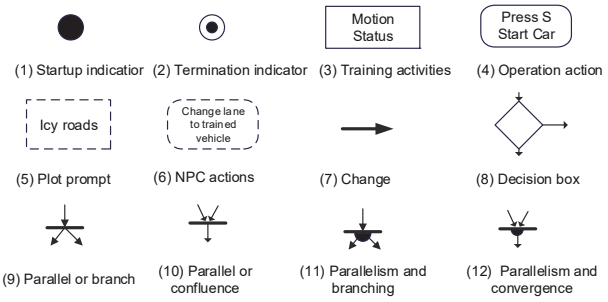

Figure 3 Formal graphic symbols for automotive safety virtual training activity diagrams

The semantics of each graphic symbol are as follows:
(1) Start indicator: Used to represent the starting state of an activity diagram or its subdiagrams, representing the first event in the activity sequence of the activity diagram, that is, the Start event.
(2) Termination indicator: It is used to represent the end state of the activity diagram. It is the last event that occurs

in the activity diagram, that is, the Finish event, indicating that the activity diagram has reached the end state.

(3) Training activities: Represents the training activities in the car safe driving virtual training system, which can be further decomposed into sub-activity diagrams composed of operation actions, decision boxes, parallel bifurcation points, parallel convergence points, etc. Training activity symbols hide unimportant details in the development process of the training plot, reducing the complexity of the virtual training system activity diagram.

(4) Operation actions: Operation actions are atomic activities and no longer contain sub-activities. They are used to represent relatively brief operations performed by the trained vehicle during the training process.

(5) Plot prompts: Used to prompt software testers about the development of training scenarios.

(6) NPC actions: NPC actions are also atomic activities and no longer contain sub-activities. They are used to represent the auxiliary, interactive or feedback actions made by NPC based on the development of the training plot and the operations of the trainees during the development of the training plot.

(7) Change: Represents transitions between training activities in an activity diagram.

(8) Judgment: There is a conditional expression in the determination box. The format of the conditional expression is:

$$f(x) < inequality > C \tag{1}$$

$$< inequality >= \left[> \| < \| = \| \; ! = \right] \tag{2}$$

The conditional expression is judged based on the data generated during the development of the training plot or the operations of the trainees. The result of the judgment determines the development of the next training plot.

(9) Paralleling or branching. It is used to decompose the virtual training plot into multiple conflicting plot branches, and the trainees can only choose one of the branch plots to execute.

(10) Parallel or confluence. It is used to merge multiple possible plot branches of training activities. As long as the plot of any branch reaches the convergence point, the training activities pointed to by the exit change can occur.

(11) Parallelism and branching. Used to decompose the virtual training plot into multiple parallel branch plots.

(12) Parallelism and convergence. It is used to merge multiple parallel branch plots. Only when all branch plots reach the converging point can the training activities directed by the exit change take place.

4 EXPERIMENT AND RESULTS

4.1 Car Safety Driving Training Level Test Case Design and Optimization Method Based on Formal Activity Diagram

The basic idea of the test case design and optimization method for the car safety driving virtual training system based on formal activity diagrams is: first, clarify the relationship between the training activities in the car driving safety training level, and generate a formal activity diagram of the car driving safety training level plot. Then decompose the formal activity diagram of the car driving safety training level plot into multiple fragments, analyze each fragment one by one, find out its possible corresponding event sequence, and design test cases for each fragment; on this basis, build the car. The fragment merging equation of the safe driving virtual training level test activity diagram is used to merge the test cases of each fragment according to the fragment merging equation; the merged test cases are optimized and formatted to generate the final car safe driving virtual training level test case set. The design and optimization process of test cases for car safety driving virtual training levels based on formal activity diagrams includes the following steps, as shown in Fig. 4.

