2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

A Model-Based Testing System for Safety of Railway Interlocking*

Haoxiang Su[1], Ming Chai[1], Hongjie Liu[1], Jinchuan Chai[2] and Chaopeng Yue[3]

Abstract— **Testing is an important safety assurance technique for railway interlocking systems. Model-based testing (MBT) allows for designing and maintaining tests with high-level models and generating test suites from these models automatically. Although MBT has the potential to improve testing efficiency and quality, it is not clear whether this technique is applicable for testing complex variant-rich interlocking software. In this paper, we report our experience in introducing MBT in interlocking testing. We develop feature models of the interlocking route control process to improve the reusability of the models. The experimental results show that our approach is able to improve the quality and efficiency of real-world interlocking acceptance testing.**

I. INTRODUCTION

In recent years, many new technologies have emerged in the field of railway transportation [1], [2], which also have higher requirements for system safety. People pursue more accurate and efficient control technology [3]–[8]. *Interlocking* is a critical system to ensure the safety of train operations at a station. An interlocking governs points, signals and routes to move in a proper and safe sequence, which is defined by interlocking principles. In most Chinese computer-based interlocking (CBI) software, these principles are instantiated on a station topology and are stored as an "*interlocking table*". The Software algorithm controls and monitors the dynamic status of points, signals and track segments, and decides whether a train is allowed to use a route with respect to the interlocking table.

Over the last decades, many efforts have been put on developing automatic interlocking testing techniques with script-based methods. However, difficulties of writing complete concrete test inputs and test oracles make script-based testing nontrivial. These problems become even worse when testing an industrial interlocking application. This is because interlocking has the feature of customization. That means every railway station deploys a specific interlocking system with respect to the station topology and special train operational requirements. Consequently, interlocking software is variant-rich and test scripts are not reusable for different industrial interlocking applications. Since developing test scripts is expensive, script-based methods are infeasible for testing industrial interlocking software.

Model-based testing (MBT) is a promising technique that automates the detailed development of tests. With this technique, a tester writes abstract models of the system under testing. The models specify behaviour and environment of the SUT, from which test inputs and oracles are generated automatically. Because tests are designed and maintained at the abstract model level, it is easy to produce as many test cases as possible and has potential to increase both the efficiency and effectiveness of testing. Unfortunately, interlocking models for generating tests are difficult to build. On the one hand, even for a medium-size station, the model involves a large number of variables, which would lead to the state explosion problem when generating test cases. On the other hand, an interlocking software is variant rich with different configurations of station topologies and station operation requirements. Developing individual models for each variant is time consuming, which neutralizes the benefits provided by MBT. Consequently, most railway companies still perform their interlocking testing in a conventional manual process, which is tedious and prone to errors.

The problem of safety assurance with railway interlocking systems with formal methods has a long tradition. Global model checking can provide complete correctness for interlocking, but it suffers from the state explosion problem for all but simple station topologies [9]. Bounded model checking (BMC) lacks completeness but is able to handle large stations [10]. Prover developed a commercial interlocking verification tool called iLock [11] with a domain-specific modelling language, which has been used by many Chinese interlocking suppliers. Although these methods are technically sound, the equivalence between models and the implementations is hard to prove. Consequently, deployed interlocking software still needs to be tested for safety guarantees.

Chai et al. proposed a formal method of runtime verification and used this method to monitor RBC system of high-speed railway [12]. Model-based testing uses system models to generate test cases automatically [13]. This technique plays an important role in model-driven interlocking validation [14]. MBT is feasible for testing industrial scale projects, whereas the major hurdle is in building suitable system models [15]. Many MBT tools have been developed with different modelling languages [16]–[18]. The capabilities of MBT tools using state-based models are reviewed in [19]. ParTeG [20] is an academic MBT tool that uses UML state machine diagrams. This modelling language is widely

*This research was partially supported by Fundamental Research Funds for the Central Universities(No.2022JBZY003), Beijing Natural Science Foundation (L201004), the subject of the science and technology research development plan of China Railway Corporation (L2021G009), Frontiers Science Center for Smart High-speed Railway System and National Railway Administration of the People's Republic of China funding program (KF2021-004).

