Safety-oriented Testing for High-speed Rail Onboard Equipment Using Petri Nets

Yike Li[1], Yin Tong[2,*], Marco Demuro[3] and Alessandro Giua[1]

[1]*Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy*
[2]*School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China*
[3]*ARST S.p.A., Trasporti Regionali della Sardegna, Cagliari, Italy*

Abstract
With its ability to operate at high speeds and capacity, high-speed rail offers a fast, dependable, and eco-friendly urban transportation option. Safety-critical systems such as high-speed rail signaling systems must be tested regularly to assess compliance with specifications and ensure reliable performance. Given that the onboard equipment is the core component of the signaling system, conducting safety testing on this equipment is of utmost importance. Current methods of analyzing test requirements mainly rely on human interpretation of specifications. However, the official technical specifications usually only outline standard operational scenarios, which could result in an inefficient and unclear safety analysis. This paper focuses on safety-oriented testing for onboard equipment. In particular, we propose a Petri net based approach to generate test cases for diverse operational scenarios. This approach improves both the efficiency and reliability of the testing process while ensuring compliance with safety requirements.

Keywords
Petri nets, High-speed railway, Onboard equipment, System testing

1. Introduction

In high-speed rail lines where the maximum train speed can reach over 350 kilometers per hour, even a small error or malfunction could have catastrophic consequences. Onboard equipment is a crucial component of high-speed rail responsible for ensuring safe and efficient train operations. Therefore, it is vitally important to test onboard equipment thoroughly. In particular, anomalies and exceptions that could lead to hazards must be identified and tested before putting into service. The primary objective of safety-oriented testing is to design test cases that assess the system's performance under anomalies.

Various factors which may affect the function execution of onboard equipment [1], including the internal states of the device itself and the external inputs from the track and balise (an electronic beacon placed between rails). Normally, the test engineer designs test cases based on

Algorithms & Theories for the Analysis of Event Data and Petri Nets for Twin Transition 2023, June 26–27, 2023, Lisbon, Portugal
*Corresponding author.
y.li@phd.poliba.it (Y. Li); yintong@swjtu.edu.cn (Y. Tong); m.demuro@arst.sardegna.it (M. Demuro); giua@unica.it (A. Giua)
https://faculty.swjtu.edu.cn/tongyin/ (Y. Tong); https://www.alessandro-giua.it/ (A. Giua)
0009-0008-9744-3946 (Y. Li); 0000-0003-1850-0188 (Y. Tong); 0000-0002-3859-9472 (A. Giua)

the specifications and then executes them in a virtual test environment while observing the system's behavior. Current studies about testing of rail signaling systems mainly focus on three aspects: a) developing a virtual simulation environment for testing [2, 3]; b) enhancing test automation and efficiency [4]; c) increasing the comprehensiveness of test scenarios [5, 6]. The third type of aforementioned studies usually needs to consider various faults or malfunctions that may lead to unexpected outcomes during the system's operation in order to meet the safety requirements. This involves identifying fault scenarios, selecting appropriate external events, and determining their timing and location. However, relying solely on expert experience and human judgment can result in inefficiencies and unreliable outcomes [7, 8].

On the other hand, formal methods rely on mathematical tools and employ various symbolic languages or representations to describe and analyze systems. These methods effectively mitigate ambiguity and reduce redundancy inherent in the use of natural languages [9, 10]. In safety analysis and testing, formal models mainly include the Unified Modelling Language (UML) [11, 12], timed automata [13, 14] and Petri nets [15, 16]. UML is a semi-formal approach and lacks the capability to examine the model, making it difficult to ensure its correctness. The main focus of timed automata lies in modeling time-dependent systems. As the number of clocks and the complexity of timing constraints in the model increase, the complexity of analysis and verification grows. Moreover, it is challenging to describe mechanisms such as concurrency and resource allocation using timed automata since the state space has to be enumerated first. Petri nets offer distinctive advantages compared to other methods with a series of well-developed mathematical tools and the ability to describe multiple structures within complex systems. For rail signaling systems, Petri nets are widely used for different safety-related issues, such as safety and reliability analysis [17], control policy design [18, 19], and fault diagnosis [20, 21]. To the best of our knowledge, there is currently no work available regarding the generation of test cases for rail signaling systems using Petri nets.

In this paper, we apply Petri nets to model the requirement specifications of onboard equipment, based on which test cases covering different scenarios through the injection of faults can be obtained. Finally, the proposed approach is illustrated by modeling the safe-critical scenario of level-conversion.

2. Preliminaries

2.1. Petri Net Model

A *Petri net* is a structure $N = (P, T, Pre, Post)$, where P is a set of m *places* graphically represented by circles; T is a set of n *transitions* graphically represented by bars; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post-incidence functions* that specify the arcs directed from places to transitions, and vice versa, where $\mathbb{N}$ is the set of non-negative integers. We also denote by $C = Post - Pre$ the incidence matrix of a net. The set of *input places* of t is the set of places $p \in P$ that have an arc going from p to t, denoted by $\cdot t$ and the set of *output places* of t is the set of places $p \in P$ that have an arc going from t to p, denoted by $t\cdot$. Analogously, $\cdot p$ and $p\cdot$ represent the set of *input transitions* and *output transitions*, namely the set of transitions $t \in T$ that have an arc going from t to p, and from p to t, respectively. Given a set A, its cardinality is denoted by $|A|$.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place a non-negative integer number of tokens, represented by black dots. We denote by $M(p)$ the marking of place p. A *Petri net system* $G = \langle N, M_0 \rangle$ is a net N with an initial marking M_0.

