An Application of Discrete Event Systems based Fault Diagnosis to German Railway Signaling System

Mustafa S. Durmuş, İlker Üstoğlu

***Abstract*—**Fixed-block railway signaling systems are both regarded as Discrete Event System (DES) due to having DES-like features in their structure and safety-critical due to the fact that the occurrence of a fault may result with a huge loss of life and property. Therefore, DES-based modeling and fault diagnosis methods which are also recommended in the railway related safety standards are applicable to fixed-block signaling systems. On the other hand, the design steps of software for safety-critical systems are guided by software lifecycles. Among different lifecycle models proposed in the literature, the V-model is one of the most well-known software development lifecycle model. In this study, a modification for the V-model is proposed by adding a DES-based fault diagnosis step to the V-model. The proposed method is also explained with a case study on the German Ks signaling system.

***Keywords*—**Fixed-block railway signaling systems, discrete event systems, fault diagnosis, German Ks system, software development lifecycle, the V-model.

I. Introduction

LIKE in all other safety-critical system applications, software development process for railway signaling systems are also guided by software lifecycle models. For railway signaling systems, in addition to the recommendations of the EN 50126, EN 50128 and EN 50129 standards, recommendations of the IEC 61508 standard should be taken into consideration. These standards recommends to use the V-model for software development processes (the V-model lifecycle). Definition of the scope of the software, hazard and risk analysis, definition of the software requirements, determination of the safety requirements, software design, software integration, software tests, installation, commissioning, validation, operation, maintenance, repair and decommissioning are dealt with in this lifecycle Moreover, the IEC 61508-7 [1] describes fault diagnosis as the process of determining if a system is in a faulty state or not whereas, DES-based fault diagnosis and the diagnosability is described by Sampath et al. [2] as the detection with a finite delay occurrence of failures of any type using the record of observable events. The diagnoser is obtained by using the system model itself and it observes online the behavior of the system [3]. In particular, the EN 50128 standard, Table A.3 (Software Architecture) highly recommends to use fault detection and diagnosis for SIL3 (Safety Integrity Level) systems [4].

In this study, the DES-based fault diagnosis method is added as an intermediate step into the V-model software development lifecycle. The proposed modification provides advantages in three ways: 1. checks if the constructed software model covers all software requirements related with the faults, 2. decrease the costs by early detecting the modeling deficiencies before passing to the coding and test phases in the V-model, 3. enables designers more plain and simple coding.
The paper is organized as follows: a brief description of the DES-based fault diagnosis concept is given with an example in section II, the V-model lifecycle and the proposed modification is given in section III, the German Ks signaling system with a case study is explained in section IV and finally, the paper ends with a conclusion in section V.

II. Discrete Event Systems Based Fault Diagnosis and a Railway Point Study

An *event* is defined by [5] as an encountered specific action, unplanned incident caused by nature or a result of numerous condition which are suddenly all met. A DES is a discrete-state, event-driven system in which the state evolution of the system depends entirely on the occurrence of discrete event over time.

Representation of such a system with a model is necessary as in the conventional control theory. Events in DESs can be classified as observable and unobservable events. A system (or a software module) is said to be *diagnosable* if it is possible to detect, with a finite delay, occurrences of certain unobservable events which are referred to failure events [2]. In other words, a system is said to be diagnosable if the type of the fault is always detected within a uniformly bounded number of firings of transitions after the occurrence of the fault [6]. The diagnoser is built from the system model itself and performs diagnostics when it observes online the behavior of the system.

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) project number 115E394 – Fail-Safe PLC implementation of Interlocking System Design with Fault Diagnosis capability for Fixed-block Railway Signaling Systems.

M.S. Durmuş is with the Electrical and Electronics Engineering Department, Pamukkale University, 20070, Denizli, Turkey (phone: +90-258-296-3156, e-mail: msdurmus@pau.edu.tr).

İ. Üstoğlu is with the Department of Control and Automation Engineering, Yildiz Technical University, Istanbul, Turkey (e-mail: ustoglu@yildiz.edu.tr).

States of the diagnoser carry failure information, and occurrences of failures can be detected with a finite delay by inspecting these states [3].

Since, Finite state Machines (*FSM*) and Petri nets (*PNs*) are regarded as DES based modeling methods and particularly, these methods are also highly recommended by the railway related functional safety standards [4], it is possible to use these methods to apply DES-based fault diagnosis. Due to the page restriction, the concept of DES, *PNs* and the rules for the diagnoser construction and checking the diagnosability of a system are not given here. The reader is referred to [2], [3], [5]-[8] for detailed explanation.