[1]The authors are with National Engineering Research Center of Rail Transportation Operation and Control System, Beijing Jiaotong University, 100044 Beijing, China. Corresponding author: Ming Chai Email:chaiming@bjtu.edu.cn

[2]National Railway Track Test Center China Academy of Railway Sciences Corporation Limited Beijing, China

[3]Beijing National Railway Research & Design Institute of Signal & Communication Group Co, Ltd., Beijing

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9922557

used in the railway community. In railways, model-based equivalence class models written in SysML are used to testing ETCS Ceiling speed monitor functions [21]. To the best of our knowledge, no previous work has presented the feasibility of MBT in black-box testing for industrial scale variant-rich interlocking software.

In this paper, we report our experience of introducing MBT to perform testing on interlocking software. To solve the above problems, we propose a novel MBT approach for industrial railway interlocking software. Inspired by model-based product line (PL) testing, in our MBT approach the interlocking are formalized by feature models. To increase the reusablility of tests, we build feature models to specify basic functional units of interlocking. For test quality assessment, we utilize mutation testing to analyze fault detection capability of the MBT approach.

The rest of the paper is constructed as follows. Section 2 describes the interlocking characteristics. Section 3 gives a framework for applying MBT to interlocking acceptance testing. Section 4 illustrates the feature models of interlocking behaviours and partial functions. Section 5 evaluates MBT by measuring the mutation coverage and test efficiency. Section 6 discusses the results and presents the lessons learned. Section 7 contains the conclusion and planned future work.

II. Railway Interlocking

In railway transportation, a *route* is a continuous piece of railway on which a train can travel. A route is composed of trackside infrastructures of signals, points and track sections that is protected by a signal at the beginning of the route. An *interlocking* is respond to guarantee that routes can be used by a train with a safe manner. It is performed by controlling the trackside infrastructures in a way such that each movement satisfies a proper logic. A computer-based interlocking (CBI) system is responsible for guaranteeing the interlocking is achieved by controlling and monitoring the status of these infrastructures.

A typical architecture of Chinese CBI systems consists of the following three layers. *The human-machine interface (HMI)* layer provides operation interfaces for controlling routes. *The interlocking computing* layer computes the interlocking control logic with respect to the operation commands given by the HMI and status of trackside infrastructures. *The monitoring* layer provides I/O interfaces between the interlocking computing system and the trackside infrastructures.

A CBI controls routes with the following four functions processed in sequence.

- **Route selecting.** With the route selecting function, an intended route is called and the interlocking safety requirements are checked. To be specific, it requires that all relevant points are not locked at inappropriate positions, appropriate flank protections are available to points, no conflict routes are selected and all relevant track sections are clear. If any of these requirements are not satisfied, the route selecting function is aborted.
- **Route locking.** After the execution of route selecting function being completed the point control is triggered. With this function, all relevant points are transited and locked to the required positions, all signals protected the conflict routes are locked in stop positions, and the intended route is locked afterwards.
- **Route maintain.** With this function, all safety requirements are continuously checked until the train has entered the route. Before the train has passed the route, the CBI keeps the route in the lock state.
- **Route release.** After the train passing the route, the route locking is released.

In a CBI software, the interlocking principles are instantiated into the engineering data of the interlocking table with respect to the railway station layout. Figure 1 depicts a concrete example of a part of the a station layout. The whole station is composed of more than 100 routes, 140 points and 300 track segments.

III. An MBT Framework for Interlocking

To automatically generate test suits for interlocking, we use the academic MBT tool ParTeG, which is based on the Eclipse modelling framework (EMF). The modelling language is UML state machine diagrams with OCL expression. The generated test case formates are JUnit 4.

The MBT framework for interlocking testing is shown in Figure 2. In this framework, the tester builds system models and transforms them into UML models. With the chosen test quality criteria, ParTeG generates abstract test cases, which are further translated into concrete test scripts for automatic executing. A typical CBI system has four components: upper computer, interlocking machine, interface circuit and track-side equipment. The upper computer provides a man-machine interface (MMI) to operate the system and to display the interlocking execution results. The interlocking machine performs interlocking functions. It controls and monitors the track-side equipment via relay interface circuits, which are replaced by simulation systems for testing. The interlocking software is crucial and is not suitable for adapting tests. For automatic test executions, the test cases are translated into concrete interlocking conditions, which are fed into the MMI. Test scripts are the simulations of the manual operations of the interlocking system.

