Qualitative and Quantitative Safety Evaluation of Train Control Systems (CTCS) With Stochastic Colored Petri Nets

Daohua Wu, Debiao Lu, and Tao Tang, *Senior Member, IEEE*

***Abstract*— Currently qualitative as well as quantitative safety analysis of railway systems are usually conducted through Fault Tree Analysis (FTA) and Event Tree Analysis (ETA). FTA and ETA display the causalities and consequences of hazards or accidents by means of linear event sequences, which makes it difficult to incorporate non-linear relationships such as feedback. The repair of failure system components is not considered in traditional FTA and ETA. Moreover, quantitative safety evaluation by FTA and ETA requires that failure events should be statistically independent. Considering these issues, we propose an approach of conducting qualitative as well as quantitative safety evaluation of the CTCS-3 with stochastic Coloured Petri Nets (CPNs). The hierarchical CPN model of the CTCS-3 takes the scenarios, movement of trains, the failure and repair of system components into account. The occurrence times of hazardous events of a CTCS-3 system is quantitatively evaluated with the numerical data collected during the simulation of the system model. The evaluation results demonstrate the feasibility of the proposed approach.**

***Index Terms*— CTCS, operation scenario, quantitative safety evaluation, system modelling, Coloured Petri Nets.**

I. Introduction

IN THE railway domain, qualitative and quantitative safety analyses are usually conducted by using the methods of Fault Tree Analysis (FTA) [1] and Event Tree Analysis (ETA) [2]. They are highly recommended for software assessment by the CENELEC standard EN 50128 [3]. In [4]–[10], a number of (extended/varied) FTA methods have been introduced to analyse the railway systems. Nevertheless, fault trees and event trees display the causalities and consequences of hazards or accidents in linear event sequences, which makes it difficult to incorporate non-linear relationships such as feedback [11]. The existence of non-linear or feedback relationships between system components or subsystems is one of the noteworthy properties of modern complex systems such as train control systems. For these systems, hazards could arise due to unsafe interactions between system components (even when these components have not necessarily failed) [12]. Moreover, traditional FTA and ETA do not consider the repair of failure system components, and quantitative safety evaluation by FTA and ETA requires that the failure events are statistically independent [13].

In contrast to FTA and ETA, the institute of iVA at TU Brauschweig proposed the ProFunD (Process-Functionality-Dependability) approach to risk and availability/dependability analysis of railway operation control systems with low-level stochastic Petri nets. With the ProFunD approach, the operational process controlled by a railway operation control system (e.g., a level crossing control system) is modelled taking undesired operational events into consideration and the functional design is carried out as an extension of the process model fulfilling the control tasks [14]–[16]. The ProFunD approach has been further applied as the basis of the ICE standard ICE 62551 (*Analysis techniques for the dependability - Petri net techniques*). Work [17] extended CPN to Extended Coloured Stochastic Petri Net (ECSPN) to facilitate availability modelling of technical systems incorporating the component age and queuing aspects into the system models. Similar to the ProFunD approach, we proposed a formal model-based approach for quantitative safety analysis using timed CPNs in [18]. The proposed approach was illustrated by a case study of a railway level crossing control system. The CPN model of the level crossing system was established based on the concept of scenario that is described by timed Message Sequence Charts (MSCs). There are 14 operation scenarios according to the Chinese standard specifications of the CTCS-3. In our previous work, however, only a single operation scenario was considered. Therefore, it is of interest to consider multiple operation scenarios while establishing the CTCS-3 model for qualitative and quantitative safety evaluation.

For Petri-net-based modelling of train control systems, in [19], the authors adopted CPNs to model the ETCS (European Train Control System). The CPN model of the ETCS is divided into different modules and each module represents a system component or a subsystem. For the modules of the on-board subsystem and the RBC (Radio Block Center), they are vertically decomposed into scenario nets, function nets and additional nets which are added for the purpose of independent simulation. In [20], similar approaches were employed to model the ETCS, but they presented more detailed nets with respect to the interfaces of each component

Manuscript received 25 August 2020; revised 29 March 2021 and 25 May 2021; accepted 2 June 2021. Date of publication 23 June 2021; date of current version 9 August 2022. This work was supported in part by the National Natural Science Foundation (NNSF) of China under Grant 61803020 and Grant 61803019 and in part by the Fundamental Research Funds for the Central Universities under Grant 2019JBM329. The Associate Editor for this article was R. Goverde. *(Corresponding author: Daohua Wu.)*

Daohua Wu is with the National Research Center of Railway Safety Assessment, Beijing Jiaotong University, Beijing 100044, China (e-mail: wudaohua@bjtu.edu.cn).

Debiao Lu is with the School of Electronic Information Engineering, Beijing Jiaotong University, Beijing 100044, China.

Tao Tang is with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China.

Digital Object Identifier 10.1109/TITS.2021.3088136

and the data exchanged between different components. In [21], we developed the CPN model of the on-board subsystem of a satellite-based train control system by integrating several operation scenarios. Works [19], [20] developed CPN models directly from textual specifications, which makes the CPN models unintelligible for other work teams or related parties for the lack of intermediate formal/semi-formal specifications. The work [21] constructed system models with the aid of sequence diagrams. In summary, all these works model train control systems from the perspective of system design. More importantly, as a key part of the system model of a train control system for quantitative safety evaluation, the module of the controlled process (i.e., the train movement) is not included.

Given the aforementioned issues, we present a qualitative as well as quantitative safety evaluation approach for the CTCS-3 using stochastic Coloured Petri Nets: firstly, the operation scenarios of the system are described by UML sequence diagrams; secondly, the hierarchical timed CPN model of the system is developed based on the system control structure, operation scenarios, functional specifications and failure injection of system components; thirdly, qualitative analysis is carried out by exploring standard state space reports and applying standard as well as non-standard queries to the state space of the (untimed) system model; finally, the quantitative safety evaluation of the system can be accomplished based on the numerical data collected during the simulation of the system model. Compared to the existing methods such as the ProFunD approach, FTA and ETA, our approach has the following advantages: (1) The modelling language of CPNs is adopted as the means of description for the system under consideration, which gives us a powerful tool to model large-scale and complex systems such as the CTCS-3; (2) The system model is established based on the control structure of the system, which makes it possible capture hazards/accidents caused by non-linear related events like unsafe interactions between the system components; (3) Operation scenarios are taken to develop the system model, which helps to generate a more realistic system model.

This paper is organized as follows. The methodology of modelling and analysis with timed CPNs is presented in Section II. Section III focus on the modelling of the CTCS-3. Section IV presents the qualitative and quantitative analysis of the CTCS-3 model and we conclude our work in Section V.

II. Modelling and Analysis

A. Scenario and System Control Structure

In the railway domain, operation scenarios have been widely accepted as an effective means for specifying system requirements. Scenarios describe how users, system components and the environment interact in order to provide system-level functionality [22]. They are partial descriptions of system behaviours and need to be synthesized to present an overall behaviour of the system. In the specification of *The General Technical Solution for the CTCS-3*, 14 operation scenarios are specified: Registration and Startup, Logout, Movement Authority, Shunting, RBC Switching, Temporary Speed Restriction, etc. In addition, 7 operation modes are specified for the on-board subsystem in this specification: Full Supervision mode (FS), Shunt mode (SH), Isolation mode (IS), Stand By mode (SB), Call On mode (CO), On Sight model (OS) and Sleeping mode (SL). Under each mode, specific activities are specified for the on-board subsystem (including the driver). The operation scenarios are synthesized by the switching of the operation mode of the on-board subsystem. Scenarios can be described and represented in various forms, for instance, natural languages (texts), (semi-formal or formal) diagrams and tables. Graphical representations are often better understood by a variety of readers such as users and design engineers. In this work, we choose UML sequence diagrams [23], a widespread notation for scenarios, as the graphical representations of scenarios. To present an overall behaviour of the CTCS-3, different operation scenarios need to be integrated and they are synthesized by the switching of the operation mode of the on-board subsystem. We, therefore, label the corresponding lifeline of the on-board subsystem in sequence diagrams with explicit operation modes (see Fig. 1 and Fig. 2).

As previously stated, we would like to include the train movement in the system model. For this purpose, we construct the top level of the hierarchical CPN model of the CTCS-3 based on its control structure (or called control loop). A system control structure describes the system in terms of a hierarchy of control based on adaptive feedback mechanisms [11]. Fig. 3 shows a typical system control structure. The controller obtains the information about the state of the process from measured variables (feedback) and uses this information to initiate control actions by manipulating controlled variables in order to keep the process operating within predefined limits. For complex systems, hierarchical control structures in which a higher-level controlled process could be the controller of a lower-level process, is desirable.

B. System Modelling With Timed Coloured Petri Nets

1) Coloured Petri Nets: Coloured Petri Nets (CPNs or CP-nets) [24] are high-level Petri nets. Similar to low-level Petri nets [25], CPN models are directed graphs containing two types of nodes: *places* (ellipses or circles) and *transitions* (rectangular boxes), where edges that connect only nodes of different types are denoted as *arcs*. Each place is assigned a type called *color set* which determines the set of *token colors* (data values) that the tokens on that place are allowed to have. With the CPN modelling language, it is possible to work with different levels of detail and abstraction because of the capability of specifying hierarchically structured models of CPNs. These hierarchical models allow a module to have submodules, a set of modules to form a new module, and the reuse of submodules in different parts of the model. In CPN models, a module is usually represented by a substitution transition on its superior hierarchical level. A *substitution transition* has a rectangular substitution tag positioned next to it. The substitution tag contains the name of a submodule which is related to the substitution transition. Additionally, fusion sets can be utilized for relating/synchronizing modules. A *fusion set* is comprised of a group of fusion places, which

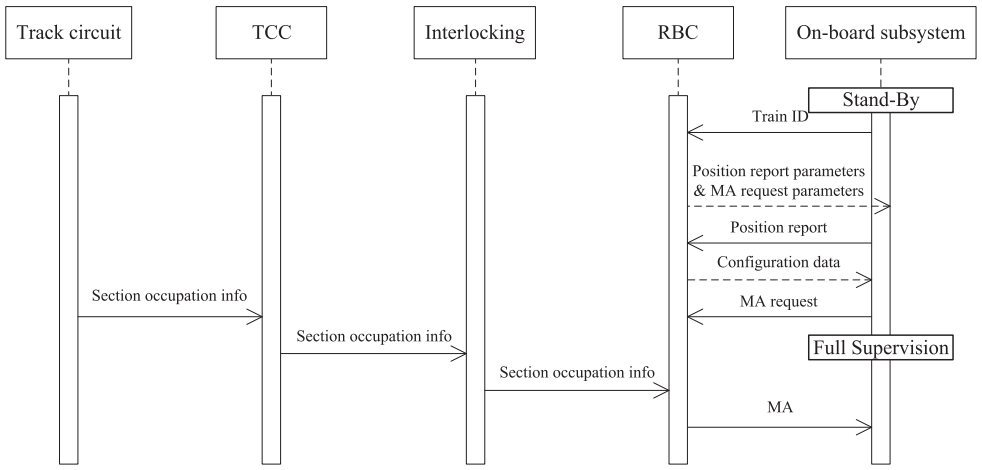

Fig. 1. The operation scenario of REGISTRATION AND STARTUP.