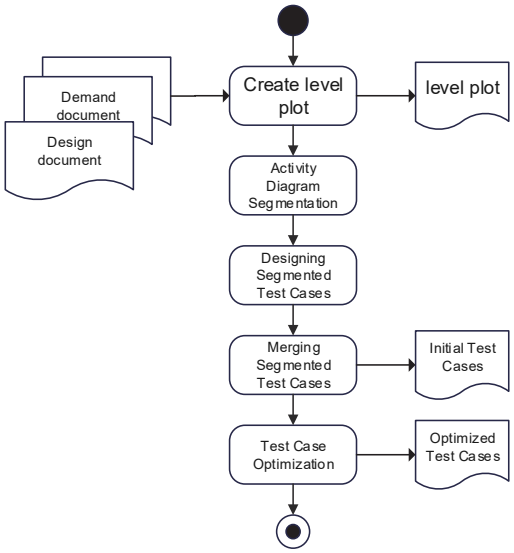

Figure 4 The design and optimization process of test cases for automobile safe driving virtual training levels based on formal activity diagrams

In its operational state, the transmitter generally emits a bell-shaped wave signal. The mathematical expression representing this weak pulse signal waveform is as follows:

Step 1: Create a plot activity diagram for the car safety driving virtual training level. Centering on the car safety driving training activities that need to be tested in the car safety driving training level, the activity sequence in the car safety driving level is modeled, and a formalized car safety driving virtual training level plot activity diagram is created.

Step 2: Activity diagram segmentation. Divide the created formal car safety driving virtual training level plot activity diagram into multiple smaller fragments. Each segment is an integral part of the plot activity diagram of the car safety driving virtual training level. Ideally, each segment should depict a plot feature in the activity diagram and include one or two operating actions.

Step 3: Design segmented test cases. Analyze each fragment of the activity diagram and design verification test cases for each fragment. A complete car safety driving virtual training level test case can be formally described by four-tuple.

$$TS = \left\langle D, S_{pre}, O, S_{feed} \right\rangle \tag{3}$$

D: Describes the test content of the test case. S_{pre}: Indicates the state of the tester at the car safety driving training level before performing the test operation. *O*: Indicates the operations that testers need to perform when testing. S_{feed}: Represents the expected output of the test, that is, the expected car safety virtual training level state after the tester performs the operation.

Step 4: Establish a fragment merging equation. The fragment merging equation of the car safety driving virtual training level plot activity diagram uses mathematical language to describe how to connect the verification test cases of each fragment to form a complete car safety driving virtual training level plot test case.

Step 5: Fragment verification test case merging. According to the segment merging equation, the test cases of each segment in the plot activity diagram of the car safety driving virtual training level are merged to generate a complete car safety driving virtual training level test case set.

1) Sequential relationship fragment test case merging. The set obtained after merging the test cases of activities *A* and *B* in the two sequence relationships is:

$$A \to B = \left\{ x \to y \middle| \left(x \in A \right)^{\wedge} \left(y \in B \right) \right\} \tag{4}$$

2) Or relationship fragment test case merging. The resulting set after merging the test cases of two OR-related activities *A* and *B* is:

$$A \oplus B = \left\{ x \in A \text{ or } x \in B \right\} \tag{5}$$

3) Parallel relationship fragment test case merging. The set obtained after merging the test cases of the two parallel relationship activities *A* and *B* is:

$$A \left\| B = \left\{ x \right\| y \middle| \left(x \in A \right) \wedge \left(y \in B \right) \right\} \tag{6}$$

Step 6: Test Case Optimization. Perform optimization processing on the merged set of test cases for automobile safe driving virtual training levels, eliminating those redundant or non-essential test cases for the software testing of automobile safe driving virtual training levels, and generate the final set of test cases for automobile safe driving virtual training levels. The optimization processing methods for test cases include deleting pseudo test cases, deleting inconsistent test cases, deleting test cases that do not contribute to the correctness verification of automobile safe driving virtual training levels, deleting non-operational test elements in test cases, merging similar test cases, etc. Additionally, some additional test cases can be constructed to check undefined automobile safe driving virtual training level behavior.