A transition t is *enabled* at marking M if $M \geq Pre(\cdot, t)$ and may fire yielding a new marking $M' = M + C(\cdot, t)$. We use $M[\sigma\rangle$ to denote that the sequence of transitions $\sigma = t_{j1} \cdots t_{jk}$ is enabled at M, and $M[\sigma\rangle M'$ to denote that the firing of σ yields M'. The set of all transition sequences that can fire in a net system G is denoted by $L(G) = \{\sigma \in T^* | M_0[\sigma\rangle\}$, where T^* denotes the Kleene closure of T, i.e., the set of all sequences over T including the empty sequence ε.

A marking M is *reachable* in G if there exists a firable sequence $\sigma \in L(G)$ such that $M_0[\sigma\rangle M$. The set of all markings reachable from M_0 defines the *reachability set* of G and is denoted by $R(G)$. A Petri net system is *bounded* if there exists a non-negative integer $K \in \mathbb{N}$ such that for any place $p \in P$ and for any reachable marking $M \in R(G)$, $M(p) \leq K$ holds.

For bounded Petri net systems, it is possible to enumerate in a systemic way the reachability set by means of the *reachability graph*. In the reachability graph, each node corresponds to a reachable marking, and each arc corresponds to a transition.

2.2. CTCS-2 Onboard Equipment

The Chinese Train Control System (CTCS) consists of five application levels, progressing from Level 0 to Level 4, each designed to meet different requirements and conditions [22]. C0 and C1 are designated for the current normal-speed railway lines, operating at approximately 120 kilometers per hour. C2 and C3, on the other hand, are well developed and widely employed for high-speed lines, while C4 is currently not in use. C2 has two features: a) continuous supervision of train movement by onboard equipment, and b) non-continuous communication between the train and the trackside through balise. The structure of C2 is shown in Fig. 1. Consisting of the onboard subsystem and trackside subsystem, C2 relies on the track circuit to detect track occupancy and train integrity and relies on balises to receive ground information encoded by the train control center. When the train passes over a balise, the Balise Transmission Module (BTM) on the train will send a signal to activate the balise to start the telegrams transmission process. The telegrams include messages such as the train location and route information (gradient, speed limit, etc.).

The onboard subsystem, also referred to as onboard equipment, is the safety platform installed on the train, composed of vital processing unit, driver-machine interface, speed sensor, balise transmission module, track circuit antennas, etc. The onboard equipment calculates the continuous speed control curve to guarantee the train's safe operation in real-time, combining the train's parameters with the received data from the track circuit and balise.

In order to support different operating conditions, the onboard equipment has a variety of operating modes. Also, the different levels of onboard equipment in C0-C4 have backward compatibility so that the C2 onboard equipment can also work in C1 or in C0. The conversion between two working levels, known as *level-conversion*, is executed by the onboard equipment by altering a series of current control policies.

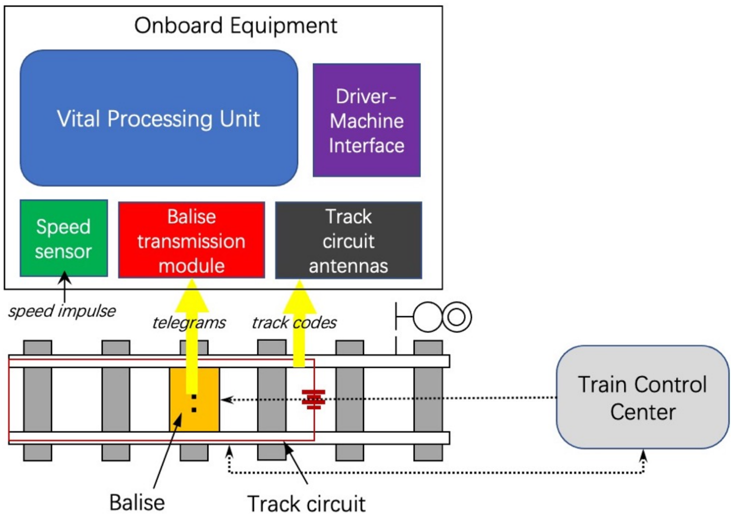


Figure 1: Structure of the CTCS-2 onboard equipment.

3. Design Safety-Oriented Test Cases Using Petri Nets

Safety-oriented testing for onboard equipment aims to evaluate the system's performance during failures or errors. The test cases should cover not only typical scenarios but also abnormal ones, in which one or more subsystems fail or are limited [1]. This section illustrates the application of Petri nets in designing safety-related test cases, covering diverse occasions that contain failures. Specifically, the level-conversion scenario has been chosen as a representative case to demonstrate the feasibility and efficiency of this approach.

3.1. Level-conversion Scenario from C2 to C0

According to the system requirements of C2 Onboard equipment [23], the level-conversion scenario from C2 to C0 consists of the following two cases.