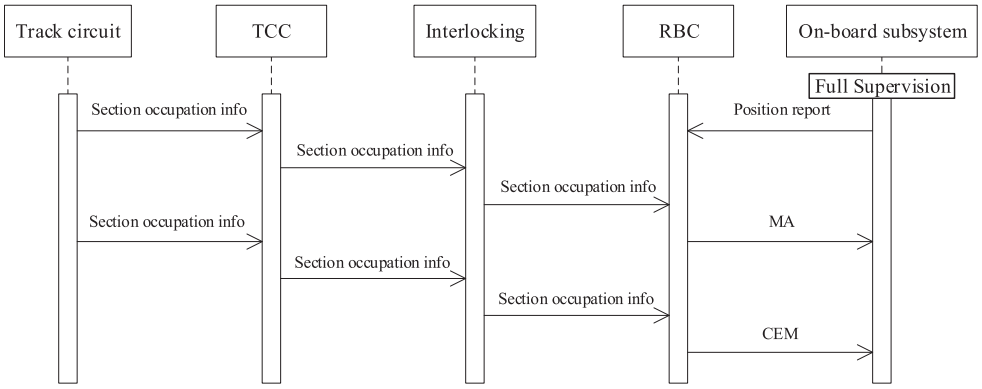

Fig. 2. The operation scenario of MOVEMENT AUTHORITY.

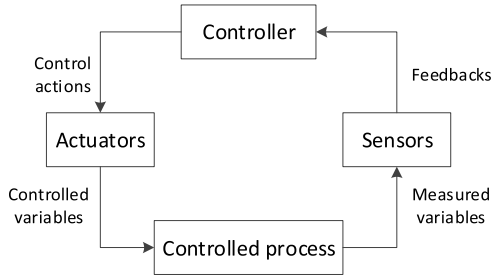

Fig. 3. A typical system control structure.

allows the places in different modules to be glued together into one compound place across the hierarchical structure of the model. The *fusion places* that are members of a fusion set represent a single compound place. For example, if a fusion place in a fusion set is marked by a token color, then the rest fusion places in that fusion set will be marked by the same token color.

In timed CPN models, in addition to the token color, tokens can carry a second value called *timestamp*. The timestamps of tokens are written after the symbol "@". The timestamp of a token specifies the time at which the token is available for being used, i.e., the time at which it can be removed by an occurring transition. A timed CPN model has a global clock representing the model time. In a hierarchical timed CPN model, there is a single global clock shared by all the modules. When a transition occurs, it is necessary to calculate the timestamps that will be given to the output tokens. A time delay inscribed on a transition applies to all the output tokens generated by that transition, whereas a time delay inscribed on an output arc is applicable only to the tokens generated on that arc. Thus, the timestamp given to the tokens generated on an output arc is the sum of the value of the global clock, the result of evaluating the time delay inscription of the transition and the result of evaluating the time delay inscription of the arc. More information about (timed) CPNs can be found in [24]. It is noteworthy that the application of timed Petri nets as a means of description is highly recommended within the CENELEC standard EN50128.

CPNs are a graphical and mathematical means of description that is well-known in describing systems characterized as being concurrent, asynchronous and distributed (e.g., trains control systems). With CPNs, it is possible to work on different levels of detail and abstraction by specifying hierarchical CPN models. This will be of great benefit when modelling large-scale and complex systems. In addition, mature tools (e.g., CPN Tools [26] adopted in this work) for editing CPN models and associated techniques (e.g., model checking) for model analysis are available.

2) Top Level of the System Model: For the top level of the hierarchical CPN model of the CTCS-3, each substitution transition corresponds to a component of the system control structure. The commands/feedbacks in the system control structure are represented by the token colors that could be resided on the places connected with the substitution transitions.

3) Module of the Train Movements (Second Level of the System Model): Regarding the safety analysis of the CTCS-3 in this work, the rear-end collision accident between two trains, one of the major hazards (top event of a fault tree), is considered. The moving processes of these two trains are thus modelled based on the following assumptions: one train (say Train-123) is moving under the control of the CTCS-3, and the other train (say Train-xxx) imitates the trains that are running ahead and/or behind Train-123, representing the moving environment of Train-123; we assume that the two trains are always running at the same direction and Train-xxx will always operate safely (i.e., it will never collide with the rear of the Train-123); the appearances of Train-xxx on the track ahead or behind Train-123 are random. Note that the naming of the trains as "Train-123" and "Train-xxx" has no specific meaning here.

4) Timing Configuration: Assuming that the length of a block section is l_S (where $l_S = 2km$), the length of a train is l_T (where $l_T = 200m$), Train-123 is running at a constant speed of $v_{T_{123}}$ (where $v_{T_{123}} = 300km/h$) while Train-xxx at a constant speed of $v_{T_{xxx}}$ (where $v_{T_{xxx}} = 200km/h$), then the duration of Train-123 running from one block section to another is $t_1 = (l_S + l_T)/v_{T_{123}} \approx 26s$, and the duration of Train-xxx running from one block section to another is $t_2 = (l_S + l_T)/v_{T_{xxx}} \approx 40s$.

Assuming that the number of trains (except Train-123) running on the rail track in a time unit is a random variable χ which is subjected to *Poisson distribution*, and the average of χ is λ (i.e. $\lambda = E(\chi)$), the time interval between two consecutive trains showing up on the rail track is a random variable τ that is *exponentially distributed* with the constant parameter λ. Thus the mean time interval between two consecutive trains showing up on the rail track is $E(\tau) = \frac{1}{\lambda}$.

In this work, timers with the time increment of one time unit are developed to regulate the movement of trains. Fig. 4 shows the timed CPN model of a timer. In the initial state, the place `P1` is marked by a token color "$0@+0$" (with a timestamp of 0) and the transition `T1` is enabled. After the firing of the transition `T1`, the token color "$0@+0$" on the place `P1` is removed and added to the place `P2` and the transition `T2` becomes enabled. After firing the transition `T2`, the token color "$0@+0$" on the place `P2` will be removed and a token color "$1@+1$" will be added to the place `P1` since the time delay inscription "$@+1$" is added to the transition `T2` and the variable $n+1$ is assigned on the directed arc from the transition `T2` to the place `P1`. We can observe that after a sequenced firing of the transitions `T1` and `T2`, the value of the token color on the place `P1` is increased by one and the timestamp of this token color is increased by one as well. In other words, the value of the token color on the place `P1` is always equal to the timestamp of the token color. Therefore, we can take the value of the token color on the place `P1` as the (global) time of the system model. Note that the transitions `T1` and `T2` have the lowest firing priority in the system model.

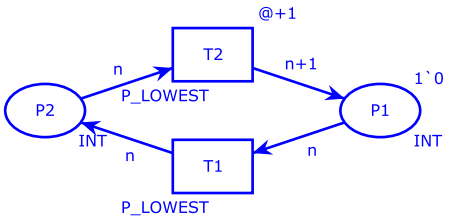

Fig. 4. The timed CPN model of a timer.

C. Failure Injection

For safety analysis, hazardous states/events must be identified in the first place. Component failures are therefore injected into the system model. In general, we distinguish the state (or called functionality condition) of a component between "OK" and "failed". Namely, a component is in either one of the two states at any moment. Moreover, the failed states could be further divided into "fail-safe" states and "fail-hazardous" states. If a failed state can be detected in time/ on demand, or it cannot be detected in time but the outcome/output of the component in this failed state will not cause hazards/accidents on the system level, this failed state can be considered as a fail-safe state. Otherwise, it is a fail-hazardous state. An undetected fail-safe state could be detected after some time (e.g., via periodic maintenance). A (undetected) fail-hazardous state will be considered as turning into a fail-safe state after it is detected. For a repairable component, the repairing work is done in the (detected) fail-safe state.

Fig. 5 shows the CPN model of a system component illustrating the failure injection. The output of the component depends on the input as well as the state of the component. If the component is in the fine state, i.e., the place `OK` is marked with a token color "e", an expected output will be generated by firing the transition `Generation1` when an input is received; if the component is in the fail-safe state, i.e., the place `Fail_safe` is marked with a token color "e", a degraded (but safe) output will be produced by firing the transition `Generation2`; if the component is in the fail-hazardous state, i.e., the place `Fail_hazardous` is marked with a token color "e", an unwanted (and hazardous) output will be provided by firing the transition `Generation3`.

Assuming that the failure time (also called "time to failure") of a system component is the random variable ξ that is exponentially distributed with the constant parameter λ (failure rate), the Mean Time To Failure (MTTF) of the component is: $MTTF = E(\xi) = \frac{1}{\lambda}$. For a repairable system component, the repair time (also called "time to repair") is often assumed to be a random variable η that is exponentially distributed or a constant time duration. If the repair time is exponentially distributed with the repair rate μ, the Mean Time To Repair (MTTR) is: $MTTR = E(\eta) = \frac{1}{\mu}$.

In Fig. 5, in the initial state, the places `OK` and `S1` are marked by the timed token color "e@0" (i.e., with a timestamp of 0) and the transition `Operate` is enabled, implying that the component is in the fine state. After the firing of the transition `Operate`, the token color "e" on the place `S1` will be removed and added to the place `S2`. The timestamp of this token color will be the sum of the time at which the transition occurs and the evaluation of the time delay expression "$@+$ Fail()" inscribed on the transition. If the component fails

by firing the transition `Fail`, the place `Safe/Hazardous` will be marked by an integer token color whose value is equal to the evaluation result of the function *FailSafeHazardous()* (see its definition below). The function *FailSafeHazardous()* is evaluated to be a random integer σ between 0 and 100. If σ is between 10 and 95, the transition `Fail1` will be fired and a token color "e" will be put onto the place `Fail_safe`. The firing of the transition `Fail1` implies that a failure occurred and is detected on demand. So the component will turn into the fail-safe state. If σ is equal to or greater than 95, the transition `Fail2` will be fired, which means a safe failure has occurred and is detected after some time (by firing the transition `Detect1`). If σ is smaller than or equal to 10, the transition `Fail3` will be fired and a token color "e" will be put onto the place `Fail_hazardous`. The firing of the transition`Fail3` indicates that an unsafe failure has occurred and the component has turned into the fail-hazardous state. After the failure has been detected by firing the transition `Detect2`, the component will be transferred to the fail-safe state. Since σ is a random integer between 0 and 100, the probabilities of firing the transitions `Fail1`, `Fail2`, and `Fail3` are 85%, 5% and 10%, respectively. When a fail-safe state is reached, the state of the component will be resumed to OK state after repairing.