Railway points are used to provide track changes. Railway points have two position indication, namely, Normal (*Nr*) and Reverse (*Rev*). Additionally, three main faults may occur in a point. These faults are identified in the software requirements specification phase of the V-model, as follows:

- F_1: Point may not reach to desired position in a predefined time while moving from *Nr* to *Rev*,
- F_2: Point may not reach to desired position in a predefined time while moving from *Rev* to *Nr*,
- F_3: Both position indications may receive at the same time.

According to these definitions, an example of diagnosable and not diagnosable *PN* models of a railway point are given in Fig. 1 and Fig. 2, respectively. The meanings of the places and the transitions of the models given in Fig. 1 and Fig. 2 are given in Table 1 and Table 2, respectively. It should be noted that, the striped places and transitions represents the unobservable places (P_{uo}) and transitions (T_{uo}) whereas the other places (P_o) and transitions (T_o) are observable. The labels (rectangles) given in Fig. 1 and Fig. 2 are used to reduce the complexity of the model and represents the related connected place.

Representation of the *PN* models given in Fig. 1 is as follows:

$$
\begin{aligned}
P_o &= \{P_{\text{PM\_1}}, P_{\text{PM\_2}}, P_{\text{PM\_5}}, P_{\text{PM\_6}}, P_{\text{PM\_8}}, P_{\text{PM\_10}}, P_{\text{PM\_12}}\},\\
P_{uo} &= \{P_{\text{PM\_3}}, P_{\text{PM\_4}}, P_{\text{PM\_7}}, P_{\text{PM\_9}}, P_{\text{PM\_11}}\},\\
T_o &= \{t_{\text{PM\_1}}, t_{\text{PM\_2}}, t_{\text{PM\_3}}, t_{\text{PM\_4}}, t_{\text{PM\_5}}, t_{\text{PM\_6}}, t_{\text{PM\_7}}, t_{\text{PM\_8}}, \ldots\\
&\quad \ldots t_{\text{PM\_9}}, t_{\text{PM\_10}}, t_{\text{PM\_11}}, t_{\text{PM\_12}}, t_{\text{PM\_13}}, t_{\text{PM\_14}}\},\\
T_{uo} &= \{t_{\text{PM\_}f1}, t_{\text{PM\_}f2}, t_{\text{PM\_}f3}\},\\
M_0 &= (M_0(P_{\text{PM\_1}}), M_0(P_{\text{PM\_2}}), M_0(P_{\text{PM\_3}}), M_0(P_{\text{PM\_4}}), \ldots\\
&\quad \ldots, M_0(P_{\text{PM\_5}}), M_0(P_{\text{PM\_6}}), M_0(P_{\text{PM\_7}}), M_0(P_{\text{PM\_8}}), \ldots\\
&\quad \ldots M_0(P_{\text{PM\_9}}), M_0(P_{\text{PM\_10}}), M_0(P_{\text{PM\_11}}), M_0(P_{\text{PM\_12}}))\\
&= (0,0,\underline{0},\underline{0},1,0,\underline{0},0,\underline{0},0,\underline{1},0).
\end{aligned}
\tag{1}
$$

Initially, the point is assumed to be in *Nr* position (place $P_{\text{PM\_5}}$ has a token). The initial position of the point in the diagnoser is illustrated by the initial state ({0,0,$\underline{0}$,$\underline{0}$,1,0,$\underline{0}$,0,$\underline{0}$,0,$\underline{1}$,0,*N*}). If all safety criteria are met, then the blades of the point moves to *Rev* position (token in place $P_{\text{PM\_5}}$ moves to place $P_{\text{PM\_6}}$) by an incoming request from the traffic control center. In this case, the state of the diagnoser will be ({0,1,$\underline{0}$,$\underline{0}$,1,0,$\underline{0}$,0,$\underline{0}$,0,$\underline{1}$,0,*N*}) by observing the marking $\hat{M}_{\text{PM\_1}}$ and the observable transition $t_{\text{PM\_2}}$.

On the other hand, if both position indications are received at the same time when the point is in initial state, that is, the marking $\hat{M}_{\text{PM\_10}}$ of the diagnoser is observed, the state of the diagnoser becomes ({0,0,$\underline{0}$,$\underline{0}$,0,0,$\underline{0}$,0,$\underline{0}$,0,$\underline{1}$,0,F_3}). Since the faults are regarded as unobservable events, the decision of any fault is realized by observing the related marking of the diagnoser, which is $\hat{M}_{\text{PM\_10}}$, in this case.