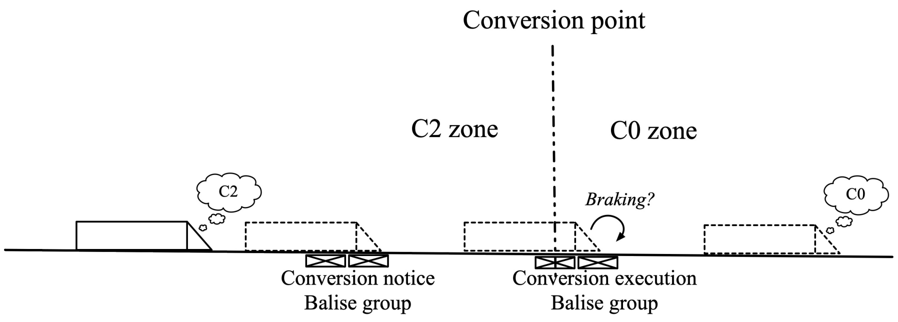
Case 1. The train runs to the boundary between the C2 zone and the C0 zone, and receives the message from the balises indicating the train should execute conversion from C2 to C0. If the train is braking, it will remain at C2 until the brake is released.

Case 2. In the C2 zone, the driver manually selects C0 through the DMI (Driver-Machine Interface).

For the first case, automatic level-conversion, the onboard equipment needs to perform an automatic degradation from C2 to C0, by changing a range of train control strategies such as operating mode, maximum speed limitation, to suit the line condition of C0.

Fig. 2 shows the layout of the two level-conversion balise groups on the track and the movement of the train crossing them. The first group is positioned in the C2 zone, usually 240m or 260m before the border, while the second one is positioned at the border.

[1]Note that in this paper, the words *safe/safety* do not refer to 1-bounded nets in Petri net theory, but refer to the general term in system engineering.



Conversion point
C2 zone
C0 zone
C2
C0
Braking?
Conversion notice Balise group
Conversion execution Balise group

Figure 2: Automatic level-conversion from C2 zone to C0 zone.

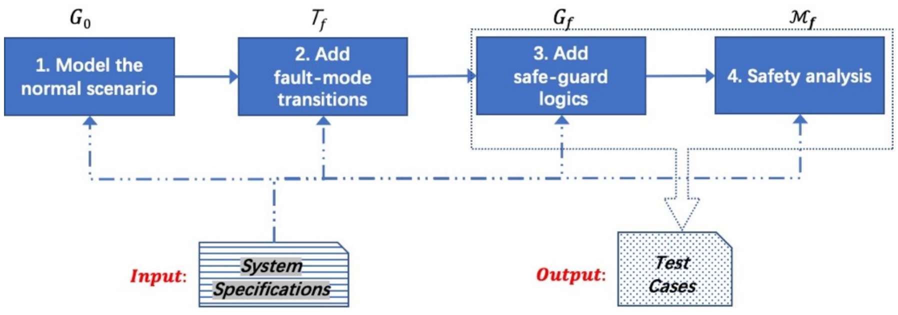


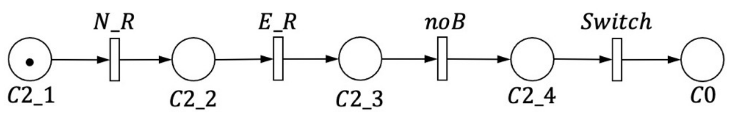
Figure 3: Overview of the proposed approach.

3.2. Scenario Modeling

As mentioned earlier, safety-oriented testing aims to verify whether the system can operate safely when faced with fault conditions that are outside normal operating conditions, which we term *fault-mode events*. The flow of our approach is illustrated in Fig. 3. There are four steps that lead to determining the Petri net model G_f and its set of final markings $\mathscr{M}_f$, from which the test cases can be derived.

- **Step 1: Model the normal scenario**
  Construct the normal scenario model G_0.
- **Step 2: Add fault-mode transitions**
  Add a set of transitions that represent fault-mode events in G_0.
- **Step 3: Add safe-guard logic**
  Specify how the occurrence of fault-mode transitions is taken into account by the safe-guard logic, and construct the fault scenario model G_f.
- **Step 4: Safety analysis**
  Analyze the safety of firing sequences and determine the final markings set $\mathscr{M}_f$.

In the rest of this section, a detailed description of each step is provided, along with an example to demonstrate the proposed method. Note that [23], the system requirement specification of C2 Onboard equipment published by the China National Railway Corporation, is the primary reference throughout the procedure of scenario modeling.



t	Name	physical meaning when t fires
t_1	N_R	receive a notice message
t_2	E_R	receive an execution message
t_3	noB	detect no braking output
t_4	$Switch$	convert to C0

Figure 4: G_0: the normal scenario model.

3.2.1. Construct normal scenario model

First, based on the description of the nominal behavior of the system, we model the normal scenario as a Petri net system G_0. This net has a simply sequential structure. When G_0 is at its initial marking, the onboard equipment is initialized to start the train and movement on a specific line.