The function *Fail()* takes a unit (written "()") as an argument and it is evaluated to be a random variable that is exponentially distributed with the parameter $failurerate$. The function *Detect()* is evaluated be a random variable that is exponentially distributed with the parameter $detectrate$. It is employed to set the time needed for the detection of a failure. The function *Repair()* is evaluated to be a random number between a and b or a random variable that is exponentially distributed with the parameter $repairrate$, representing the time needed for the repair of a failure. The functions *FailSafeHazardous()*, *Fail()*, *Detect()* and *Repair()* can be defined as follows.

```
fun FailSafeHazardous() = discrete(0,100);
fun Fail() =
    round(exponential(!failurerate));
fun Fail() =
    round(exponential(!detectrate));
fun Repair() = discrete(a,b);
```

or

```
fun Repair() =
    round(exponential(!repairrate));
```

D. Model Analysis

1) Qualitative Analysis: To ensure its correctness, the CTCS-3 model need to be verified. The verification of the timed CPN model of the CTCS-3 system could be accomplished by conducting the following two steps: (1) Verifying the corresponding untimed CPN model by using state-space-based methods; (2) Checking the timing information by executing the timed CPN model. In this subsection, we focus on the first step. State spaces calculate all the reachable states (markings) and state changes (occurring binding elements) of CPN models, and represent

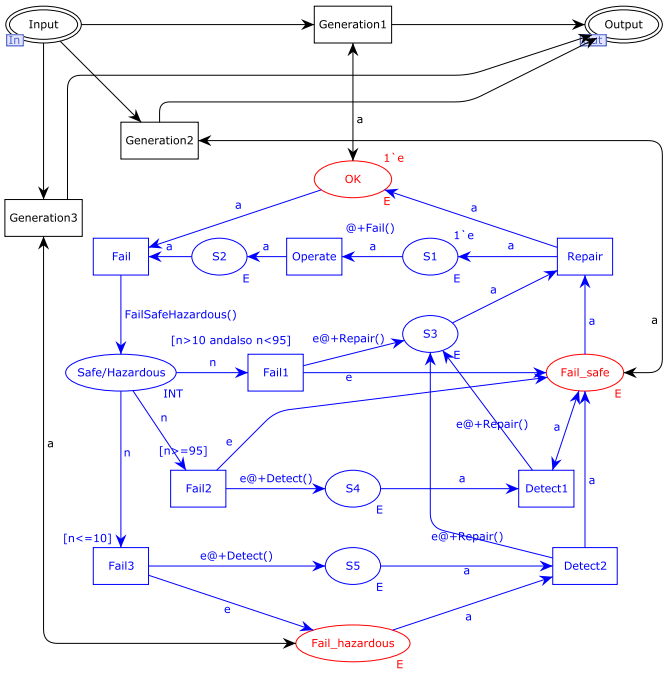

Fig. 5. Failure injection for a system component.

these in a directed graph where nodes correspond to the set of reachable markings and arcs correspond to occurring binding elements. A standard state space report generated by the CPN Tools provides some basic information and standard properties including statistics, boundedness properties, home properties, liveness properties and fairness properties of a CPN model. The statistical information about the state space shows the number of nodes and arcs of the state space and the strongly connected component graph (SCC graph), and the construction time of the state space and the SCC graph. The nodes in the SCC graph are strongly connected components (SCCs) that are obtained by making a disjoint division of the nodes in the state space such that two state space nodes are in the same SCC if and only if they are mutually reachable. The *boundedness properties* specify how many and which tokens a place may hold, when all reachable markings are considered. The *home properties* show the home markings. A *home marking* is a marking which can be reached from any reachable marking. The *liveness properties* provide the information on dead markings, dead transitions and live transitions. A *dead marking* is a marking under which no transition is enabled. A transition is *dead* if there are no reachable markings in which it is enabled. A transition is *live* if from any reachable marking we can always find an occurrence sequence containing the transition. The *fairness properties* provide information about how often transitions occur in infinite occurrence sequences. This part of the state space report lists those transitions that are impartial. A transition is *impartial* if it occurs infinitely often in all infinite occurrence sequences. Apart from the standard properties provided by a state space report, one may also want to investigate properties that are not general enough to be part of the state space report. For this purpose, a number of predefined query functions are available in

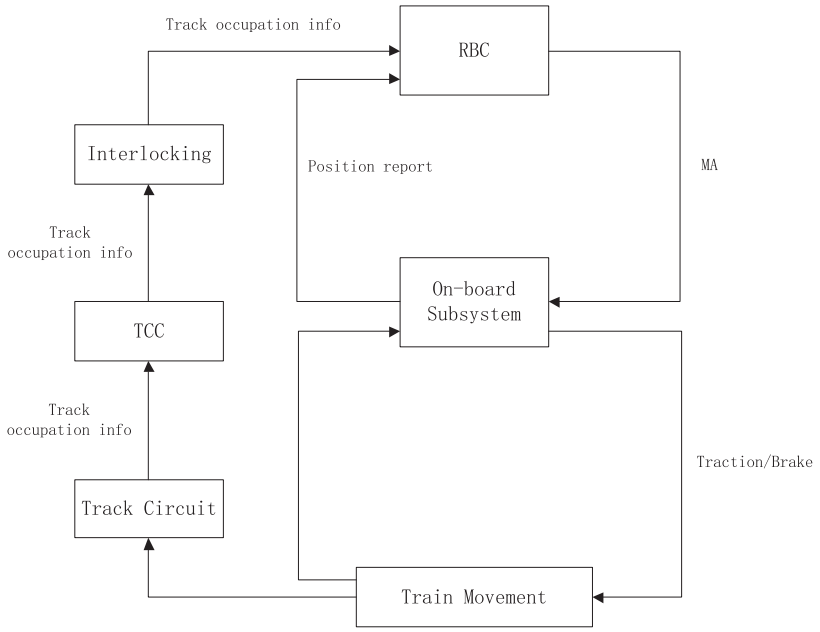

Fig. 6. The control structure of the simplified CTCS-3.

CPN Tools that make it possible to write user-defined and model-dependent queries [27]. These queries are written by using CPN ML programming language. With standard or non-standard queries, the following properties are usually verified [28]: *a) the absence of invalid dead markings; b) the absence of self-loop terminal markings; and c) the absence of livelocks.* A *livelock* of a system corresponds to an infinite execution that produces useless information. A Petri net model is in a livelock condition when it reaches a subset of its markings from which it has no possibility of exiting.

2) Quantitative Analysis: For the quantitative safety evaluation of a safety-related system, as we have presented in [18], the safety characteristics of *Mean Time To Hazardous Event (MTTHE)* and *Probability of keeping in normal and safe states* are usually evaluated based on the numerical data collected during the simulation of the system model. In this paper, to exhibit the feasibility of the proposed approach, we show the numbers of times that the CTCS-3 reaches a hazardous state (i.e, two trains in the same block section) according to different failure rates of the TCC and different show-up rates of the Train-xxx.

III. Modelling of the CTCS-3

A. Brief Introduction of the CTCS-3

The CTCS-3 (Chinese Train Control System - Level 3) was first applied to the Wuhan-Guangzhou High-speed Railway service allowing a maximum speed of 350 km/h in 2009 [29]. It has two subsystems: the ground subsystem and the on-board subsystem. The ground subsystem includes balises, track circuits, wireless communication networks (GSM-R), Radio Block Centers (RBC), Train Control Centers (TCC), and etc. The on-board subsystem consists of Vital Computers (VC), Balise Transmission Modules (BTM), Track Circuit Readers (TCR), Speed and Distance Units (SDU), Train Interface Units (TUI), Juridical Recorder Units (JRU), Driver Machine Interfaces (DMI), and etc. The RBC generates Movement Authorities (MAs) based on the information received from the track circuit and the interlocking, and sends the MAs, line data and temporary speed restriction orders to the on-board subsystem via the GSM-R. Meanwhile, the RBC receives position reports and train data from the on-board subsystem via the GSM-R. The TCC receives information from the track circuit and delivers it to the RBC. The on-board subsystem calculates static and dynamic speed curves monitoring the safe operation of the train according to the received MAs, line data, temporary speed restriction orders and other necessary data.

As stated in Section I, the CTCS-3 has 14 operation scenarios. To illustrate our approach, we select two operation scenarios, i.e., Registration and Startup and Movement Authority, for the modelling. These two scenarios are simplified to keep the CPN model concise. For the scenario of Registration and Startup (Fig. 1), when the on-board subsystem is in the Stand By mode, it sends the train ID to the RBC via the GSM-R in the first place; after receiving the train ID, the RBC sends a feedback of position report parameters and MA request parameters to the on-board subsystem; and then the on-board subsystem delivers a position report to the RBC; after that the RBC sends a feedback of configuration data to the on-board subsystem; then the on-board subsystem asks for a movement authority by sending an MA request to the RBC, and switches to the Full Supervision mode; after receiving the information of section occupation status from the interlocking, an MA will be sent to the on-board subsystem. For the scenario of Movement Authority (Fig. 2), when the on-board subsystem is in the Full Supervision mode, it sends position reports to the RBC and once the RBC receives the information of section occupation status from the interlocking, an MA will be sent to the on-board subsystem. Note that the information of section occupation status is generated by the track circuits and sent to the interlocking via the TCC. Based on the operation scenarios, the control structure of the CTCS-3 could be developed as shown in Fig. 6.