IV. Formalization of Interlocking Models

To perform MBT, most efforts have been spent on building interlocking behaviour models. Inspired by feature models of product line testing techniques, we build parametrized interlocking models to adapt the variant-rich character of the system. Instead of describing the data configuration of an entire station and interlocking control principles within one model, we build interlocking behaviour models based on route control procedures. This modelling approach has several advantages. Using routes as modelling units can limit the sizes of the model, which improves the test generation efficiency and avoids the state explosion problem. Formalizing route control procedures can disregard the different route topologies of different stations, which gives the potential to standardize the interlocking models.

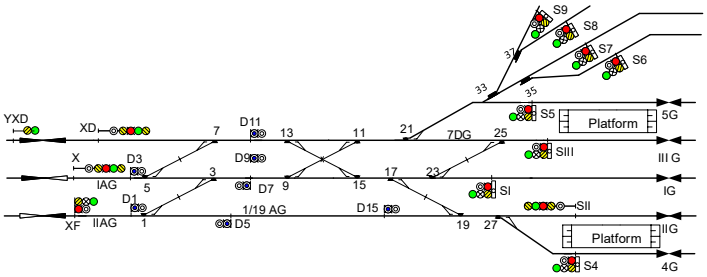

Fig. 1: Layout of a railway station

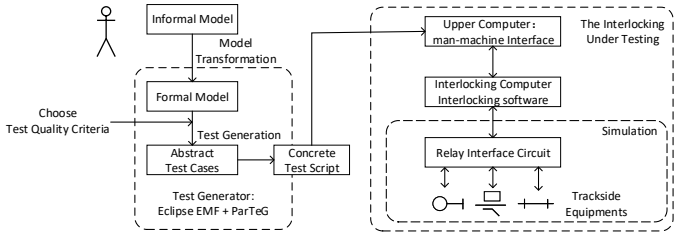

Fig. 2: The framework of the MBT system for railway interlocking

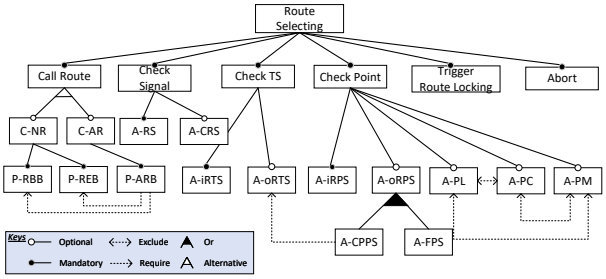

Fig. 3: Feature model for Route Selection

A. *Interlocking Behaviour Model for the Route Selection*

In this part, we present the models for testing the the route selection function of interlocking. The feature model of the function is as shown in Figure 3. The abbreviations used in the model are shown in Table I. The route selecting function has mandatory features of call a route, check status of signals, check status of track segments, check status of points, trigger the route locking function and abort the route selecting process. That is, for selecting any route, these features are required for performing the function. The Call Route feature has two sub-features of calling a normal route and calling an alternative route. To call a route, one needs to decide the scope of the route by sequentially pressing the route beginning button and the route ending route. When calling an alternative route, an additional feature of pressing the alternative route button needs to appear. Due to it is not allowed to just press the alternative route button, the feature P-ARB requires the feature P-RBB and P-REB.

The check signal feature has sub-features of checking route protecting signal and checking conflict route, if exits, protecting signals. To test this feature, the status of these signals are expected to be assigned by test cases. Therefore, these features are designed as assigning status of route signal and conflict route signal in the feature model. There are two kinds of track segments to be checked: The clearances of all track segments in the route are required to be checked. If the route has potential to be collided at flanking position, the clearances of relative track segments are to be checked. This collision condition is establishment if the clearance protection points being at certain positions.

TABLE I: Abbreviations used in the feature model of route selecting

Feature	Description
TS	Track segment
P-RBB	Press route calling button of the beginning-of-the-route
P-REB	Press route calling button of the ending-of-the-route
P-ARB	Press calling-alternative-route function button
A-RS	Assign route protecting signal a status
A-CRS	Assign conflict route protecting signal a status
A-iRTS	Assign occupancy status to track segment in the route
A-oRTS	Assign occupancy status to track segment outside the route
A-iRPS	Assign position to point in the route
A-oRPS	Assign position to point outside the route
A-PL	Assign status “lock” to point
A-PC	Assign status “closed-up” to point
A-PM	Assign status “malfunction” to point
A-CPPS	Assign a status to follow-up point
A-FPS	Assign a status to clearance protection point

A point to be checked can be either a component of the route or a relative point outside the route. To test check point feature, the points are supposed to be assigned some abnormal status by some test cases, including malfunctioned, closed-up and locked. Only one of these status can be appeared for a point, thus they apply the exclude constraints.