Example 1 Consider the automatic level-conversion scenario, whose Petri net model G_0 is shown in Fig. 4. It is clear that G_0 typically has a simple sequential structure. The initial marking is $[1\ 0\ 0\ 0\ 0]^T$, to indicate that the onboard equipment works at C2 while the train runs before entering the conversion zone. The markings of the places $C2\_1, C2\_2, C2\_3, C2\_4$ represent that the passage of the train along consecutive track segments while onboard equipment works at C2. When $C0$ is marked, it indicates that the onboard equipment works at C0, and the train has entered the C0 zone. The process of level-conversion corresponds to the firing of the transition sequence $\sigma_0 = t_1t_2t_3t_4$. ◇

3.2.2. Add fault-mode transitions

Referring to the system specifications, we do an exhaustive analysis by going through every transition in G_0, identifying the vulnerable transitions that have fault modes. The definitions of vulnerable transitions and corresponding fault-mode transitions are given as follows.

Definition 1. The set of *vulnerable transitions* $T_v \subseteq T$ defined for the normal scenario model G_0 is the set of transitions whose occurrence can be replaced by another faulty case. Given a vulnerable transition $t_i \in T_v$, we can associate t_i to a set of fault-mode transitions $T_f(t_i) = \{t_i', t_i'', \dots, t_i^{n_i}\}$ that represent different types of malfunction of t_i. We denote $T_f = \{t | \exists t_i \in T_v, t \in T_f(t_i)\}$ the set of all fault-mode transitions. ◇

The fault-mode transitions in $T_f(t_i)$ have exactly the same input and output as its corresponding vulnerable transition t_i. Fig. 5 illustrates the structure of a vulnerable transition t_i and its fault-mode transitions $T_f(t_i)$.

Fault-mode transitions, even if they belong to different vulnerable transitions' fault-mode transition sets, may have associations with the same interface or functional module of the

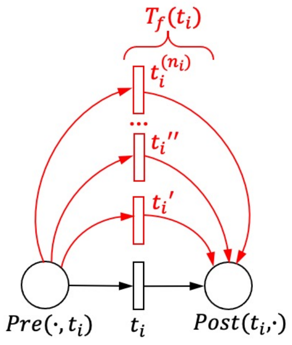


Figure 5: A vulnerable transition t_i and its fault-mode transition set $T_f(t_i)$.

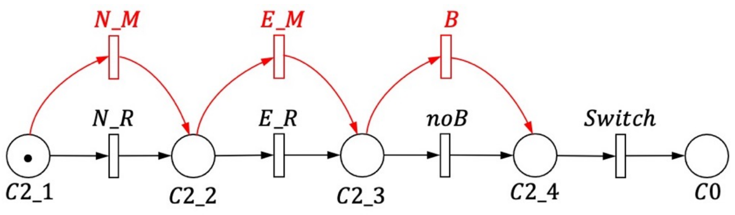
onboard equipment. Based on the causes of faults, we categorize all fault-mode transitions into several fault classes (denoted as q where q is in the set 1,2,3,...), namely $T_f = \bigcup_{j=1}^{q} T_f^j$.

When $q = 1$, it means that all the fault-mode transitions belong to the same fault class. When $q = |T_f|$, it means that each fault-mode transition is in a class by itself. Besides, the fault-mode transitions associated with the same vulnerable transition do not necessarily belong to the same fault class.

Example 2 Let us consider the model for level-conversion in Fig. 4. Consider transitions *N_R* and *E_R*, describing that in the nominal model, the train passes a balise while the onboard equipment correctly receives the message. In an abnormal scenario, the train fails to receive the message when passing the balise. By *Definition 1*, *N_R* and *E_R* are vulnerable transitions, and two fault-mode transitions *N_M* and *E_M* are added, which represent missing the notice message and missing the execution message, respectively. The transition *noB*, presenting that no braking has been detected, is vulnerable to the detection of the braking. This is modeled by the fault-mode transition *B*. The firing of transition *Switch* represents the fact that the onboard equipment executes the level-conversion from C2 to C0. Since the onboard equipment will certainly execute the level-conversion under appropriate conditions, we assume *Switch* is not vulnerable.

The model with the fault-mode transitions is shown in Fig. 6. For the vulnerable transition set $T_v = \{t_1, t_2, t_3\}$, fault-mode transition sets are $T_f(t_1) = \{t_5\}$, $T_f(t_2) = \{t_6\}$, $T_f(t_3) = \{t_7\}$, and thus $T_f = \{t_5, t_6, t_7\}$. Transitions *N_M* and *E_M* both mean missing the balise message, i.e., they all relate to the balise receiving function of the onboard equipment. Thus, they are classified in the same class, written in $T_f^1 = \{t_5, t_6\}$ and $T_f^2 = \{t_7\}$. ◇

The balise message is essentially a data input for the train, and the onboard equipment is required to check its legality. If it is not compliant with the encoding rules of balise telegrams, which is called illegal telegrams, the onboard equipment should reject such a message, determining a message loss. When focusing on the vulnerability of *N_R* or *E_R*, we can simultaneously test the balise receiving function of onboard equipment by adding various message failures in the fault-model, but this goes beyond the scope of this paper.



t	Name	physical meaning when t fires
t_5	N_M	miss the notice message
t_6	E_M	miss the execution message
t_7	B	detect braking output

Figure 6: G_0 with the fault-mode transitions.