In case of detection of any predefined fault, the interlocking system warns the traffic control center and moves the system to the predetermined safe state. For instance, if this condition occurs while executing a route reservation procedure, then the reservation will be rejected immediately.

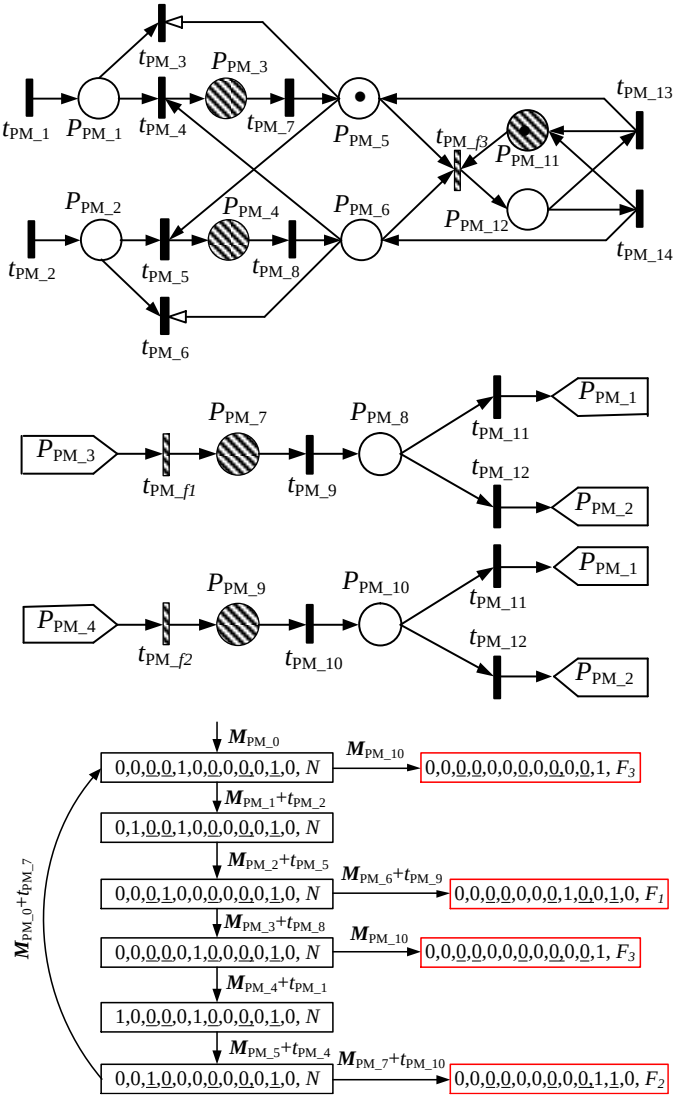


Fig. 1 *PN* model and its diagnoser (diagnosable) given in (1)

TABLE I. MEANINGS OF PLACES AND TRANSITIONS IN THE *PN* MODEL GIVEN IN FIG. 1.

Place	Meaning	Transition	Meaning
P_{PM_1}	*Nr* position requested	t_{PM_1}	Safety criteria are met, point *Nr* position request
P_{PM_2}	*Rev* position requested	t_{PM_2}	Safety criteria are met, point *Rev* position request
P_{PM_3}	Point is moving to *Nr* position	t_{PM_3} (t_{PM_6})	Request ignored
P_{PM_4}	Point is moving to *Rev* position	t_{PM_4}	Point left the *Rev* position
P_{PM_5}	Point is in *Nr* position	t_{PM_5}	Point left the *Nr* position
P_{PM_6}	Point is in *Rev* position	t_{PM_7} (t_{PM_8})	Point reached to *Nr* (*Rev*) position
P_{PM_7}	Fault type F_1 has occurred	t_{PM_9} (t_{PM_10})	Predefined filter time has expired
P_{PM_8} (P_{PM_10})	Point is faulty, F_1 (F_2)	t_{PM_11} (t_{PM_12})	*Nr* (*Rev*) position request
P_{PM_9}	Fault type F_2 has occurred	t_{PM_13} (t_{PM_14})	Point moved to *Nr* (*Rev*) position and the fault acknowledged
P_{PM_11}	Unobservable fault restriction	t_{PM_f1}, t_{PM_f2}	Point indication fault
P_{PM_12}	Point is faulty (F_3)	t_{PM_f3}	Point position fault