4.2 Car Safety Driving Virtual Training Level Test Case Design

This article takes the design process of a test case for testing the driving of a vehicle towards a destination on a virtual driving training platform for automobile safety driving as an example to illustrate the design method of test cases for automobile safe driving virtual training levels based on formal activity diagrams. The training scenario is briefly described as follows: After receiving the task of driving towards the destination, the trainee directs the vehicle to advance towards the destination. During the driving process, if the "weather is severe" training activity is triggered, the game NPC will inform the trainee of information such as vehicle skidding. After receiving warning information such as vehicle skidding, the trainee takes corresponding disposal operations to direct the vehicle to move forward correctly before it becomes dangerous. The correct approach is to direct the vehicle to advance towards the destination, and after reaching the warning location, perform operations such as reducing speed and changing to snow tires. Besides, the trainee may also take other operations, such as continuing to advance along the original route or reducing speed along the original route. If the trainee correctly implements operations after receiving meteorological warnings, after passing through the meteorological warning area, he/she directs the vehicle to continue to advance towards the destination along the preset route. After arriving at the destination on time, the training task is completed. If the trainee fails to correctly implement operations after receiving meteorological warnings, the vehicle will trigger damage information and the task will fail. The test case design is illustrated in Fig. 5.

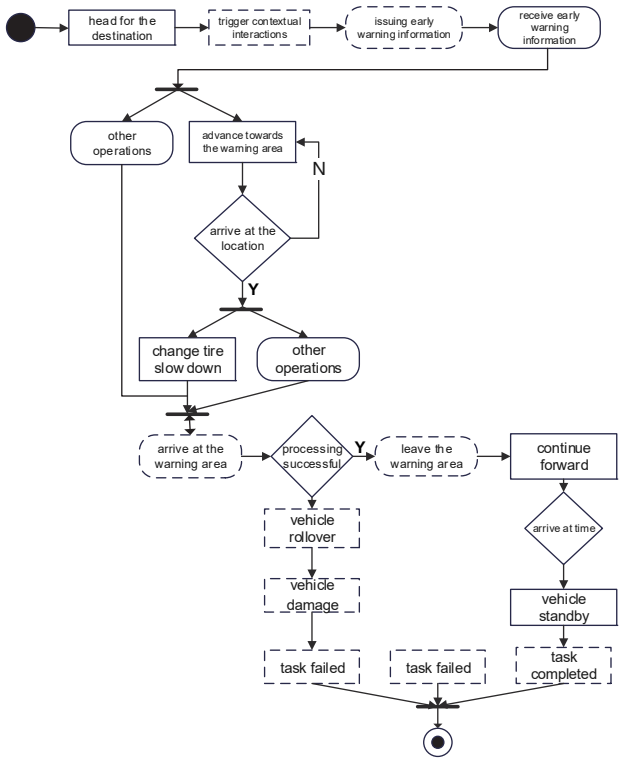

Figure 5 Drive towards the destination formalized plot activity diagram

5 DISCUSSION

Each operation training activity in the "Task Story Test Activity Diagram" can be seen as an atomic fragment of the "Drive to Destination" task story test activity diagram. The atomic fragments are combined to form fragments of the task story test activity diagram, and through continuous merging, a complete "Drive to Destination" task story

activity diagram is eventually generated. Taking the operation training activity "Accepting Weather Warning Information" as an example to illustrate the verification test case design process for operation training activity fragments. The "Accepting Weather Warning Information" operation training activity contains two test cases.

Test Case 1: Trainee correctly performs operation for receiving meteorological warning information.

Test Case 2: Trainee fails to correctly receive meteorological warning information.