3.2.3. Add safe-guard logic

Including T_f in G_0 enables all potential fault modes to be considered. Meanwhile, we need to further specify how the occurrence of these abnormal events affects the operational scenario or functional execution result.

To address this problem, we model the *safe-guard logic* which restricts the behavior of the system in case of faulty execution, as per the provided specification or the *fail-safe principle*. In particular, if a fault-mode event and its consequence have been explicitly stated in the requirement documents, the added safe-guard logic should comply with the specifications outlined in those documents. When such specifications are not provided, we adhere to the fail-safe principle, which is a crucial concept in railway signaling systems. The fail-safe principle mandates that the train should continue to operate safely in the event of a failure or malfunction. This may require actions such as applying brakes, initiating a stop, interrupting the execution of functions, or transferring control of the train to the driver.

There are various fault measuring and contingency mechanisms within the onboard equipment. Due to space limitations, here we only discuss a typical one that works by prohibiting critical events when a certain number of failures have been detected.

Definition 2. Given a fault class T_f^j, let $t_{cr}^j \in T$ be a normal transition, the *critical blocking policy* is to block t_{cr}^j whose firing should be blocked after k ($k \leq |T_f^j|$) or more fault-mode transitions in this class occur. ◇

The critical blocking policy can be implemented by a simple safe-guard logic defined as follows.

- Add a new place named *guard place* p_g^j.
- Add a post-arc from each transition in T_f^j to p_g^j.
- Add $|T_f^j| - k + 1$ pre-arcs from p_g^j to t_{cr}^j.
- The initial marking of the guard place is $M(p_g^j) = |T_f^j|$.

When k or more fault-mode transitions in T_f^j occur, the guard place will block critical transition t_{cr}^j. Fig. 7 illustrates the structure of a guard place for a critical blocking policy. The normal

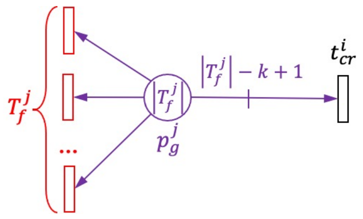


Figure 7: Structure of a guard place for a critical blocking policy.

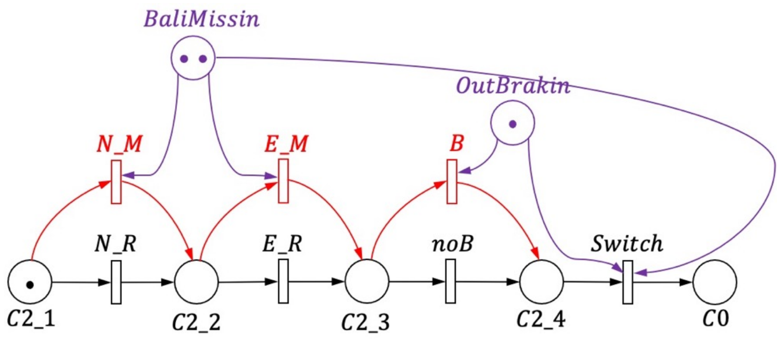


Figure 8: G_f: the fault scenario model.

scenario model G_0 with the fault-mode transitions and the safety-guard logic is denoted by G_f and called a *fault scenario model.*

Example 3 Continue with *Example 2*, to demonstrate the effect of missing balise messages and braking. The obtained fault scenario model G_f is shown in Fig. 8. Two safe-guard logics are considered here:
- safe-guard logic 1: no conversion if both two balise messages are missed. A guard place *BaliMissin* (p_6) is added to describe the critical blocking policy for the fault class T_f^1 and the normal transition to be blocked is *Switch* with $k = 2$.
- safe-guard logic 2: no conversion if the train is braking. A guard place *OutBrakin* (p_7) is added to describe the critical blocking policy for the fault class T_f^2 and the normal transition to be blocked is *Switch* with $k = 1$. ◇

3.2.4. Safety analysis on final markings

This section analyzes the evolution of G_f from the viewpoint of safety. First, we introduce a new place that will be marked whenever a fault occurs. Then we compute the reachability graph of the new model and identify the final markings and the firing sequences reaching them.

The *explicit fault scenario model* $\hat{G}_f$ is obtained by adding a new place p_f in G_f, which we call *flag place.* Such a place has initial marking $M_0(p_f) = 0$, and has an input arc from each transition in T_f.

Fig. 9 illustrates the structure of the flag place p_f and all the fault-mode transitions in T_f. We can monitor the firing of any fault-mode transition by observing the token change in p_f.

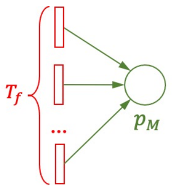


Figure 9: The fault flag place p_f and all the fault-mode transitions.

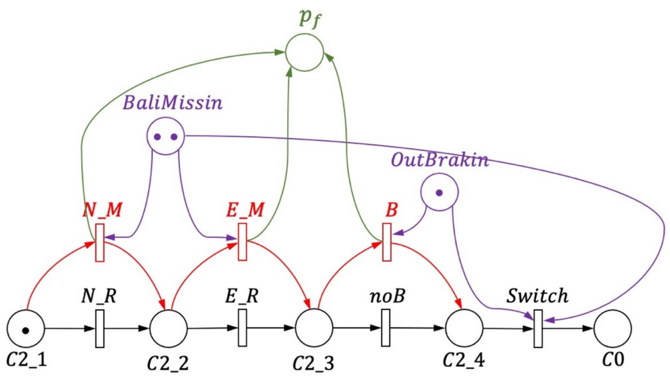


Figure 10: $\hat{G}_f$: the explicit fault scenario model.