Representation of the Petri net models given in Fig. 2 is as follows:

$$
\begin{aligned}
&P_o = \{P_{PM\_1}, P_{PM\_2}, P_{PM\_3}, P_{PM\_4}, P_{PM\_5}, P_{PM\_6}\},\\
&P_{uo} = \{P_{PM\_7}\},\\
&T_o = \{t_{PM\_1}, t_{PM\_2}, t_{PM\_3}, t_{PM\_4}, t_{PM\_5}, t_{PM\_6}, t_{PM\_7}, t_{PM\_8}\},\\
&T_{uo} = \{t_{PM\_f1}, t_{PM\_f2}, t_{PM\_f3}\},\\
&M_0 = (M_0(P_{PM\_1}), M_0(P_{PM\_2}), M_0(P_{PM\_3}),...\\
&\quad ..., M_0(P_{PM\_4}), M_0(P_{PM\_5}), M_0(P_{PM\_6}),\\
&\quad ...M_0(P_{PM\_7}) = (0,0,1,0,0,0,\underline{0}).
\end{aligned}
\tag{2}
$$

The diagnoser given in Fig. 2 is not diagnosable because it is not possible to distinguish the type of the fault after the observation of the marking $\hat{M}_{PM\_5}$. The software model given in Fig. 2 seems to contain all of the software requirements related with the faults but the developed software model is not diagnosable which means only one of the faults F_1 or F_2 will be warned by the *PN* model. Therefore, the designers should revise their *PN* model before passing to the coding phase. Otherwise, this deficiency will result an unsuccessful test case in the module testing phase of the V-model.

In general, these kind of design faults are detected in the testing phase but by adding the diagnosability checking, it become possible to detect the faults in the design phase just before passing to the testing phase. According to EN 50128, the software requirements should fully coincide with the software model, in order to pass the software tests. In other words, since there are three kinds of faults defined in the software requirements at the beginning of the software development phase, the developed software model should fully contain the software requirements. The V-model and advantages of adding the fault diagnosis step into the V-model will be explained in the next section.

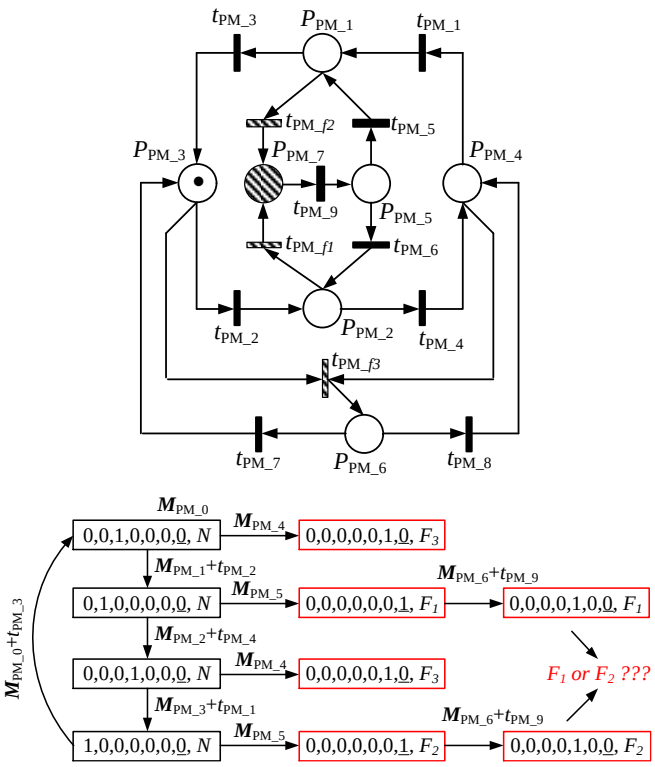


Fig. 2 *PN* model and its diagnoser (not diagnosable) given in (2)

TABLE II. MEANINGS OF PLACES AND TRANSITIONS IN THE *PN* MODEL GIVEN IN FIG. 2.

Place	Meaning	Transition	Meaning
P_{PM_1}	Point is moving to *Nr* position	t_{PM_1} (t_{PM_2})	Movement request is received and safety criteria are met for *Nr* (*Rev*) position
P_{PM_2}	Point is moving to *Rev* position	t_{PM_3} (t_{PM_4})	Point reached to *Nr* (*Rev*) position
P_{PM_3}	Point is in *Nr* position	t_{PM_5} (t_{PM_6})	Point request to *Nr* (*Rev*) position
P_{PM_4}	Point is in *Rev* position	t_{PM_7} (t_{PM_8})	Point moved to *Nr* (*Rev*) position and the fault acknowledged
P_{PM_5}	Fault type F_1 or F_2 has occurred	t_{PM_9}	Predefined filter time has expired
P_{PM_6}	Point is faulty (F_3)	t_{PM_f1} (t_{PM_f2})	Point indication fault
P_{PM_7}	Point is moving from one position to another	t_{PM_f3}	Point position fault