The detailed design of the two test cases is as follows:

Test Case1:

{

Test content: The trainees are tested on their ability to correctly receive meteorological warning information;

The status before operation: NPC issues meteorological warning information;

Operation of trainees: correctly receiving meteorological warning information;

Expected output of the test: Display the content of the meteorological warning information;

}

Test Case 2:

{

Test content: The trainee did not correctly receive the meteorological warning information operation test;

The status before the operation: NPC issued a meteorological warning message;

Operation of trainees: did not correctly receive the meteorological warning information;

Expected output of the test: did not display the content of the meteorological warning message;

5.1 Generate Task Plot and Merge Equations of Plot Activities

After analyzing and designing test cases for each training activity, create task plot fragment merging equations to describe how to merge test case fragments of various training activities.

Start

→Command the vehicle to advance to the destination. Issue meteorological warning information.

→Issue meteorological warning information. destination. Issue meteorological warning information.

→(Receive meteorological warning information.

→(Command the vehicle to advance to the meteorological warning area.

→(Arrive at the warning area = True?}

→(Command the vehicle to change to snow tires andand slowly ⊕ other operations))) ⊕ other operations)

→(Meteorological warning area→(({Disposal correct = True?}

→Meteorological warning lifted→Command the vehicle to continue moving forward

→(({Arrive at the destination on time = True?}

→Command the vehicle to standby→Mission completed

⊕ ({Arrive at the destination on time = False?}→Mission fails))

⊕({Warning area processing successful = False?} → Danger occurs → Vehicle destroyed →Mission fails))

→Finish

5.2 Test Case Optimization

Optimize the generated task plot test case set of "advancing to the destination".

5.2.1 Delete Pseudo Test Cases

After generating the test case set, if there are any test cases with impossible conditions, they should be deleted. For example, in the test case set of the "advancing to the destination" task plot, the second test case states that the trainee fails to correctly command the vehicle to change to snow tires and drive slowly after receiving meteorological warning information. Therefore, when reaching the warning area, the judgment condition is impossible to be true, so the test case should be deleted from the test case set.

5.2.2 Delete Inconsistent Test Cases

If a test case has inconsistent plot, it should be deleted from the test case set. For example, the sequential relationship activities "A: publish meteorological warning information" and "B: receive meteorological warning information" have merged test cases. Among them, activity "A: publish meteorological warning information" contains two test cases "X: publish meteorological warning information" and "Y: do not publish meteorological warning information"; training activity "B: receive meteorological warning information" contains two test cases "U: correctly receive meteorological warning information" and "V: do not correctly receive meteorological warning information". After merging the sequential relationship fragments A and B, the resulting test case set is:

Test Cases = {
Start→X→U→Finish,
Start→X→V→Finish,
Start→Y→U→Finish,
Start→Y→V→Finish
}

The test case: Start→Y→U→Finish is unlikely to occur, so it should be deleted after the test case merge.

5.2.3 Delete Test Cases That Do Not Contribute to Task Plot Testing

For example, the test case Start→Y→V→Finish obtained after merging the sequential relationship fragments "A: publish meteorological warning information" and "B: receive meteorological warning information" is meaningless for training plot testing and does not contribute to system testing. Therefore, it should also be deleted. The test case set obtained after merging the sequential relationship fragments "A: publish meteorological warning information" and "B: receive meteorological warning information" will be:

Test Cases = {
Start→X→U→Finish,
Start→X→V→Finish

}

5.2.4 Delete Non-Operational Test Elements in Test Cases

Non-operational test elements are mainly used to assist in the design of the "advancing to the destination" task plot test cases. After the test cases are generated, the mission of the non-operational test elements has been completed. Therefore, it is possible to delete the non-operational test elements from the test cases to simplify the testing of the plot.

Test Cases: Start→A→B→C→…→Finish

A = {command the vehicle to advance to the destination};

B = {publish meteorological warning information};

C = {receive meteorological warning information}.