When $M(p_f) > 0$, it means that one or more fault-mode transitions have fired. Otherwise, no fault-mode transitions have fired.

There are many techniques to analyze the evolution of the Petri net model, such as the state equation and the reachability graph. In this paper, we assume $\hat{G}_f$ is bounded and investigate its reachability problem by constructing the reachability graph. Then two types of final markings in $\hat{G}_f$ are defined as follows.

Definition 3. Given a model $\hat{G}_f$, its set of *normal final markings* $\mathscr{M}^n_f$ is defined as $\mathscr{M}^n_f = \{M \in R(\hat{G}_f) \mid M(p_f) = 0 \textit{ and } (\nexists t \in T)M[t\rangle\}$. The set of *fault final markings* $\mathscr{M}^a_f$ is defined as $\mathscr{M}^a_f = \{M \in R(\hat{G}_f) \mid M(p_f) > 0 \textit{ and } (\nexists t \in T)M[t\rangle\}$. The set of all final markings $\mathscr{M}_f = \mathscr{M}^n_f \bigcup \mathscr{M}^a_f$. ◇

Example 4 The explicit fault scenario model $\hat{G}_f$ for *Example 3* is shown in Fig. 10. Assume $M_0 = [1\ 0\ 0\ 0\ 0\ 2\ 1\ 0]^T$ (p_f is p_8), the reachability graph of $\hat{G}_f$ is depicted in Fig. 11, from which we can derive the two sets of final markings $\mathscr{M}^n_f = \{M_4\}$, $\mathscr{M}^a_f = \{M_5, M_9, M_{10}, M_{12}, M_{13}\}$. ◇

3.3. Deriving Test Cases

In this section, we extract test cases from the acquired fault scenario model. First, we provide a formalization of the test case utilizing Petri net semantics, building upon the model $\hat{G}_f$ and its previously introduced set of final markings. Subsequently, we establish a correspondence

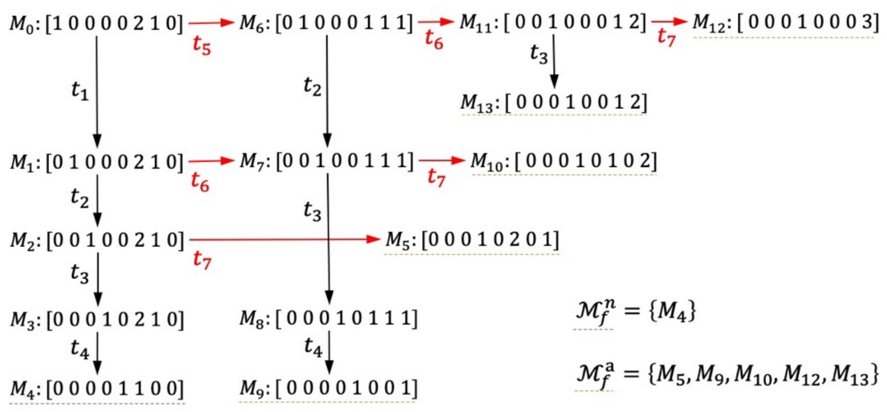


Figure 11: Reachability graph of $\hat{G}_f$.

Table 1
Mapping from a *TC* to a test case

element in a *TC*	test case component
Initial marking	Preset condition
Transition sequence	Operation sequence
Final marking	Expected result

relationship between the formalized test case and the real test case that is prepared for execution by the test engineer.

Definition 4. Given an explicit fault scenario model $\hat{G}_f$ and its set of final markings $\mathscr{M}_f$, a *test case* is defined as a three-tuple $TC = (M_0, \sigma, M)$, where M_0 is the initial marking, σ is the firing sequence, and $M \in \mathscr{M}_f$ is a final marking reached from M_0 by firing σ. The set of all test cases is defined as $\mathbb{X} = \{TC = (M_0, \sigma, M) \subseteq R(G) \times T^* \times R(G) | \exists \sigma \in L(\hat{G}_f), M_0[\sigma\rangle M \textit{ and } M \in \mathscr{M}_f\}$.

A test case $TC = (M_0, \sigma, M) \in \mathbb{X}$ is called *normal* if its final marking M is normal, otherwise it is called *faulty*. ◇

Referring to the Union Industry of Signaling (UNISIG) scheme on ERTMS/ETCS system testing, a widely recognized technical document introduced by the European Union Agency for Railways, a test case consists of several essential components, including a test name, test objective, operation sequence, preset condition, and expected result [24]. Among these components, the preset condition (the initial state of the system under test before executing the case), the operation sequence (a series of events or operations that the test engineer need to perform), and the expected result (the expected final outcome after performing actions specified in the operation sequence) are of the utmost importance and constitute the backbone of a test case. In particular, the operation sequence is the element that makes a test case distinct from others. The relationship between a test case generated by an explicit fault scenario model and these primary components can be illustrated in Table 1.