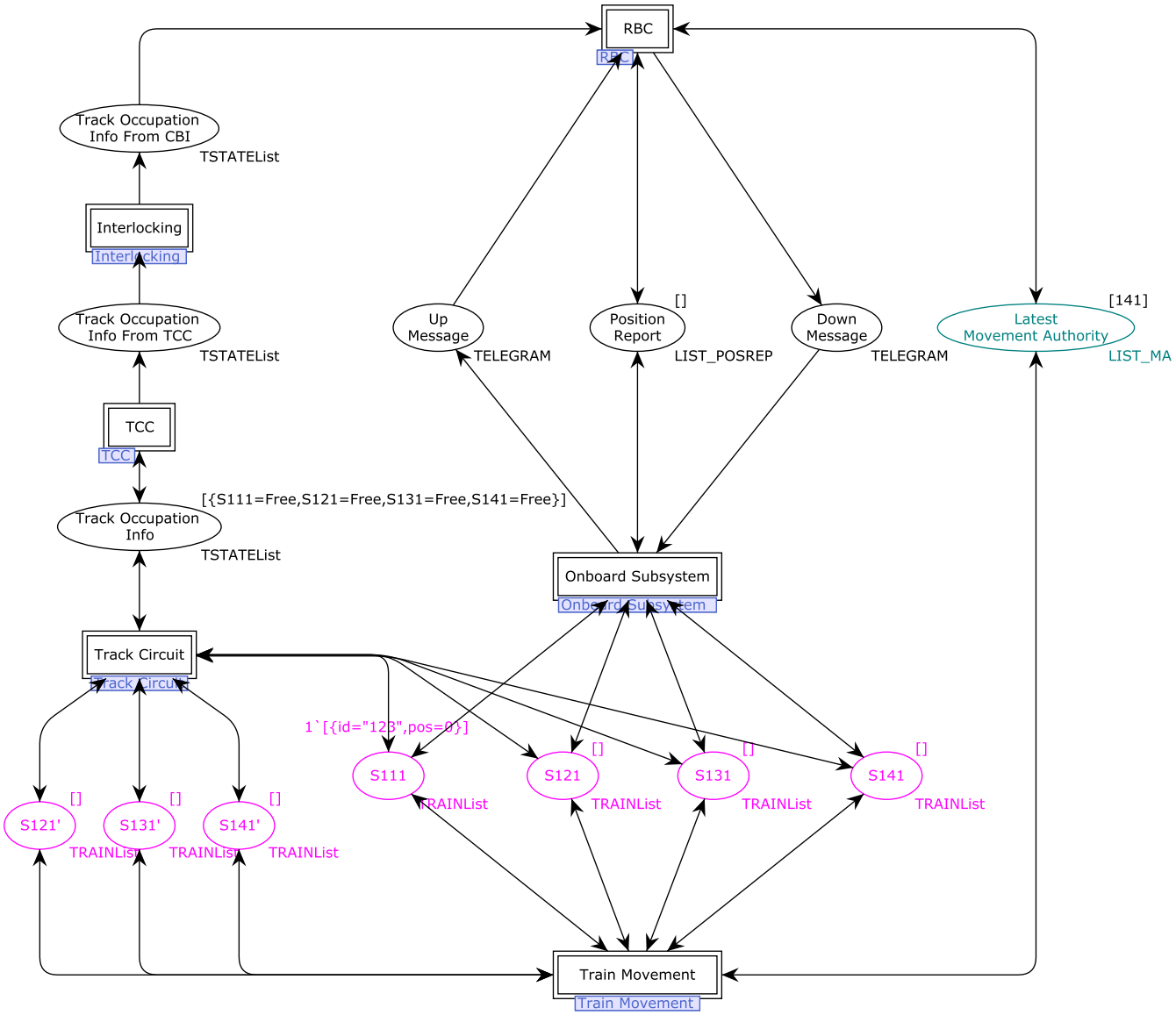

Fig. 7. Top level of the CPN model of the CTCS-3 system.

B. Top Level of the CTCS-3 Model

Given the control structure in Fig. 6, the top level of the CPN model of the CTCS-3 could be established as in Fig. 7. Valid token colors (commands/feedbacks) of the system model are defined by the color sets declared in the declarations as follows.

```
colset UNIT = unit;
colset BOOL = bool timed;
colset ID = string;
colset POS = int; (*train position*)
colset MA = int; (*movemment authority*)
colset LIST_MA = list MA timed;
colset TRAIN = record id:ID * pos:POS;
       (*train ID and position*)
colset TRAINList = list TRAIN timed;
colset POSREP = record id:ID * pos:POS;
       (*position report*)
colset LIST_POSREP = list POSREP timed;
colset STATE = with Free|Occupied;
       (*section state*)
colset TSTATE = record S111:STATE
                * S121:STATE * S131:STATE
                * S141:STATE;
       (*track state*)
colset TSTATEList = list TSTATE timed;
colset STATEList = list STATE timed;
colset E = with e timed;
colset TELEGRAM = with TrainID
                    |PosMaParInfo
                   |ConfInfo|MaReq;
       (*telegram*)
var tel: TELEGRAM;
var a: E;
var s: STRING;
var lTrain, lTrain', lTrain1, lTrain1',
    listT, listT1: TRAINList;
var id, id': ID;
```

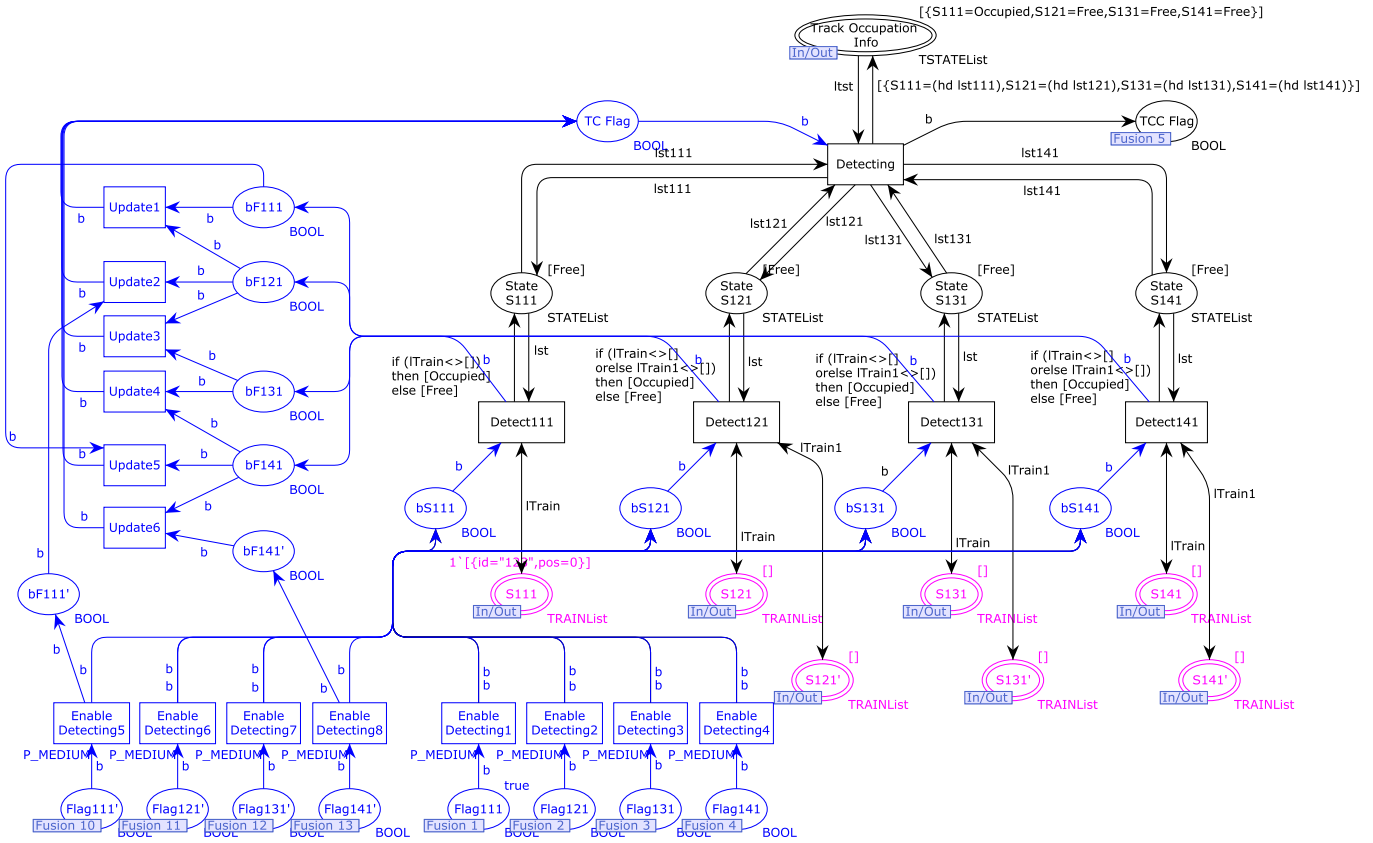

Fig. 8. Subpage of substitution transition `Track Circuit`.

```
var pos, pos': POS;
var lma, lma', lcem: LIST_MA;
var b: BOOL;
var lst, lst111, lst121, lst131,
    lst141: STATEList;
var tst: TSTATE;
var ltst, ltst': TSTATEList;
var lrep: LIST_POSREP;
```

C. CPN Models of the System Components

In this subsection, we present the CPN model of each component in the control structure (Fig. 6), namely, the subpage of each substitution transition in Fig. 7.

1) Module `Track Circuit`: The track circuits of the CTCS-3 operate as a sensor in terms of the control loop. One of the main functions of the track circuit is to detect whether a block section is occupied. Fig. 8 shows the subpage of the substitution transition `Track Circuit`. The token colors on the places `State S111`, `State S121`, `State S131` and `State S141` indicate the detected states of the block sections 111, 121, 131 and 141, respectively. When one of the transitions `Leave 111`, `Leave 121`, `Leave 131`, `Leave 141`, `Leave 111'`, `Leave 121'`, `Leave 131'` and `Leave 141'` in Fig. 15 fires (meaning that a train moves from one block section to another), two of the following transitions: `Detect111`, `Detect121`, `Detect131` and `Detect141`, will be fired and the token colors on their output places (showing that the occupation status of the corresponding two block sections have been changed) will be updated afterwards. This is realized by firing one of the transitions `Enable Detecting1`, `Enable Detecting2`, ⋯, `Enable Detecting8`. If two of the transitions `Detect111`, `Detect121`, `Detect131` and `Detect141` fire, the transition `Detecting` will be fired and the token color on the place `Track Occupation Info` that indicates the real-time occupation status of the four block sections will be updated. This is achieved by firing one of the transitions `Update1`, `Update2`, ⋯, `Update6`.