Among these elements, A and C are operational training activities that need to be performed by the trainees, and activity B is used to assist in designing the test cases. Therefore, after the test cases are generated, B can be deleted to obtain a test case composed of the operational sequences of the trainees: Start→A→C→…→Finish.

Through the analysis and comparison of the optimized final "advancing to the destination" training level test case set with the test case set before optimization, we found:

(1) The optimized test case set covers all possible operations that the trainees may perform in various atomic training activities.

(2) Each test case in the test case set represents a different level plot, and there are no more test cases that can be merged.

(3) Non-operational test elements that are not related to the trainees' operations have been removed from the test cases.

From the analysis of the optimization results, it can be seen that through the optimization of level test cases, while maintaining full coverage and verification testing of the training activities in the training level, the number of test cases in the test case set has been greatly reduced, and the efficiency of verification testing has been improved. After the test case design of the virtual training system for automobile safe driving uses the form activity diagram symbols and formulas proposed in this paper for test case design, different scenarios can be generated to test the ability of VR driving simulation, and at the same time, the expansion ability is effectively provided for the large amount of data required for comprehensive testing, and the generated test data can accurately reflect the real-world driving scene, and the validity of the simulation results is fully guaranteed. The efficiency of the operation of the test cases is greatly improved, resulting in a reduction in the test cycle.

6 CONCLUSION

Software testing is an important part of the development of virtual driving training systems for automobile safety. A large number of hidden problems and defects in the virtual driving training system for automobile safety need to be found and discovered through software testing. After the software developers make modifications, they need to conduct verification testing again. Test case design is a key factor that affects the efficiency and quality of software testing. This article proposes a method for designing and optimizing the test cases of the virtual training levels of automobiles based on formal activity diagrams. When designing the test cases of the virtual driving training levels for automobile safety, the combination explosion is avoided by optimizing the designed test cases. In the design of the syntax and semantics of formal activity diagrams for testing, the distribution of test cases in system distributed deployment was not extensively considered, which will be a focus of future research.

Acknowledgements

This work was supported in part by the Sichuan Education Information Technology Research, China (No.DSJ2022077).

7 REFERENCES

[1] Ma, F., Liu, Z. M., & Guo, F. (2019). Direct position determination for wideband sources using fast approximation. *IEEE Transactions on Vehicular Technology, 68*(8), 8216-8221. https://doi.org/10.1109/TVT.2019.2921981

[2] Kumar, G. & Bhatia, P. (2013). Software testing optimization through test suite reduction using fuzzy clustering. *CSJ Transactions on ICT, 1*(3), 253-260. https://doi.org/10.1007/s40012-013-0023-3

[3] Zhang, P. C., Li, F. G., & Wang, F. Y. (2023). Optimization and test of ginger-shaking and harvesting device based on EDED software. *Computers and Electronics in Agriculture, 213*(8), 108257. https://doi.org/10.1016/j.compag.2023.108257

[4] Vilela, R. & Neto, J. (2023). Bio-inspired optimization to support the test data generation of concurrent software. *Concurrency and Computation: Practice and Experience, 35*(2), e7489. https://doi.org/10.1002/cpe.7489

[5] Dijkstra, E. (1965). Solution of a problem in concurrent programming control. *Commun ACM, 8*(9), 569. https://doi.org/10.1145/365559.365617

[6] Yuan, X. & Yang, J. (2020). Effective Concurrency Testing for Distributed Systems. *Association for Computing Machinery*, 1141-1156. https://doi.org/10.1145/3373376.3378484

[7] Goodge, T., Kroll, V., Vernon, M., Ventsislavova, P., & Crundall, D. (2021). A comparison of cybersickness symptoms across 360-degree hazard perception and hazard prediction tests for drivers. *Applied Ergonomics, 97*, 103549. https://doi.org/10.1016/j.apergo.2021.103549