Example 5 Continuing with *Example 4*, in the reachability graph of $\hat{G}_f$, there are 8 transition sequences leading to 6 final markings. Accordingly, 8 test cases are obtained, and they apply the same preset condition $M_0 = [1\ 0\ 0\ 0\ 0\ 2\ 1\ 0]^T$, which implies that the train runs within the C2 zone, nearing the border of the C0 zone, while functioning at level-2. The full test case set can be found in the appendix. Besides the normal test case TC_1, seven faulty test cases $TC_2 - TC_8$ are obtained. In particular, final markings in test cases TC_1, TC_3 and TC_4 imply the successful execution of level-conversion, and the three test cases cover the exceptions where one of the two balise messages is missing. Whereas TC_2, TC_5, TC_6, TC_7 and TC_8 represent the scenarios of unsuccessful execution, covering the combinatorial occurrences of fault-mode events including braking output and missing balise messages. $\diamond$

After obtaining the test cases $\mathbb{X}$, additional document work is necessary for executing them in a real test environment, such as compile to scripts, which however is beyond the scope of this paper. However, the structure provided by the obtained test cases already offers a solid foundation for formalizing and organizing the subsequent version of the mature test case set.

We point out that the size of a reachability graph is exponential in the size of the net and its initial marking. In the worst case, we need to find all the firing transition sequences leading to the final markings, which could be done by exhaustively computing all corresponding directed paths in the reachability graph. A promising topic for future studies is that of finding more efficient algorithms for generating test cases using Petri nets.

4. Conclusion

This paper provides an approach to formalize the test scenario containing abnormalities. We apply Petri nets to describe the system behavior, calculate the reachability graph to derive test cases that cover various types of scenarios. We take a critical scenario in Chinese high-speed rail as an example to illustrate the proposed approach.

By designing test cases in the appearance of faults, the approach allows a better understanding of system specifications, which is important for the testing of safety-critical systems, to discover potential drawbacks aroused by accidental failure.

Acknowledgments

This work was supported by the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 1, INVESTIMENTI 3.4 e 4.1 – Decreto del Ministero dell'Università e della Ricerca n.351 del 09/04/2022) within the Italian National Ph.D. Program in Autonomous Systems (DAuSy).

References

[1] L. Zhou, J. Dang, Z. Zhang, Fault classification for on-board equipment of high-speed railway based on attention capsule network, International Journal of Automation and Computing 18 (2021) 814–825.

[2] Y. Liu, T. Tang, Research on the method of interoperability test for the onboard equipment of CTCS-3 train control system of chinese railway, in: 2011 Tenth International Symposium on Autonomous Decentralized Systems, 2011, pp. 415–419.

[3] R. Nardone, S. Marrone, U. Gentile, A. Amato, G. Barberio, M. Benerecetti, R. De Guglielmo, B. Di Martino, N. Mazzocca, A. Peron, G. Pisani, L. Velardi, V. Vittorini, An OSLC-based environment for system-level functional testing of ERTMS/ETCS controllers, Journal of Systems and Software 161 (2020) 110478.

[4] R. Huang, C. Rao, Y. Lei, J. Guo, Y. Zhang, Applying combinatorial testing to high-speed railway automatic train protection system, in: 2022 IEEE Int. Conf. on Software Testing, Verification and Validation Workshops (ICSTW), 2022, pp. 49–56.

[5] C. Rao, J. Guo, N. Li, Y. Lei, Y. Zhang, Y. Li, Safety-critical system modeling in model-based testing with hazard and operability analysis, in: 2018 IEEE Int. Conf. on Software Quality, Reliability and Security (QRS), 2018, pp. 397–404.

[6] J. Zhu, X. Huang, F. Wang, L. Zhou, Research on automated testing scheme of exceptional scenarios in unattended train operation of urban transit, in: 2021 6th Int. Conf. on Intelligent Transportation Engineering (ICITE 2021), Springer, 2022, pp. 558–568.

[7] E. Sarmiento, J. C. Leite, E. Almentero, G. Sotomayor Alzamora, Test scenario generation from natural language requirements descriptions based on Petri-nets, Electronic Notes in Theoretical Computer Science 329 (2016) 123–148. CLEI 2016 - The Latin American Computing Conference.

[8] A. Piccolo, V. Galdi, F. Senesi, R. Malangone, Use of formal languages to represent the ERTMS/ETCS system requirements specifications, in: 2015 Int. Conf. on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS), 2015, pp. 1–5.

[9] Y. Wang, L. Chen, D. Kirkwood, P. Fu, J. Lv, C. Roberts, Hybrid online model-based testing for communication-based train control systems, IEEE Intelligent Transportation Systems Magazine 10 (2018) 35–47.

[10] A. Ait Wakrime, Y. Ouhammou, Advances in modeling, verification and testing of safety-critical software architectures, 2022.

[11] H. Su, M. Chai, H. Liu, J. Chai, C. Yue, A model-based testing system for safety of railway interlocking, in: 2022 IEEE 25th Int. Conf. on Intelligent Transportation Systems (ITSC), 2022, pp. 335–340.