III. The V-Model Lifecycle and The Modified V-Model

The V-model lifecycle is introduced by Paul Rook in 1986 [9] to represent a guideline for software development processes. The main aim of the V-model is to improve the efficiency of the software development and the reliability of the software produced. The V-model offers a systematic way from project initiation to product phase out [9]. Instead of leaving the tests at the end of the project development, V-model proposes the verification of each phase of the development process. The V-model, defined in IEC61508-3 [10] is given in Fig. 3.

Before initializing a *software development process* according to the V-model, a *software planning phase* have to be realized where *software quality assurance plan*, *software verification* and *software validation plans* and *software maintenance plan* are fully defined. Later, the software requirements should be determined in cooperation with the customer and the stakeholders. By using the selected software architectures including the modeling methods, the designers develop software modules (or components). Verification of each phase is also realized immediately when the phase is over. As a summary, the left-hand side of the V-model represents the decomposition of the problem from business world to technical world [11]. After the coding phase, the right side of the V-model [10] concerns the testing of the developed software.

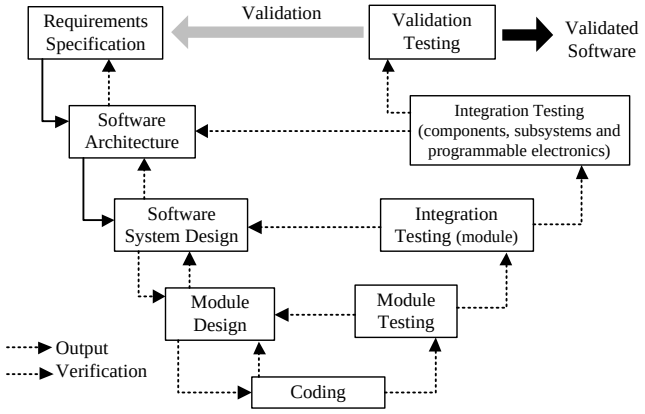


Fig. 3 The V-model, software safety integrity and development lifecycle

A. The Modified V-Model

As mentioned in [9] and [12], the workforce needed and the cost of the software development process increases with respect to the early phases of the development lifecycle. The proposed modification of the V-model in this paper enables designers to check their software modules once again before passing to the coding phase. This additional workload is realized by checking the diagnosability of each module. As mentioned in the previous section, if a module is diagnosable, then it is assumed as the software module it fully meets the software requirements; especially, the requirements related with the failure modes. This intermediate step can be seen as time-consuming extra work. But, instead of turning back from module testing to the module design phase, the proposed phase provides a last inspection of the modules before passing to the coding phase and module tests. The modified V-model is given in Fig. 4.

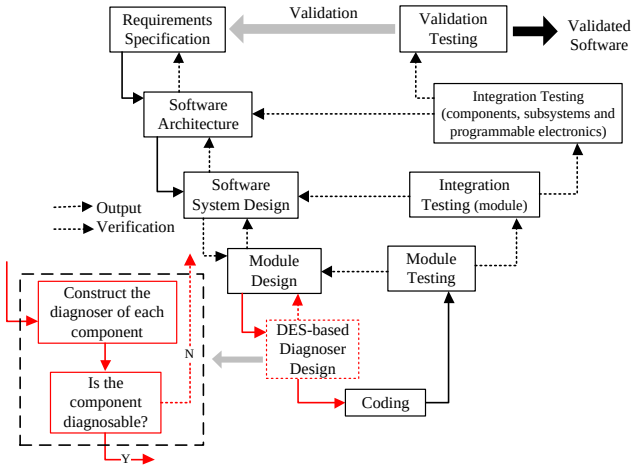


Fig. 4 The modified V-model

The proposed modification given in Fig. 4 has advantages in three ways:

1. *Checks if the constructed model covers all software requirements related with the faults*,

If the developed software model is not diagnosable, then it means that the software model does not contain all software requirements.

2. *Decrease the costs by early detecting the modeling deficiencies before passing to the coding and test phases in the V-model*,

It is obvious from the Fig. 4 that, after passing to the coding phase, the designer can only go back to module design phase at the end of the module tests. The cost of fixing an error at the design phase is 3-8 units whereas, the cost of fixing an error at the test phase becomes 21-78 units [13], [14].