[8] Baric, D., Grigore, M., & Cornelia, M. (2020). Attitudes of learner drivers toward safety at level crossings: do they change after a 360° video-based educational intervention? *Transportation Research, 69*(0), 335-348. https://doi.org/10.1016/j.trf.2020.01.018

[9] Cutello, C., Gummerum, M., & Hanoch, Y. (2020). Evaluating an intervention to reduce risky driving behaviors:taking the fear out of virtual reality. *Risk Analysis, 41*(9), 1662-1673. https://doi.org/10.1111/risa.13643

[10] Chraif, M., Anitei, M., & Alex, S. (2013). The effects of exposure to the publicity campaign "Stop the Accidents" on the willingness to take risks in traffic situations. *Social & Behavioral Sciences, 78*(0), 562-566. https://doi.org/10.1016/j.sbspro.2013.04.351

[11] Cutello, C. A., Hellier, E., Stander J., & Hanoch, Y. (2020). Evaluating the effectiveness of a young driver-education intervention: Learn2Live. *Transportation Research Part F, Traffic Psychology and Behaviour, 69*(0), 375-384. https://doi.org/10.1016/j.trf.2020.02.009

[12] Tatale, S. & Prakash, V. (2022). Combinatorial test case generation from sequence diagram using optimization algorithms. *International Journal of System Assurance Engineering and Management, 13*(1), 642-657. https://doi.org/10.1007/s13198-021-01579-w

[13] Hettab, A., Chaoui, A., & Boubakir, M. (2022). Automatic scenario-oriented test case generation from UML activity diagrams: a graph transformation and simulation approach. *International Journal of Computer Aided Engineering and Technology, 16*(3), 379-415. https://doi.org/10.1504/IJCAET.2022.122153

[14] Bala, A. & Chhillar, R. S. (2019). Automatic Generation and optimization of test cases using genetic algorithm with UML diagram. *Journal of Engineering and Applied Sciences, 14*(5), 1590-1600. https://doi.org/10.36478/jeasci.2019.1590.1600

[15] Jatana, N., Suri, B., & Rani, S. (2017). Systematic literature review on search based mutation Testing. *E-Information Software Engineering Journal, 11*(1), 59-76.

[16] Kundu, D. & Samanta, D. (2009). A novel approach to generate test cases from UML activity. *Journal of Object Technology, 8*(3), 65-83. https://doi.org/10.5381/jot.2009.8.3.a1

[17] Li, L., Li, X., He, T., & Xiong, J. (2013). Extenics-based test case generation for UML activity diagram. *Procedia Computer Science, 17*(0), 1186-1193. https://do.org/ 10.1016/j.procs.2013.05.151

[18] Ma, Y. S., Offutt, J., & Kwon, Y. R. (2005). MuJava: an automated class mutation system. *Software Testing Verification and Reliablity, 15*(2), 97-133. https://doi.org/10.1002/stvr.308

[19] Ntafos, S. C. (2001). On comparisons of random, partition, and proportional partition testing. *IEEE Transactions on SoftwareEngineering, 27*(10), 949-960. https://doi.org/10.1109/32.962563

[20] Sun, C. A., Zhao, Y., Pan, L., He, X., & Towey, D. (2016). A transformation-based approach to testing concurrent programs using UML activity diagrams. *Software-Practice and Experience, 46*(4), 551-576. https://doi.org/10.1002/spe.2324

Contact information:

Zhiyuan LI, Associate Professor
(Corresponding author)
School of Intelligence Technology, Geely University of China, Chengdu, Sichuan,641423, P. R. China
No. 123, SEC.2, Chengjian Avenue, Eastern New District, Chengdu City, Sichuan Province
E-mail: lizhiyuan@guc.edu.cn

Mingju YAO, Master lecturer
School of Intelligence Technology, Geely University of China, Chengdu, Sichuan, 641423, P. R. China
No. 123, SEC.2, Chengjian Avenue, Eastern New District, Chengdu City, Sichuan Province
E-mail: yaomingju@guc.edu.cn