For the interlocking under testing as shown in Section 2, an iSMD as shown in Figure 4 can be obtained from the feature model. In the model, every self-transition assigns status of some track-side equipment. For the *Init* state, the self-transition assigns the route selected buttons, where Bun1st and Bun2nd represent the buttons of the beginning of a route and the end of a route, respectively. The self transition of state *Route Selected* assigns status of the protection signal of a route. The self transition of state *Signal NotLock* assigns

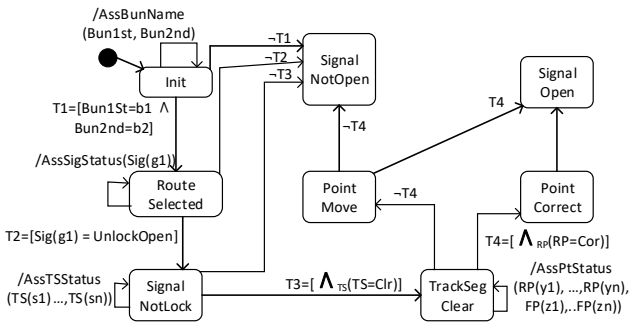

Fig. 4: Interlocking behaviour model for the route set-up

status of track segments of a route. The parameters $s_1, ..., s_n$ specify that a route has n segments. The self transition of state *TrackSeg Clear* assigns status of route relative points (RP) and follow-up points (FP).

The transitions between states specify the interlocking principles. The guards of transitions are boolean expressions obtained from the interlocking table. Guard $T1$ represents the *Route Button* column of the interlocking table, meaning that the route buttons are correctly pressed. By configuring an interlocking table, the parameters b_1 and b_2 are assigned the route buttons of a concrete route. Guard $T2$ represents that the protection signal of the route is not open nor locked. Guards $T3$ and $T4$ are represents the *Point* and *TrackSegment* columns of the interlocking table, meaning that all track segments of the route are not occupied by a train and all points are moved to and locked at correct positions. If these guards are satisfied, the system translate to the *Signal Open* state; otherwise, the system translate to the *Signal NotOpen* state. With those two states, test cases generated from the iSMD contain expected output of the interlocking.

In this model, the attributes are used to generate the test inputs, and the modes are used to generate the expected outputs of the interlocking.

B. Interlocking Partial Functional Models

According to our experience, with only normal interlocking behaviour models, the MBT tool cannot generate enough test cases to meet all the test requirements (TRs) as provided by our industry partner, the China Railway Corporation (CRC). The TRs have 33 abstract items for the interlocking safety and usability requirements that need to be verified. To adapt these TRs, we develop additional interlocking partial functional models, which are illustrated as follows.

TR1 After opening a route signal, the points (excluding the follow-up points) cannot be moved.

TR2 If a track segment of a route is occupied, the opened signal should be closed.

TR3 If a point on a route has no representation or is blocked, the opened signal should be closed.

TR4 After opening a signal, the conflict routes cannot be set up.

The iSMD for these requirements is as shown in Figure 5. The self transition of state *SignalOpen(x1)* assigns one of the

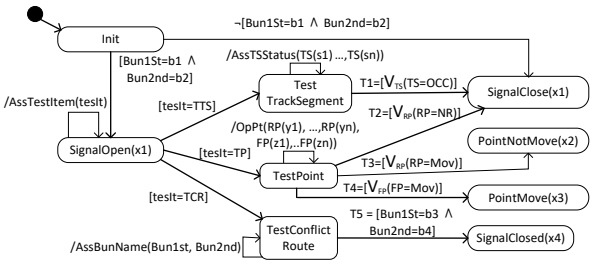

Fig. 5: A partial Functional Model for TR1 to TR4

TABLE II: Test generation results with different coverage criteria.