2) Module `Onboard Subsytem`: According to Fig. 1 and Fig. 2, the on-board subsystem exchanges messages with the RBC in two operation modes. Fig. 9 shows the subpage of the substitution transition `Onboard Subsystem`. For the operation modes Stand By mode and Full Supervision mode, the substitution transitions `Stand By` and `Full Supervision` are deployed. The place `Flag_MA` is a flag for activating the Full Supervision mode. The substitution transition `Localization` is a relatively independent functional block calculating the position of the train and generating position reports. It can be called by different modules/submodules.

a) Submodule `SB`: Fig. 10 shows the subpage of the substitution transition `Stand-By`. In the initial state, the place `Start` is marked by a boolean token color "true" and the transition `T1` is enabled. After firing the transition `T1`, the train ID is sent to the RBC via the output place `Up Message`. If a message of parameter information of MA and position report is received by the input place `Down Message`, the transition

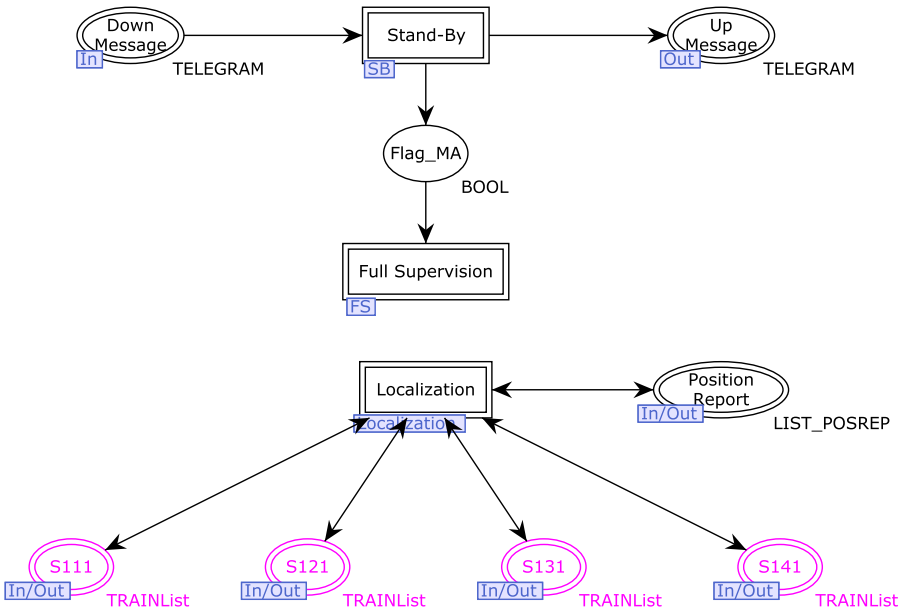

Fig. 9. Page of `Onboard Subsystem`.

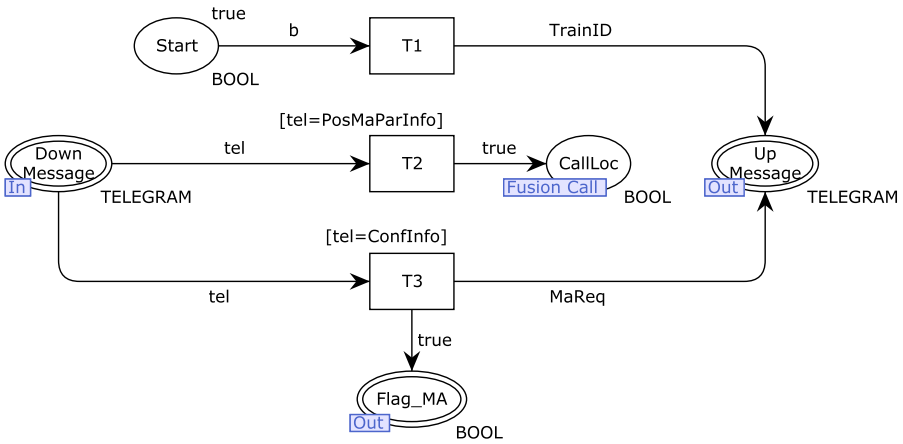

Fig. 10. Subpage of the substitution transition `Stand-By`.

`T2` will be fired and the function *localization* will be called to generate a position report. After receiving a message of configuration information from the input place `Down Message`, the transition `T3` is enabled. After firing the transition `T3`, an MA request will be sent to the RBC and a boolean token color "true" will be put onto the place `Flag_MA` to activate the Full Supervision mode.

b) Submodule `FS`: According to the operation scenarios described in Fig. 1 and Fig. 2, the main activities of the on-board subsystem in the Full Supervision mode are sending position reports and supervising the movement of the train within its movement authority. Fig. 11 shows the subpage of the substitution transition `Full Supervision`. When the on-board subsystem turns into this operation mode (i.e., the place `Flag_MA` is marked by a boolean token color "true"), it will call the function *localization* to generate position reports.

c) Submodule `Localization`: Fig. 12 shows the subpage of the substitution transition `Localization`. When the fusion place `CallLoc` is marked by a boolean token

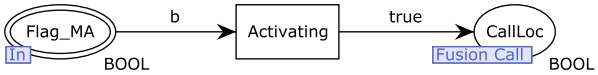

Fig. 11. Subpage of the substitution transition `Full Supervision`.

color "true", it means that the function of localization is being called. At this moment, if any one of the transitions, that is, `Leave 111`, `Leave 121`, `Leave 131`, `Leave 141`, `Leave 111'`, `Leave 121'`, `Leave 131'` and `Leave 141'` in Fig. 15 has been fired (meaning that the train has moved from one block section to another), one of the transitions `Positioning111`, `Positioning121`, `Positioning131` and `Positioning141` will be fired and a position report (represented by the token color of the place `Position Report`) will be generated.

3) Module `Interlocking`: Fig. 13 shows the subpage of the substitution transition `Interlocking`. It transfers the state information of track occupation to the RBC.

4) Module `RBC`: Fig. 14 shows the subpage of the substitution transition `RBC`. If a message of train ID is received

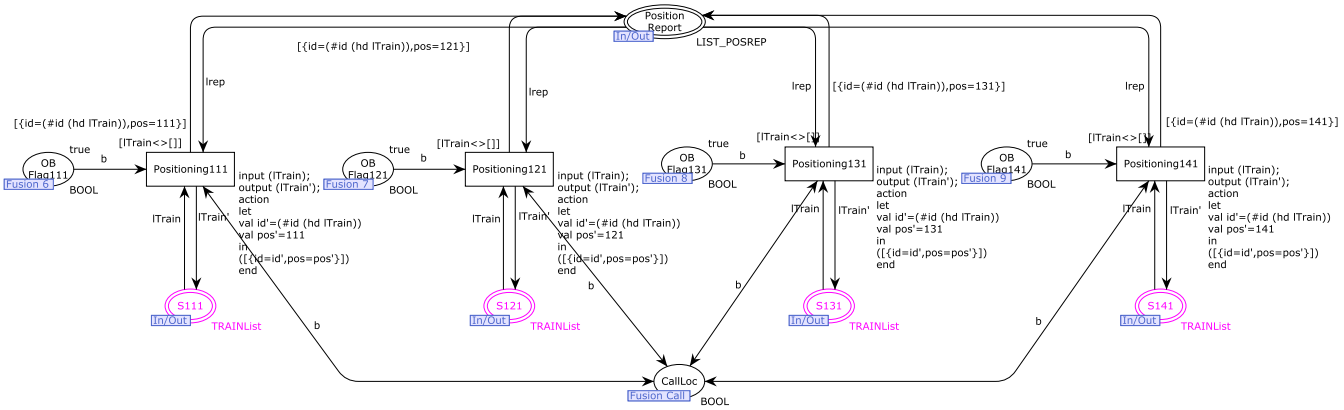

Fig. 12. Subpage of the substitution transition `Localization`.

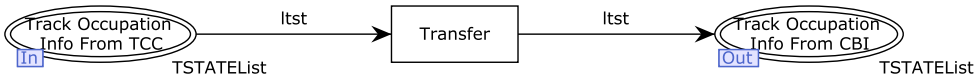

Fig. 13. Subpage of the substitution transition `Interlocking`.

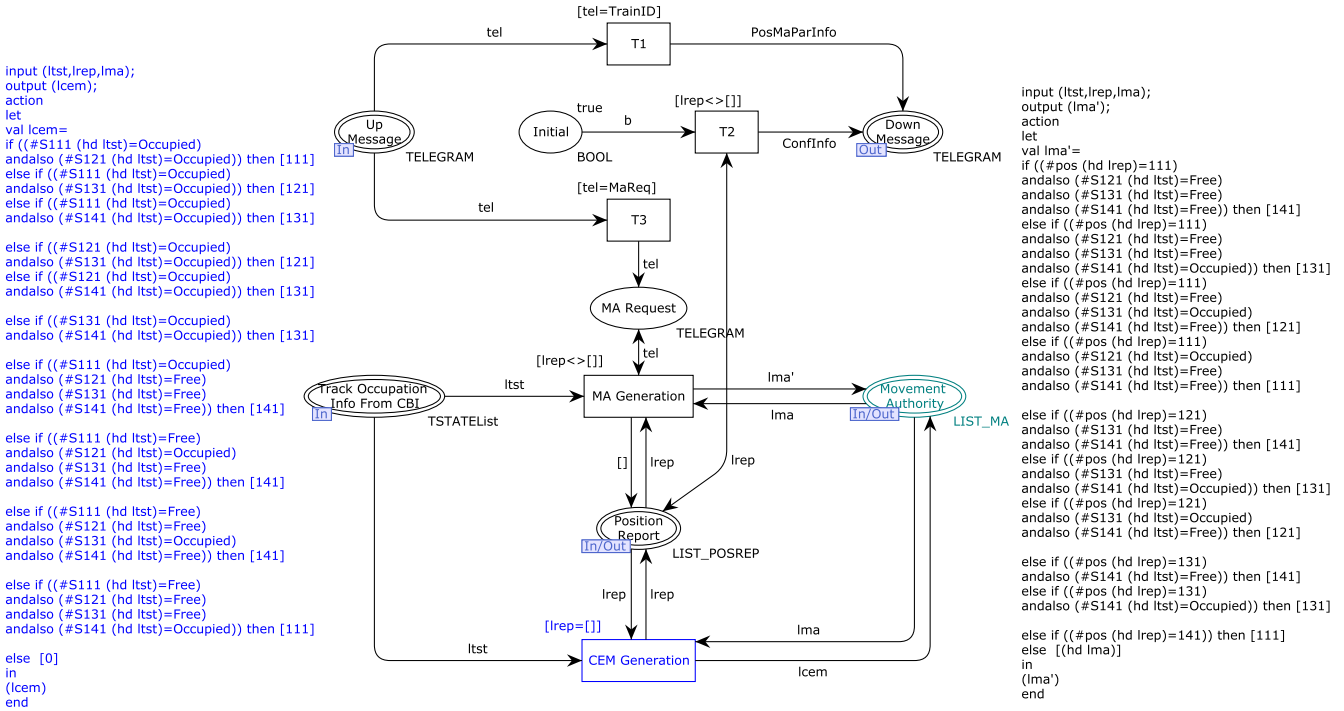

Fig. 14. Subpage of the substitution transition `RBC`.

by the input place `Up Message`, a message of parameter information of MA and position report will be sent to the on-board subsystem by firing the transition `T1`. Initially the place `Initial` is marked by a boolean token color "true". If a position report is received by the place `Position Report`, the transition `T2` will be fired and the configuration information will be sent to the on-board subsystem by the output place `Down Message`. If an MA request is received by the place `Up Message`, the transition `T3` will be fired and the place `MA Request` will be marked, which enable the MA generation. If a position report has been received and a message of the state information of track occupation is received by the input place `Track Occupation Info From CBI`, the transition `MA Generation` will be fired and an MA will be generated following the MA generation rules which are specified by the code segment of the transition. If no position report has been received and a message of the state information of track occupation is received, a message of CEM (Conditional Emergency brake Message) will be generated by firing the transition `CEM Generation`.