[12] A. Bucaioni, F. D. Silvestro, I. Singh, M. Saadatmand, H. Muccini, T. Jochumsson, Model-based automation of test script generation across product variants: a railway perspective, in: 2021 IEEE/ACM Int. Conf. on Automation of Software Test (AST), 2021, pp. 20–29.

[13] Y. Sun, K. Li, L. Yuan, Mutation-based test case generation of train-ground transmission function for CTCS, in: Proceedings of the 4th Int. Conf. on Electrical and Information Technologies for Rail Transportation (EITRT) 2019, Springer, 2020, pp. 133–141.

[14] L. Yao, Z. Xiaoxia, Z. Yadong, G. Jin, G. Hao, Towards a test paths generation method for CTCS level transition, in: MATEC Web of Conferences, volume 325, EDP Sciences, 2020, p. 01001.

[15] P. Barger, W. Schön, M. Bouali, A study of railway ERTMS safety with Colored Petri Nets, in: G. S. . M. e. Bris (Ed.), The European Safety and Reliability Conference (ESREL'09), volume 2, Taylor & Francis Group, Prague, Czech Republic, 2009, pp. 1303–1309.

[16] C. Zhang, D. Wu, X. Li, Safety analysis of train-ground communication failure based on

unfolding of Petri nets, in: 2022 China Automation Congress (CAC), 2022, pp. 5794–5798.
[17] D. Wu, D. Lu, T. Tang, Qualitative and quantitative safety evaluation of train control systems (CTCS) with stochastic colored Petri nets, IEEE Transactions on Intelligent Transportation Systems 23 (2022) 10223–10238.
[18] A. Giua, C. Seatzu, Modeling and supervisory control of railway networks using Petri nets, IEEE Transactions on Automation Science and Engineering 5 (2008) 431–445.
[19] J. Luo, M. Zhou, J.-Q. Wang, A place-timed Petri net-based method to avoid deadlock and conflict in railway networks, IEEE Transactions on Intelligent Transportation Systems 23 (2022) 10763–10772.
[20] H. Lan, Y. Tong, C. Seatzu, Crucial states estimation in radio block center handover using Petri nets with unobservable transitions, IEEE Transactions on Automation Science and Engineering 19 (2022) 1268–1276.
[21] M. S. Durmus, İ. Ustoglu, R. Y. Tsarev, M. Schwarz, Modular fault diagnosis in fixed-block railway signaling systems, IFAC-PapersOnLine 49 (2016) 459–464.
[22] M. Lawrence, R. Bullock, Z. Liu, China's high-speed rail development, World Bank Publications, 2019.
[23] System Requirements Specification of the CTCS-2 Train Control System, 2014.
[24] UNISIG SUBSET 076-4-l: Test sequence generation: Methodology and Rules, 2008.

Appendix: Full Test Case Set in Example 5

1. $TC_1 = (M_0, \sigma_1, M_4)$, $\sigma_1 = t_1t_2t_3t_4$.
• Operation sequence:
a. receive the level-conversion notice message
b. receive the level-conversion execution message
c. detect no braking output
d. convert to C0
• Expected result:
The onboard equipment works at C0.

2. $TC_2 = (M_0, \sigma_2, M_5)$, $\sigma_2 = t_1t_2t_7$.
• Operation sequence:
a. receive the level-conversion notice message
b. receive the level-conversion execution message
c. detect the braking output
• Expected result:
The onboard equipment works at C2.

3. $TC_3 = (M_0, \sigma_3, M_9)$, $\sigma_3 = t_5t_2t_3t_4$.
• Operation sequence:
a. miss the level-conversion notice message
b. receive the level-conversion execution message
c. detect no braking output
d. convert to C0

• Expected result:
The onboard equipment works at C0.

4. $TC_4 = (M_0, \sigma_4, M_9)$, $\sigma_4 = t_1 t_6 t_3 t_4$.
•Operation sequence:
a. receive the level-conversion notice message
b. miss the level-conversion execution message
c. detect no braking output
d. convert to C0
• Expected result:
The onboard equipment works at C0.

5. $TC_5 = (M_0, \sigma_5, M_{10})$, $\sigma_5 = t_5 t_2 t_7$.
• Operation sequence:
a. miss the level-conversion notice message
b. receive the level-conversion execution message
c. detect the braking output
• Expected result:
The onboard equipment works at C2.

6. $TC_6 = (M_0, \sigma_6, M_{10})$, $\sigma_6 = t_1 t_6 t_7$.
• Operation sequence:
a. receive the level-conversion notice message
b. miss the level-conversion execution message
c. detect the braking output
• Expected result:
The onboard equipment works at C2.

7. $TC_7 = (M_0, \sigma_7, M_{12})$, $\sigma_7 = t_5 t_6 t_7$.
• Operation sequence:
a. miss the level-conversion notice message
b. miss the level-conversion execution message
c. detect the braking output
• Expected result:
The onboard equipment works at C2.

8. $TC_8 = (M_0, \sigma_8, M_{13})$, $\sigma_8 = t_5 t_6 t_3$.
• Operation sequence:
a. miss the level-conversion notice message
b. miss the level-conversion execution message
c. detect no braking output
• Expected result:
The onboard equipment works at C2.