Model	Coverage Criteria	# of Test Cases	Gen. Time (h)
BM	AS	199	0.012
	AT	589	0.014
	DC	623	0.016
	MC/DC	4,982	3.47
	MCC	38,267	21.9
PFM	AS	32,193	0.4
	AT	39,493	0.5
	DC	61,831	0.51
	MC/DC	127,166	0.53
	MCC	203,899	1.06

above four test items. Parameter $x1$ of the state indicates the route to be tested. For testing track segment (TTS), if one of the track segment of the route is occupied, the signal $x1$ must be closed. That is the test requirement TR2 specified by the guard T1. For testing points (TP), the guard $T2$ specifies that if a point of the route has no representation, the signal must be closed (TR3); and the guards $T3$ and $T4$ specify that after opening a route signal, the route relevant points cannot moved, whereas the follow-up points can be moved. For testing conflict route (TCR), the guard $T5$ specifies an operation of fail to setup a conflict route.

V. Evaluations of Test Cases

To bridge the gap between the abstract model and the implementation, we develop a test adapter that maps abstract test cases to the interlocking. In the adapter, the test input is translated into operations of the interlocking upper computers, and outputs in test cases that are translated into a display of the execution results of the interlocking.

To meet all 33 test requirements as mentioned in section IV-B, we develop 8 behavioural models and 56 parametrized partial functional models. We perform the test generation for the whole station with coverage criteria of all states (AS), all transitions (AT), decision coverage (DC), modified condition decision coverage (MC/DC) and Multiple condition coverage (MCC). The test generation results are show in Table II, where BM and PFM represent the behaviour model and the partial functional model, respectively.

We apply mutation analysis techniques to evaluate the fault detection ability of the generated test cases. The mutation operators used in our experiments are designed with the following principles.

(1) Insert Element (IE): Inserting a new model element of a state or a transition.

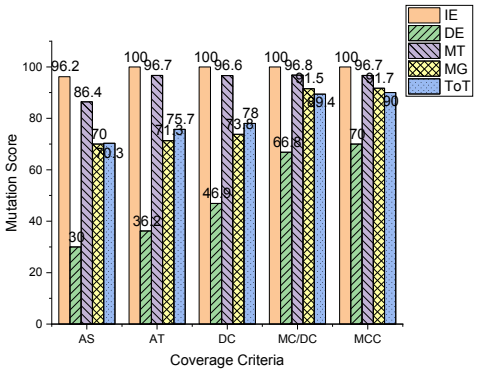

Fig. 6: Mutation analysis results of generated tests

(2) Delete Element (DE): Deleting a model element of a state or a transition.
(3) Modify Transition (MT): Changing the source or the target of a transition.
(4) Modify Guard (MG): The Boolean expression of the guard of a transition is modified, including replacing the logic operators with other logical operators, introducing a sub-formula, deleting a sub-formula and partially negating of the formula.

By combining these operators with the interlocking features, we generate various mutants and execute test cases on these mutants. For a mutant, if there exists a test case in the test suite that reports a failure, we say this mutant is killed. The number of killed mutants divided by the number of all mutants is the mutation score. The mutation scores of the generated test cases are shown in Figure 6, where ToT means the mutation score with all mutation operators. With manual testing, each interlocking engineer achieves a different score from 76.5 to 90.3.

We present the runtime for applying MBT for interlocking testing and compare it to that of manual tests. We measure the two main tasks for model-based interlocking testing: creating the models and implementing the test tools.

Building all 64 models take approximately 200 hours. Because the models are station-independent, the costs of developing these models are initial, one-time costs. They do not need to be rebuild in other projects. Applying specific interlocking data to the parametrized models (model configurations) and making small changes to the models (model maintenance) results in a few minutes of effort, which can be omitted by comparing this time to the overall testing time.

For manual interlocking acceptance testing, testers usually do not design test cases first but perform testing directly with respect to the test requirements. It takes an expert interlocking engineer approximately 2 hours to test one route. If an error is detected and the interlocking software is updated, all the routes should be tested again for regression testing. The second iteration of testing takes as much time as the first iteration.

To perform automatic test execution, a test tool is implemented. The tool includes functions of the test transformer and interlocking-testing interfaces. The effort to develop the test tool takes approximately 112 hours. The costs of

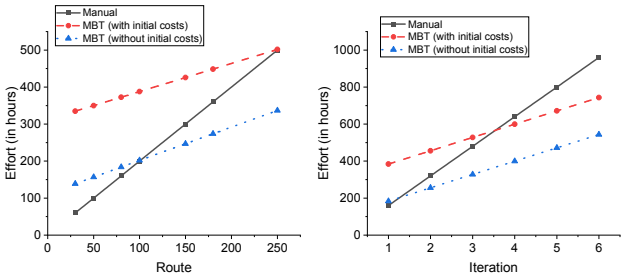

Fig. 7: Effort expended to introduce MBT into interlocking acceptance testing

introducing MBT (with the MCC criteria) in interlocking acceptance testing projects are as shown in Figure 7.