5) Module `Train Movement`: Fig. 15 shows the subpage of the substitution transition `Train Movement`. The following four combinations, the places `S111` and `In111`, `S121` and `In121`, `S131` and `In131`, `S141` and `In141`, represent four consecutive block sections 111, 121, 131 and 141 in a rail track, respectively. If any one of these places is marked by a non-empty list token color, it means this block section is occupied by the Train-123. Assuming that the place `S111` is

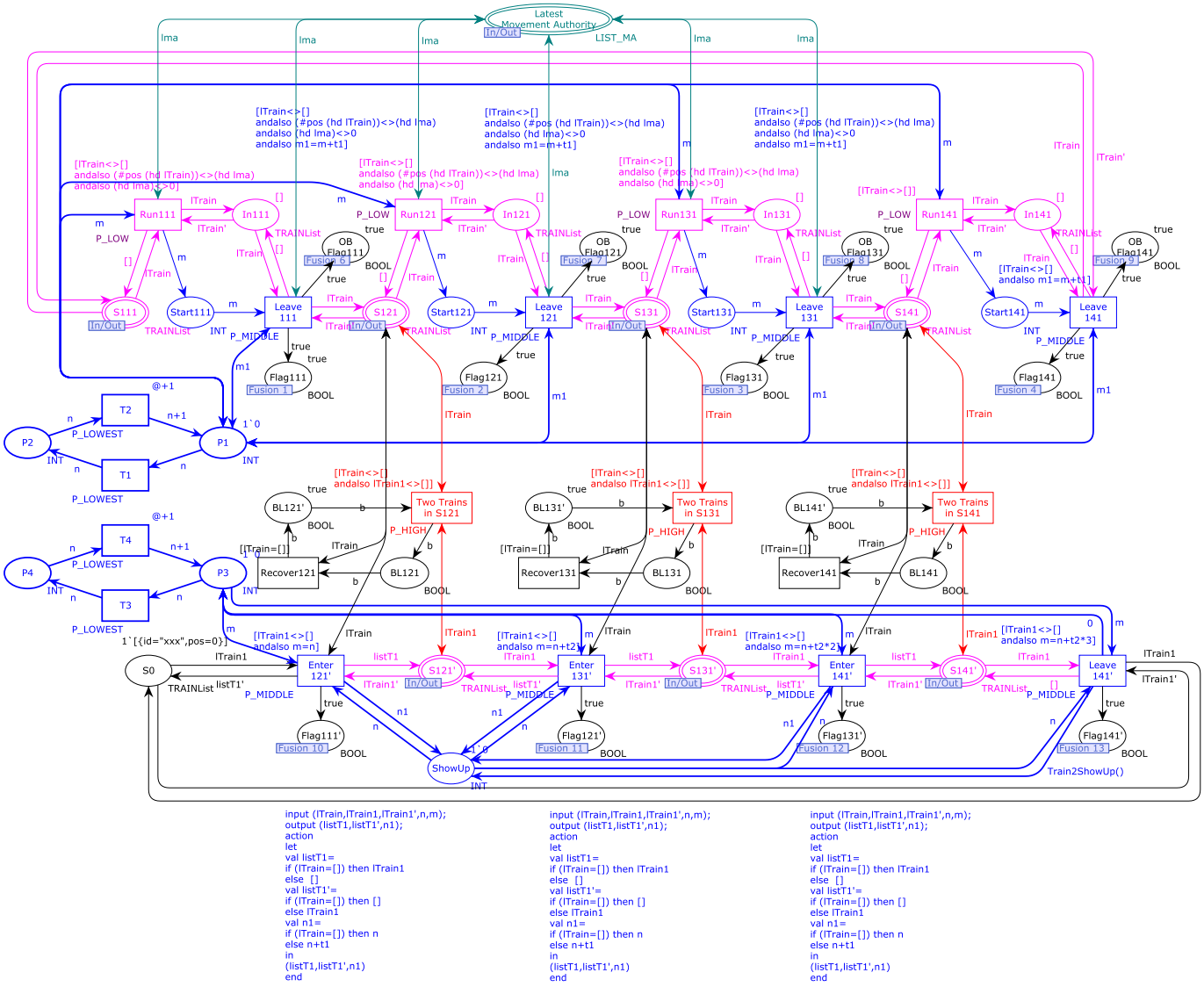

Fig. 15. Subpage of the substitution transition `Train Movement`.

marked by a token color "[{$id = 123$, $pos = 0$}]" (see Fig. 7) in the initial state (i.e., the block section 111 is occupied by the Train-123), if the place `Latest Movement Authority` is marked by a list token color "[121]", "[131]" or "[141]" (i.e., if the latest MA for the Train-123 is "to S121", "to S131" or "to S141"), the transition `Run 111` will be fired and the token color "[{$id = 123$, $pos = 111$}]" on the place `S111` will be removed and added to the place `In111`. After firing the transition `Leave 111`, the position of the train will be updated to be 121 by calling the `Localization` module, namely, the token color on the place `S121` will be updated to be "[{$id = 123$, $pos = 121$}]". So do the occurrences of the transitions `Run121`, `Leave 121`, `Run131` and `Leave 131`. When the place `S141` is marked by a non-empty list token color, the transitions `Run141` and `Leave 141` could be fired sequentially and the token color on the place `S141` will be removed and put back to the place `S111`. Note that the places `Start111`, `Start121`, `Start131` and `Start141` are deployed to store the timing information of the TIMER1 (see Subsection III-D). The places `S121'`, `S131'` and `S141'` represent the block sections 121, 131 and 141, respectively. When any one of these places is marked, it means that the Train-xxx is in this block section. Initially, the place `S0` is marked by a list token color "[{$id = xxx$, $pos = 0$}]", which implies that the Train-xxx is somewhere outside the set of the block sections 121, 131 and 141. If the place `S121` is marked by the empty list token color "[]" (meaning that the block section 121 is not occupied by the Train-123), the transition `Leave 111'` can be fired. After the firing of the transition, the place `S121'` will be marked by the token color "[{$id = xxx$, $pos = 0$}]" (meaning that the Train-xxx shows up in the block section 121). Note that "pos = 0" has no meaning here. So will the occurrences of the transitions `Leave 121'` and `Leave 131'`. When the place `S141'` is marked by the token color "[{$id = xxx$, $pos = 0$}]", the transition `Leave 141'` could be fired and the place `S0` will be marked by the token color "[{$id = xxx$, $pos = 0$}]" again.

Since the movement of the Train-123 depends on the latest movement authority generated by the RBC, the following state could be considered as a hazardous situation:

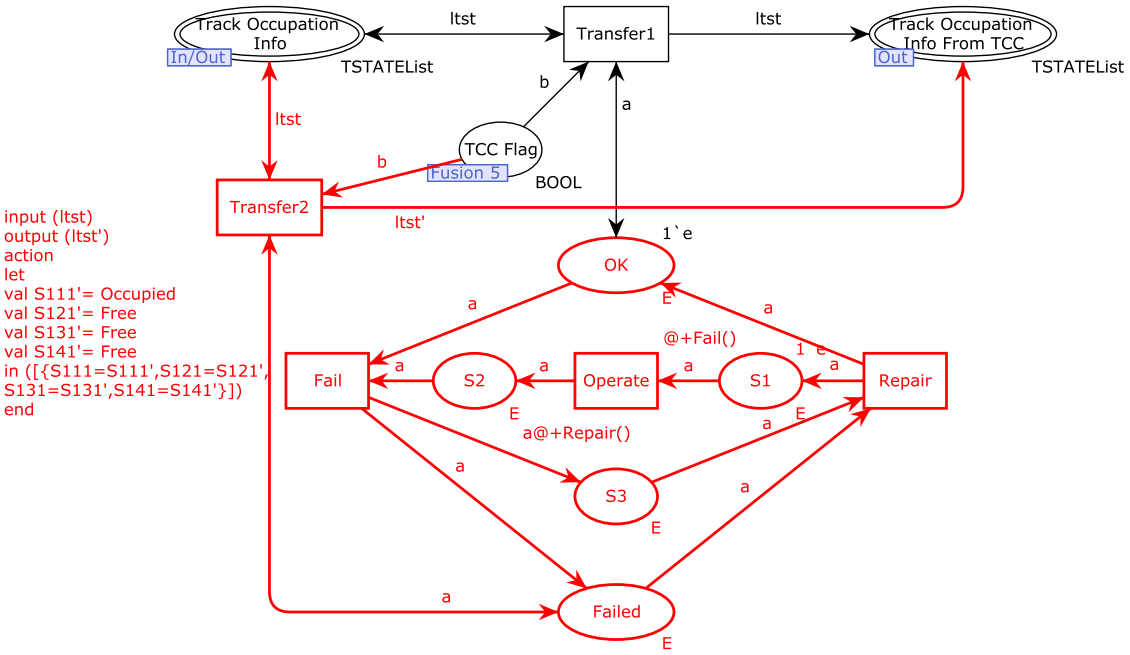

Fig. 16. Subpage of the substitution transition TCC.

the places S111/In111 and S121′ are marked by the token colors "[{$id = 123, pos = 111$}]" and "[{$id = xxx, pos = 0$}]" respectively, and the place Latest Movement Authority is marked by a token color "[121]", "[131]" or "[141]". This state suggests that the Train-123 will run into the block section 121 even though the block section 121 is occupied by the Train-xxx at the moment. Once this state is reached, the transition Two train in S121 will be fired and a boolean token color "true" will be added to the place BL121. After the Train-xxx leaves the block section 121, i.e., the token color "[{$id = xxx, pos = 0$}]" on the place S121′ is removed by firing the transition Leave 121′, the transition Recover will be fired, which suggests a recovery from the hazards situation. So do the occurrences of the transition Two trains in S131 and Two trains in S141. Note that the transitions Two trains in S121, Two trains in S131 and Two trains in S141 have higher firing priorities compared with other transitions in this module.

In Fig. 15, there are two timers. One (say TIMER1) consisting of the places P1, P2 and the transitions T1, T2 is deployed to regulate the movement of the Train-123. The other (say TIMER2) comprising the places P3, P4 and the transitions T3, T4 is developed to regulate the movement of the Train-xxx. Without such timers, the timing information needs to be directly added to the token colors on the places that represent the block sections such as the place S121. In this case, it is difficult to implement the detection of hazardous situations that two trains are in the same block section. For example, following the firing mechanism of timed CPNs, the transition Two Trains in S121 will be fired only when the places S121 and S121′ are marked by non-empty list token colors and the timestamps of both token colors are exactly the same as well. This fails to model the realistic hazardous situation that two trains enter the same block section successively, which could lead to a rear-end collision.

TIMER1: as described in Sub-subsection II-B.4.