3. *Enables designers more plain and simple coding*,

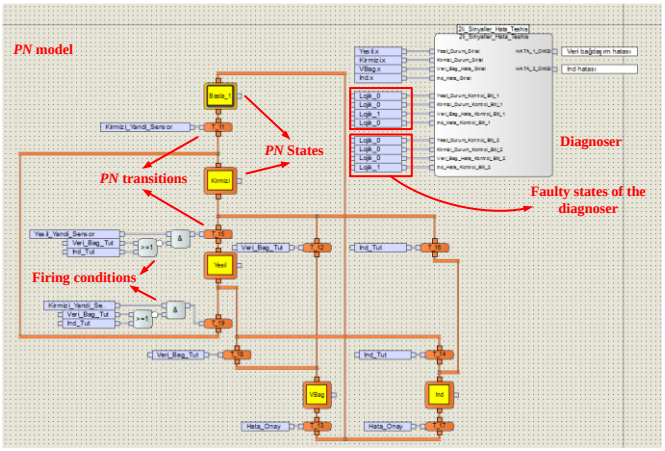
An example programmable logic controller (PLC) code snippet for a two-aspect signal model with diagnoser and without diagnoser is given in Fig. 5. It is obvious from the Fig. 5 (a) and Fig. 5 (b) that the model with diagnoser is simpler. The diagnoser compares the actual bit values of the states of the *PN* model with its faulty states.

IV. The KS Signaling System And A Case Study

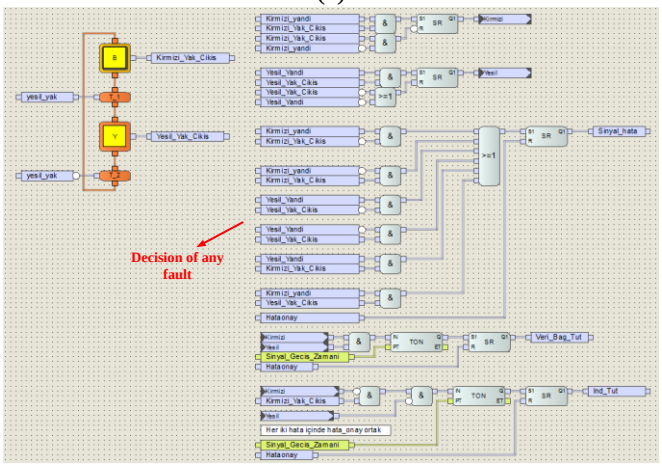
A. Fixed-Block Signaling System Components

In fixed-block railway systems, the railway lines are divided into fixed-length sections (railway tracks). Each railway track is equipped with an *axle counter* or a *track circuit* which can detect the train occupancy and, each railway track has an *entrance signal* and an *exit signal* to warn drivers about the railway track in front of them. Trains can pass from one track to another by *the points* placed at the necessary locations. Since the trains do not have any steering mechanism, they use railway points to pass from one track to another. In addition to

this, all railway traffic is provided by the cooperative work of *the traffic control center* and *the interlocking system*. Trains can move according to the route reservation procedure where a route can be assumed as cascade-connected railway tracks. *Routes* are defined at the beginning of the software development phase in the interlocking table. For detailed explanations and definitions on fixed-block signaling systems, the reader is referred to [15] [16].



(a)



(b)

Fig. 5 Two-aspect signal PN model with diagnoser (a) and conventional programming (b)

B. The Ks System

The Ks (Kombinationssignale) system was designed to ultimately replace the West German Hp system and the east German Hl System with a single new one, in connection with the installations of new electronic interlocking systems. The Ks signals like the Hl signals combine the function of distant and main signals in one single head, that is they indicate the speed after this signal, as well as the speed the next signal will induce. Note that the Ks system has three lights; green, yellow and red (see Fig. 6). It uses one colored light which basically shows the number of open blocks, whereas red means no block is open, yellow one and green two. A green light can also mean that information is only available for one block. At distant signals, the track section following the signal is treated like a block which will always be open, therefore never allowing a red light. For each block a speed and direction information can be given. Alphanumerical indicators are used to display that information. There are also additional indicators, post plates, subsidiary signals (Zs), protection signals (Sh) and etc. [17].