VI. Discussion and Lessons Learned

Model-based testing is a state-of-the-art technique for automatic testing. Because the technique is not applicable for "one-button" systems, building variant-rich interlocking system models is a challenge for adapting the MBT. We developed feature models of the system behaviour for controlling routes and partial functions to meet the interlocking acceptance test requirements of the CRC.

We generate different test suites with different coverage criteria. With the MCC criteria, the largest number of test cases are generated, and the highest mutation score is achieved. None of the existing test suites kill 100% of the injected faults. The main reason is that some of the mutant behaviours such as an introduced incorrect state or transition, are not described in the system models. Consequently, test cases cannot involve this information.

We show that without considering the initial costs, the application of MBT pays off for testing a medium-sized station with approximately 100 routes. During the course of interlocking acceptance testing, if there is no need for a second test interaction, then manual testing is more efficient than MBT for small stations. However, this assumption is unrealistic: according to the experience of the CRC, in all interlocking acceptance test projects, at least one mistake has been found during the first round of testing. In addition, changing the interlocking requirements or even changing the interlocking engineering data often occurs during acceptance testing. If the models need to be rebuilt for every project, MBT will pay off at the 4th iterations of testing. In interlocking acceptance testing, more than 80% projects takes two or three iterations. These results show that MBT is a feasible technique for railway interlocking acceptance testing. In the following, we present the lessons learned.

First, designing models for controlling one route instead of modelling the entire interlocking system provides various benefits. On the one hand, as a variant-rich system, each individual interlocking is closely related to the signal layout of a station. Formalizing the control process of a signal route with feature models can avoid describing the topology of a station, which improves the reusability of the models. On the other hand, the model sizes cannot be ignored when using MBT to generate tests for real-world implementations.

If the goal is to describe the whole picture of interlocking in one model, the model will be too large to collapse the ParTeG tool even for a very small station with only 10 routes. Describing the functions controlling one route is a way to keep the sizes of the models smaller.

Second, compared to manual testing, MBT generates a large number of test cases. For acceptance testing, test cases cannot be executed directly in the interlocking machines because the interlocking has to be modified to adopt test executions. Such software is not the same version as the one deployed in the field, which invalidates the test results. Therefore, the right way to perform acceptance tests is to execute them in upper computers. In our experiments, the interlocking supplier opens the upper computer interfaces and helps accelerate the development. In most projects, suppliers refuse to open these interfaces to protect their technical secrets. As an upper computer contains the signal layout of the station, these interfaces cannot be reused in different projects. Methods for creating universal test execution agents are desired.

Third, during the experiments, we obtain feedback that although most CRC engineers considered MBT to be a powerful tool, two issues were raised by the engineers. On the one hand, people from railway communities are not familiar with formal modelling languages and need extensive training to understand models and use MBT efficiently. And tool integration is a problem for adapting MBT into the existing interlocking acceptance procedure. On the other hand, railway engineers consider the safety of using MBT tools. As already mentioned, none of the test suites reach mutation scores of 100. Although many mutations have no physical meaning in the context of interlocking, domain-related test-stopping strategies should be developed to enhance the confidences of users.

VII. Conclusion

In this paper, we present our experiences of introducing model-based testing in railway interlocking acceptance testing. We describe the Chinese interlocking implementations and the introductions of MBT for interlocking testing. For this purpose, we concentrated on presenting the MBT organization framework, the designed interlocking models and the lessons learned. Furthermore, we show our experimental results on the effectiveness and efficiency of MBT: We measure the fault detection abilities of MBT and compare the effort required for MBT and manual testing. The results show that MBT is a promising technique to improve the quality and efficiency of interlocking acceptance testing.

There are several interesting directions for future work. First, to improve MBT efficiency, we will develop a modelling method that generates models directly from the test requirements written in natural languages. Second, domain-related test strategies are required to generate better test suites. Last but not least, it remains open how to create a universal test execution agent that is able to perform tests on upper computers without additional interface knowledge.