TIMER2: the same as TIMER1 except that the value of the token color on the place P3 may not be equal to the timestamp of the token color, since the occurrence of the transition Leave 141′ will restore the value of the token color to 0. Note that since a CPN model only has one model time and only one transition could be fired at any time instance, the values of the timestamps of the token colors on the places P1 and P3 are either equal or have a difference of one time unit. Therefore, the maximal deviation of the model in time is one second (or one time unit).

6) Module TCC: Fig. 16 shows the subpage of the substitution transition TCC. If the state information of track occupation is received by the place Track Occupation Info, it will be transferred to the interlocking by firing the transition Transfer1.

D. Timing Configuration and Failure Injection

According to the assumptions given in Sub-subsection II-B.4 that the time durations of the Train-123 and the Train-xxx running from one block section to another is 26 seconds and 40 seconds, respectively, the parameters t1 and t2 of the model are defined by the declarations val t1 = 26 and val t2 = 40.

According to Sub-subsection II-B.4, we assume that the mean time interval between two consecutive trains showing up on the rail track is $E(\tau) = \frac{1}{\lambda} = 10min = 600s$, then $\lambda = 0.00167$. This is implemented by adding the function $Train2ShowUp()$ to the directed arc from the transition Leave 141′ to the place ShowUp (see Fig 15). The definition of the function $Train2ShowUp()$ is as in the following:

```
globref showuprate=0.00167 (*10 minutes*);
fun Train2ShowUp() =
      round(exponential(!showuprate));
```

In this work, to exemplify the failure injection method and keep the model as concise as possible at the same time, only the failure and repair of the component TCC is discussed (see Fig. 16). As the fail-safe state of the component does not incur hazard/accident, it is therefore also not considered in our discussion. Otherwise, the model presented in Fig. 5 can be integrated into the CPN model of the TCC (Fig. 16). It is noteworthy that the fail-safe state of a system component should be considered when we evaluate the availability or dependability of the system. Assuming that the failure time of the TCC is 30 minutes and the repair time of the TCC is 10 to 15 minutes, the functions *Fail()* and *Repair()* could be defined as in the following:

```
globref failurerate=0.000556
                    (*30 minutes*);
fun Fail() =
     round(exponential(!failurerate));
fun Repair() = discrete(600,900)
     (*10 minutes ot~15~minutes*);
```

Note that the values for the failure time and repair time of the TCC are chosen for the sake of simulation convenience.

IV. Analysis of the CTCS-3 Model

A. Qualitative Analysis

For the purpose of accomplishing state-space-based verification, the CPN model of the CTCS-3 could be modified as in the following: (1) The timing information is removed, i.e., the "TIMER1" and "TIMER2" (including the place `ShowUp`) in Fig. 15, the places `S1, S2, S3` and the transition `Operate` in Fig. 16 are removed; (2) The unnecessary loops are eliminated in order to reduce the state space and provide sufficient state information, i.e., the arcs between the transition `Leave 141` and the place `S111`, as well as the arcs between the transition `Leave 141'` and the place `S0` in Fig. 15 are eliminated. This implies that the train movement process will terminate after two trains have finished their journeys starting at the block section `S121` and leaving from the block section `S141`. Rear-end collision between these two trains might happen if something went wrong in the control system (i.e., the TCC failed in our case). The transitions `Fail` and `Repair` in Fig. 16 are deleted, and a token color "e" is put onto the place `Failed`. It means that both states (i.e., "OK" and "Failed") are possible for the TCC. Additionally, the transitions `Recover121`, `Recover131` and `Recover141` in Fig. 15 are further eliminated; (3) The firing priorities of the transitions `Leave111`, `Leave121`, `Leave131`, `Leave141`, `Enter121'`, `Enter131'` and `Enter141'` are set to be `P_LOW` (i.e., the lowest firing priority in the modified CPN model). This is valid because the time duration of a train in a block section is much longer than the time that other information processing processes of the CTCS-3 needed.

TABLE I
PART OF INFORMATION OF THE STANDARD STATE SPACE REPORT

Statistics
State Space
Nodes: 2505
Arcs: 2931
Secs: 4
Status: Full
SCC Graph
Nodes: 2505
Arcs: 2931
Secs: 0
Home Properties
Home Markings
None
Live Properties
Dead Markings
22[960,934,2505,2504,2497,...]
Dead Transition Instances
None
Live Transition Instances
None
Fairness Properties
No infinite occurrence sequences.

Table I shows part of information of the standard state space report of the modified CPN model of the CTCS-3. The state space is complete and there are 2505 markings and 2931 arcs. The construction of the state space took 4s. The SCC graph has the same number of markings and arcs as the state space, which means there are no cycles in the state space. The home properties show that there are no home markings. The liveness properties express that there are 22 dead markings, no dead transition instances and no live transition instances. The fairness properties tell us that there are no infinite occurrence sequences. As expected, the home properties, the number of live/dead transition instances and the fairness properties are desirable. As for the dead markings, further analysis is required to prove that the dead markings are correct terminations of the system model. This is verified by applying non-standard queries to the state space as in the following paragraph.

Table II shows the query for listing all dead markings and invalid dead markings of the system model. The query result shows that there are 22 dead markings, which is exactly the same number as reported in the standard state space report as expected and none of these dead markings is invalid. Function `ValidTerminal` specifies the valid terminations that the Train-123 and the Train-xxx have been moved out from the block section `S141` and `S141'`, respectively, i.e., all block sections are free. Function `InvalidTerminal` checks for the invalid terminations (invalid dead markings). Table III and Table IV show the queries to investage self-loops and livelocks of the system model. The query results indicate that there are no self-loops and livelocks.

B. Quantitative Analysis

To enable the model-based simulation, necessary loops have been introduced into the CPN model of the CTCS-3. The model will keep running unless intervention is made. Thus, it is necessary to set a stop criterion to terminate a model simulation process. For this purpose, we have set up

TABLE II
QUERY OF DEAD MARKINGS AND INVALID DEAD MARKINGS

Query formula	Query result

`fun ValidTerminal n=(Mark.HighSpeedRailway'S111 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S121 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S131 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S141 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S121' 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S131' 1 n=[[]]`
`andalso Mark.HighSpeedRailway'S141' 1 n=[[]]`
`andalso Mark.TrainMovement'S0 1 n=[[]])`
`fun InvalidTerminal()=PredNodes(ListDeadMarkings(),`
`fn n=>not(ValidTerminal n),NoLimit);`
`let`
`val filename=TextIO.openOut "DeadMarkingAnalysis.txt"`
`val _ =TextIO.output(filename, "List of dead markings:\n")`
`val _ =EvalNodes(ListDeadMarkings(), fn n=>INT.output(filename, n))`
`val _ =TextIO.output(filename, "\nNumber of dead markings:")`
`val _ =INT.output(filename, length(ListDeadMarkings()))`
`val _ =TextIO.output(filename,"\nList of invalid dead markings:\n")`
`val _ =EvalNodes(InvalidTerminal(),fn n=>INT.output(filename,n))`
`val _ =TextIO.output(filename, "\nNumber of invalid dead markings:")`
`val _ =INT.output(filename, length(InvalidTerminal()))`
`in`
`TextIO.closeOut(filename)`
`end;`
	`List of dead markings:`
	`960 934 2505 2504 2497`
	`2496 2495 2494 2493`
	`2492 2473 2472 2465`
	`2464 2463 2462 2422`
	`2421 1630 1590 1569`
	`1557`
	
	`Number of dead mark`
	`-ings:`
	`22`
	
	`List of invalid dead`
	`markings:`
	
	`Number of invalid dead`
	`markings:`
	`0`

TABLE III
QUERY OF SELF-LOOP TERMINAL MARKINGS

Query formula	Query result

`fun SelfLoopTerminal n=(OutNodes(n)=[n])`
`fun InvalidTerminal()=PredNodes(EntireGraph,`
`fn n=>(SelfLoopTerminal n),NoLimit);`
`let`
`val filename=TextIO.openOut"SelfLoops.txt"`
`val _ =TextIO.output(filename,"List of self-loop terminals:\n")`
`val _ =EvalNodes(InvalidTerminal(),fn n=>INT.output(filename,n))`
`val _ =TextIO.output(filename, "\nNumber of self-loop terminals:")`
`val _ =INT.output(filename, length(InvalidTerminal()))`
`in`
`TextIO.closeOut(filename)`
`end;`
	`List of self-loop`
	`terminal markings:`
	
	`Number of self-loop`
	`terminal markings:`
	`0`

TABLE IV
QUERY OF LIVELOCKS

Query formula	Query result

`fun ListTerminalSCCs()=PredAllSccs(SccTerminal);`
`fun InvalidTerminalSCC()=PredSccs(ListTerminalSCCs(),`
`fn n=>not(SccTrivial n),NoLimit);`
`let`
`val filename=TextIO.openOut"Livelocks.txt"`
`val _ =if InvalidTerminalSCC()=[]`
`then TextIO.output(filename,"No Livelocks!")`
`else TextIO.output(filename,"There exist livelocks!")`
`in`
`TextIO.closeOut(filename)`
`end;`
	`No Livelocks!`

a *break point* monitor which will terminate the simulation when the model time exceeds 1,000,000 time units (representing 277.78 hours in the real world). To monitor how many times the CTCS-3 have reached the hazardous states (i.e, two trains in the block sections `S121`, `S131` or `S141`), three *count transition occurrence data collector* monitors

TABLE V
STATISTICS FROM SIMULATION PERFORMANCE REPORT ACCORDING TO DIFFERENT FAILURE RATES OF TCC

TCC failure rate (MTTF)	Average of occurrences of `Two Trains in S121` (n_1)	Average of occurrences of `Two Trains in S131` (n_2)	Average of occurrences of `Two Trains in S141` (n_3)	Average of occurrences of hazardous events ($n_1 + n_2 + n_3$)
0.000333 (50min)	0.600000	0.400000	0.600000	1.600000
0.000417 (40min)	0.800000	0.800000	1.200000	2.800000
0.000556 (30min)	1.200000	1.400000	1.400000	4.000000
0.000833 (20min)	1.400000	1.600000	1.200000	4.200000
0.00111 (15min)	1.000000	1.800000	1.600000	4.400000
0.00167 (10min)	1.800000	2.400000	2.200000	6.400000
0.00333 (5min)	2.600000	2.800000	3.400000	8.800000
0.00556 (3min)	5.200000	5.200000	5.200000	15.600000

[1] Note that the show-up rate of the Train-xxx is set to be 0.00167, i.e., the mean time to show up for the Train-xxx is 10 minutes.