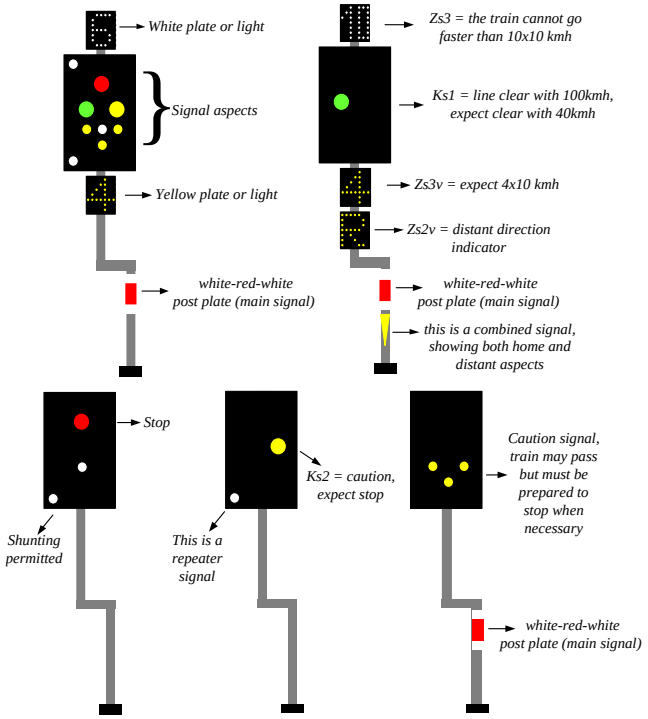


Fig. 6 Example aspects of the Ks system

C. Case Study: Constructing the PN model and diagnoser of a Ks signal

In this section, the *PN* model of signal A (see Fig. 7) and its diagnoser will be obtained. There are two possible routes from signal A (Ks signal) where the route 1 allows 160 km/h speed. Route 2 allows only 100 km/h speed and there are three other sub routes available. Signal B (Hp signal) has direction indicator which will show 'R' for the routes 2.1 and 2.2, and 'W' for the route 2.3. Additionally, the route 2.1 allows reduced speed.

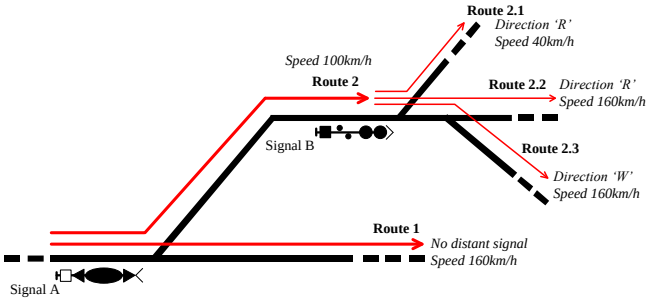


Fig. 7 Case study track layout

The signal A and the signal B will have the following main aspects:

Signal A:

- `H0`: Stop,

- `H160`: Route 1 is set and there is no speed restriction,
- `H100`: Route 2 is set and there is 100 km/h speed limit.

Signal B:

- `H0`: Stop,
- `H40R`: Route 2.1 is set and there is 40 km/h speed limit,
- `H160R`: Route 2.2 is set and there is no speed restriction,
- `H160W`: Route 2.3 is set and there is no speed restriction.

On the other hand, if the route 2 is set, then the main signals (Hauptsignal - `H`) of the signal B becomes distant signal (Vorsignal - `V`) for the signal A. Therefore, the signal A shows the following aspects (see Fig. 8):

- `H0`: Stop,
- `H160`: Route 1 is set and there is no speed restriction,
- `H100+V0`: Route 2 is set and there is 100 km/h speed limit.
- `H100+V40R`: Route 2 + Route 2.1 are set, there is 100 km/h speed limit and expect 40 km/h speed limit,
- `H100+V160R`: Route 2 + Route 2.2 are set, there is 100 km/h speed limit and expect no speed limit,
- `H100+V160W`: Route 2 + Route 2.3 are set, there is 100 km/h speed limit and expect no speed limit.

Moreover, seven faults may occur in the signal faults are identified in the software requirements specification phase of the V-model, as follows:

- F_1: Signal red color request is received but the signal is not lit,
- F_2: Signal yellow color request is received but the signal is not lit,
- F_3: Signal green color request is received but the signal is not lit,
- F_4: Red and yellow are lit at the same time,
- F_5: Red and green are lit at the same time,
- F_6: Yellow and green are lit at the same time,
- F_7: Red, yellow and green are lit at the same time.