References

[1] M. Chai, H. Su, and H. Liu, "Long short-term memory-based model predictive control for virtual coupling in railways," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[2] Q. Wang, M. Chai, H. Liu, and T. Tang, "Optimized control of virtual coupling at junctions: a cooperative game-based approach," in *Actuators*, vol. 10, no. 9. MDPI, 2021, p. 207.

[3] Y. Liu, Y. Zhou, S. Su, J. Xun, and T. Tang, "An analytical optimal control approach for virtually coupled high-speed trains with local and string stability," *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 102886, 2021.

[4] X. Wang, T. Tang, S. Su, J. Yin, Z. Gao, and N. Lv, "An integrated energy-efficient train operation approach based on the space-time-speed network methodology," *Transportation Research Part E: Logistics and Transportation Review*, vol. 150, p. 102323, 2021.

[5] W. Liu, S. Su, T. Tang, and X. Wang, "A dqn-based intelligent control method for heavy haul trains on long steep downhill section," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103249, 2021.

[6] S. Su, X. Wang, T. Tang, G. Wang, and Y. Cao, "Energy-efficient operation by cooperative control among trains: A multi-agent reinforcement learning approach," *Control Engineering Practice*, vol. 116, p. 104901, 2021.

[7] W. Liu, S. Su, T. Tang, and Y. Cao, "Study on longitudinal dynamics of heavy haul trains running on long and steep downhills," *Vehicle System Dynamics*, pp. 1–19, 2021.

[8] S. Su, J. She, K. Li, X. Wang, and Y. Zhou, "A nonlinear safety equilibrium spacing-based model predictive control for virtually coupled train set over gradient terrains," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 2, pp. 2810–2824, 2021.

[9] K. Winter, W. Johnston, P. Robinson, P. Strooper, and L. Van Den Berg, "Tool support for checking railway interlocking designs," in *Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55*. Citeseer, 2006, pp. 101–107.

[10] L. H. Vu, A. E. Haxthausen, and J. Peleska, "Formal modelling and verification of interlocking systems featuring sequential release," *Science of Computer Programming*, vol. 133, pp. 91–115, 2017.

[11] A. Borälv, "Interlocking design automation using prover trident," in *International Symposium on Formal Methods*. Springer, 2018, pp. 653–656.

[12] M. Chai, H. Wang, T. Tang, and H. Liu, "Runtime verification of train control systems with parameterized modal live sequence charts," *Journal of Systems and Software*, vol. 177, p. 110962, 2021.

[13] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software testing, verification and reliability*, vol. 22, no. 5, pp. 297–312, 2012.

[14] A. Bonacchi, A. Fantechi, S. Bacherini, and M. Tempestini, "Validation process for railway interlocking systems," *Science of Computer Programming*, vol. 128, pp. 2–21, 2016.

[15] J. Peleska, "Industrial-strength model-based testing-state of the art and current challenges," *arXiv preprint arXiv:1303.1006*, 2013.

[16] C. Artho, M. Seidl, Q. Gros, E.-H. Choi, T. Kitamura, A. Mori, R. Ramler, and Y. Yamagata, "Model-based testing of stateful apis with modbat," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 858–863.

[17] F. Basanieri, A. Bertolino, and E. Marchetti, "The cow_suite approach to planning and deriving test suites in uml projects," in *International Conference on the Unified Modeling Language*. Springer, 2002, pp. 383–397.

[18] W. Krenn, R. Schlick, S. Tiran, B. Aichernig, E. Jobstl, and H. Brandl, "Momut:: Uml model-based mutation testing for uml," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–8.

[19] M. Shafique and Y. Labiche, "A systematic review of state-based test tools," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 1, pp. 59–76, 2015.

[20] S. Weißleder and B.-H. Schlingloff, "Automatic test generation from coupled uml models using input partitions," in *4 th MoDeVVa workshop Model-Driven Engineering, Verification and Validation*, 2007, p. 23.

[21] C. Braunstein, A. E. Haxthausen, W.-l. Huang, F. Hübner, J. Peleska, U. Schulze, and L. Vu Hong, "Complete model-based equivalence class testing for the etcs ceiling speed monitor," in *International Conference on Formal Engineering Methods*. Springer, 2014, pp. 380–395.