TABLE VI
STATISTICS FROM SIMULATION PERFORMANCE REPORT ACCORDING TO DIFFERENT SHOW-UP RATES OF TRAIN-XXX

Show-up rate of Train-xxx (mean time to show up)	Average of occurrences of `Two Trains in S121` (n_1)	Average of occurrences of `Two Trains in S131` (n_2)	Average of occurrences of `Two Trains in S141` (n_3)	Average of occurrences of hazardous events ($n_1 + n_2 + n_3$)
0.000333 (50min)	1.000000	0.800000	1.000000	2.800000
0.000417 (40min)	0.600000	1.000000	0.800000	2.400000
0.000556 (30min)	1.400000	1.200000	0.800000	3.400000
0.000833 (20min)	0.800000	0.600000	0.600000	2.000000
0.00111 (15min)	0.400000	0.400000	0.600000	1.400000
0.00167 (10min)	1.800000	1.400000	2.000000	5.200000
0.00333 (5min)	1.000000	1.000000	0.800000	2.800000
0.00556 (3min)	0.600000	0.800000	1.200000	2.600000

[1] Note that the failure rate of the TCC is set to be 0.000556, i.e., the mean time to failure of the TCC is 30 minutes.

on the transitions `Two Trains in S121`, `Two Trains in S121` and `Two Trains in S121`, respectively, in the module of `Train Movement` (Fig. 15) have been specified. Since the model contains random variables (which means that a simulation of the model contains random behaviour), the data collected during the simulation also exhibit random behaviours. Therefore, a number of different simulations should have been done when interpreting and analysing the collected data. In our case study, 5 simulations are taken into account. The values for the break point monitor and the simulation times are set up by considering the time needed for simulation and the acceptable level of accuracy based on our practical experiences. The averages of the observed data are used to illustrate the simulation results.

Table V shows the averages of occurrences of hazardous events according to different failure rates of the TCC for 5 simulations. Note that we assume that the repair time for the TCC is a random discrete number between $10min$ and $15min$, the speed of the Train-123 is $300km/h$ and the speed of the Train-xxx is $200km/h$. Table VI shows the averages of occurrences of hazardous events according to different show-up rates of the Train-xxx for 5 simulations. As can be seen, hazardous events increase in number as the failure rate of the TCC goes up, and the parameter of show-up rate of the Train-xxx does not have an obvious impact on the average number of hazardous events.

V. CONCLUSION

In this paper, we present a stochastic CPN-based approach for qualitative and quantitative safety evaluation of the CTCS-3. Our approach relies on two fundamental constitutes: realistic system models and stochastic simulations. The realistic system model of the CTCS-3 is developed based on the concepts of operation scenario of the system, operation mode of the on-board subsystem, and the control structure of the system. In particular, the movement of trains is included in the system model. To enable safety analysis, faults and failures could be inserted into the system model components. The stochastic simulation is carried out by adding timing information to the system model exhibiting the deterministic and stochastic behaviours of the CTCS-3. The correctness of the system model is verified by standard state space report analysis as well as standard and non-standard queries to the state space. The number of times that the CTCS-3 reaches the specified hazardous states according to different failure rates of the TCC and different show-up rates of the Train-xxx are provided to demonstrate the feasibility of the proposed approach at the end of the work. Compared with FTA and ETA, the proposed approach develops more realistic system models and the correctness of the models could be verified by various methods. Additionally, stochastic system behaviours can be well handled by the proposed approach. For future work, the failures of other components (besides the TCC) could be added to the system model, other safety characteristics (such as MTTHE) or availability of the CTCS-3 could be evaluated as well.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the anonymous reviewers for their valuable comments.

REFERENCES

[1] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, "Fault tree handbook," Nuclear Regulatory Commission Washington DC, Washington DC, USA, Tech. Rep. NUREG-0492, 1981.

[2] C. A. Ericson, *Hazard Analysis Techniques for System Safety*. Hoboken, NJ, USA: Wiley, 2005.
[3] *EN 50128:2001. Railway Applications—Communications, Signalling and Processing Systems—Software for Railway Control and Protection Systems*, CENELEC, Brussels, Belgium, 2001.
[4] T. P. K. Nguyen, J. Beugin, and J. Marais, "Method for evaluating an extended fault tree to analyse the dependability of complex systems: Application to a satellite-based railway system," *Rel. Eng. Syst. Saf.*, vol. 133, pp. 300–313, Jan. 2015.
[5] P. Liu, L. Yang, Z. Gao, S. Li, and Y. Gao, "Fault tree analysis combined with quantitative analysis for high-speed railway accidents," *Saf. Sci.*, vol. 79, pp. 344–357, Nov. 2015.
[6] B. S. Medikonda, P. S. Ramaiah, and A. A. Gokhale, "FMEA and fault tree based software safety analysis of a railroad crossing critical system," *Global J. Comput. Sci. Technol.*, vol. 11, no. 8, May 2011.
[7] F. De Felice and A. Petrillo, "Methodological approach for performing human reliability and error analysis in railway transportation system," *Int. J. Eng. Technol.*, vol. 3, no. 5, pp. 341–353, 2011.
[8] J. Xiang, K. Futatsugi, and Y. He, "Fault tree and formal methods in system safety analysis," in *Proc. 4th Int. Conf. Comput. Inf. Technol.*, Sep. 2004, pp. 1108–1115.
[9] K. M. Hansen, A. P. Ravn, and V. Stavridou, "From safety analysis to software requirements," *IEEE Trans. Softw. Eng.*, vol. 24, no. 7, pp. 573–584, Jul. 1998.
[10] G. Schellhorn *et al.*, "Formal fault tree semantics," in *Proc. 6th World Conf. Integr. Design Process Technol.*, Pasadena, CA, USA, 2002, pp. 1–8.
[11] N. Leveson, "A new accident model for engineering safer systems," *Saf. Sci.*, vol. 42, no. 4, pp. 237–270, Apr. 2004.
[12] C. H. Fleming and N. G. Leveson, "Early concept development and safety analysis of future transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3512–3523, Dec. 2016.
[13] A. Bobbio, G. Franceschinis, R. Gaeta, and L. Portinale, "Exploiting Petri nets to support fault tree based dependability analysis," in *Proc. 8th Int. Workshop Petri Nets Perform. Models*, Sep. 1999, pp. 146–155.
[14] R. Slovák, "Methodische modellierung und analyse von sicherungssystemen des eisenbahnverkehrs," Ph.D. dissertation, Dept. Mech. Eng., Technische Universität Braunschweig, Jul. 2006. [Online]. Available: https://publikationsserver.tu-braunschweig.de/receive/dbbs_mods_00018016
[15] R. Slovák, J. May, and E. Schnieder, "Profund modelling for holistic risk and availability analysis by means of stochastic Petri nets applied to a level crossing control system," in *Proc. Symp. Formal Methods Railway Oper. Control Syst. (FORMS)*, 2003, pp. 221–232.
[16] E. Schnieder and L. Schnieder, *Verkehrssicherheit—Maße und Modelle, Methoden und Maßnahmen für den Straßen-und Schienenverkehr*. Berlin, Germany: Springer-Verlag, 2013.
[17] P. Pozsgai and B. Bertsche, "Conjoint modelling with extended coloured stochastic Petri net and reliability block diagram for system analysis," in *Probabilistic Safety Assessment and Management*. London, U.K.: Springer, 2004, pp. 1382–1387.
[18] D. Wu and W. Zheng, "Formal model-based quantitative safety analysis using timed coloured Petri nets," *Rel. Eng. Syst. Saf.*, vol. 176, pp. 62–79, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832017312735
[19] L. Jansen, M. M. Z. Hörste, and E. Schnieder, "Technical issues in modelling the European train control system (ETCS) using coloured Petri nets and the Design/CPN tools," in *Proc. Workshop Practical Coloured Petri Nets and Design (CPN)*, Aarhus, Denmark, 1998, pp. 103–115.
[20] M. M. Z. Hörste and E. Schnieder, "Modelling functionality of train control systems using Petri nets," in *Proc. FM-RAIL-BOK Workshop*, Madrid, Spain, 2013.
[21] D. Wu and E. Schnieder, "Scenario-based modeling of the on-board of a satellite-based train control system with colored Petri nets," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3045–3061, Nov. 2016.
[22] S. Uchitel, J. Kramer, and J. Magee, "Synthesis of behavioral models from scenarios," *IEEE Trans. Softw. Eng.*, vol. 29, no. 2, pp. 99–115, Feb. 2003.
[23] Object Management Group. (Feb. 2009). *OMG Unified Modeling LanguageTM (OMG UML), Superstructure: Version 2.2*. [Online]. Available: http://www.omg.org/spec/UML/2.2/Superstructure
[24] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Berlin, Germany: Springer-Verlag, 2009.
[25] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Berlin, Germany: Springer-Verlag, 2001.
[26] Aarhus University. (2000). *CPN Tools Homepage*. [Online]. Available: http://cpntools.org/
[27] K. Jensen, S. Christensen, and L. M. Kristensen, "CPN tools state space manual," Dept. Comput. Sci., Univerisity Aarhus, Aarhus, Denmark, Tech. Rep., 2006.
[28] P. Katsaros, "A roadmap to electronic payment transaction guarantees and a colored Petri net model checking approach," *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 235–257, Feb. 2009.
[29] H. Dong, B. Ning, B. Cai, and Z. Hou, "Automatic train control system development and simulation for high-speed railways," *IEEE Circuits Syst. Mag.*, vol. 10, no. 2, pp. 6–18, May 2010.

Daohua Wu received the B.Sc. and M.Sc. degrees from Beijing Jiaotong University, Beijing, China, in 2007 and 2010, respectively, and the Dr.-Ing. degree from the Faculty of Mechanical Engineering, Technische Universität Braunschweig, Germany, in 2014. Then, he became a Post-Doctoral Fellow with the Institute of Traffic Safety and Automation Engineering, Braunschweig University of Technology. Since July 2015, he has been with the National Research Center of Railway Safety Assessment, Beijing Jiaotong University. His research interests include safety assessment, modelling/system design, particularly for train control systems with formal languages, system verification, and validation.

Debiao Lu received the B.Sc. degree in telecommunication engineering and the M.Sc. degree in traffic information engineering and control from Beijing Jiaotong University, China, and the Ph.D. degree from the Faculty of Mechanical Engineering, Technische Universität Braunschweig, Germany, in 2014. Since 2015, he has been working with the Faculty of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include GNSS safe localization and GNSS propagation simulation and performance assessment.

Tao Tang (Senior Member, IEEE) received the Ph.D. degree in engineering from the Chinese Academy of Sciences, Beijing, China, in 1991. He is currently the Director of the School of Electronic and Information Engineering and the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing. His research interests include communication-based train control, high-speed train control systems, and intelligent transportation systems. He is also a member of the Experts Group of High Technology Research and Development Program of China (863 Program) and the Leader of the Field of Modern Transportation Technology Experts Group. He is also a Specialist with the National Development and Reform Commission and the Beijing Urban Traffic Construction Committee.