According to the definitions given above, the *PN* model and

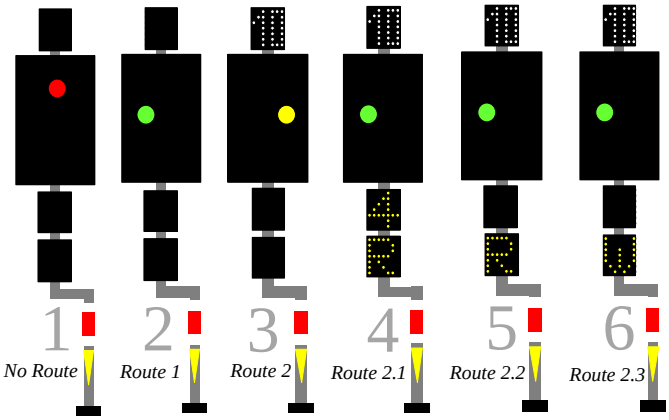


Fig. 8 Aspects of Signal A

the diagnoser of the signal A is given in Fig. 9. The meanings of the places and the transitions of the *PN* model given in Fig. 9 is given in Table 3.

$$
\begin{aligned}
P_o &= \{P_{A\_1}, P_{A\_2}, P_{A\_3}, P_{A\_4}, P_{A\_5}, P_{A\_6}, P_{A\_F1}, P_{A\_F2}, \ldots \\
&\quad \ldots P_{A\_F3}, P_{A\_F4}, P_{A\_F5}, P_{A\_F6}, P_{A\_F7}\}, \\
P_{uo} &= \{P_{A\_7}, P_{A\_8}, P_{A\_9}, P_{A\_10}, P_{A\_11}, P_{A\_12}, P_{A\_13}\}, \\
T_o &= \{t_{A\_1}, t_{A\_2}, t_{A\_3}, t_{A\_4}, t_{A\_5}, t_{A\_6}, t_{A\_7}, t_{A\_8}, t_{A\_9}, \ldots \\
&\quad \ldots t_{A\_10}, t_{A\_11}, t_{A\_12}, t_{A\_13}, t_{A\_14}, t_{A\_15}, t_{A\_16}, \ldots \\
&\quad \ldots t_{A\_17}, t_{A\_18}, t_{A\_19}, t_{A\_20}, t_{A\_21}, t_{A\_22}\}, \\
T_{uo} &= \{t_{A\_f1}, t_{A\_f2}, t_{A\_f3}, t_{A\_f4}, t_{A\_f5}, t_{A\_f6}, t_{A\_f7}\}, \\
M_0 &= \left(M_0(P_{A\_1}), M_0(P_{A\_2}), M_0(P_{A\_3}), M_0(P_{A\_4}), \ldots\right. \\
&\quad \ldots M_0(P_{A\_5}), M_0(P_{A\_6}), M_0(P_{A\_7}), M_0(P_{A\_8}), \ldots \\
&\quad \ldots M_0(P_{A\_9}), M_0(P_{A\_10}), M_0(P_{A\_11}), M_0(P_{A\_12}), \ldots \\
&\quad \ldots M_0(P_{A\_13}), M_0(P_{A\_F1}), M_0(P_{A\_F2}), M_0(P_{A\_F3}), \ldots \\
&\quad \left. \ldots M_0(P_{A\_F4}), M_0(P_{A\_F5}), M_0(P_{A\_F6}), M_0(P_{A\_F7})\right) \\
&= (0,0,0,0,0,0,\underline{0},\underline{0},\underline{0},\underline{1},\underline{1},\underline{1},\underline{0},0,0,0,0,0,0,0).
\end{aligned}
\tag{3}
$$

TABLE III. MEANINGS OF PLACES AND TRANSITIONS IN THE *PN* MODEL GIVEN IN FIG. 9.

Place	Meaning	Transition	Meaning
P_{A_1}	Signal is red	t_{A_1} - t_{A_3}	Turn signal to yellow
P_{A_2}	Signal is green	t_{A_4} - t_{A_6}	Turn signal to green
P_{A_3}	Signal is yellow	t_{A_7} - t_{A_9}	Turn signal to red
P_{A_4}	Red signal request is received	t_{A_10}	Signal is red
P_{A_5}	Yellow signal request is received	t_{A_11}	Signal is yellow
P_{A_6}	Green signal request is received	t_{A_12}	Signal is green
P_{PM_7} (P_{PM_8} or P_{PM_9})	Fault type F_1 (F_2 or F_3) has occurred	t_{A_13} - t_{A_15}	Predefined filter time has expired
P_{PM_10} - P_{PM_13}	Color fault restriction of the signal	t_{A_16} - t_{A_23}	Fault acknowledgment
P_{A_F1} - P_{A_F7}	Signal color fault	t_{A_f1} - t_{A_f7}	Related fault has occurred

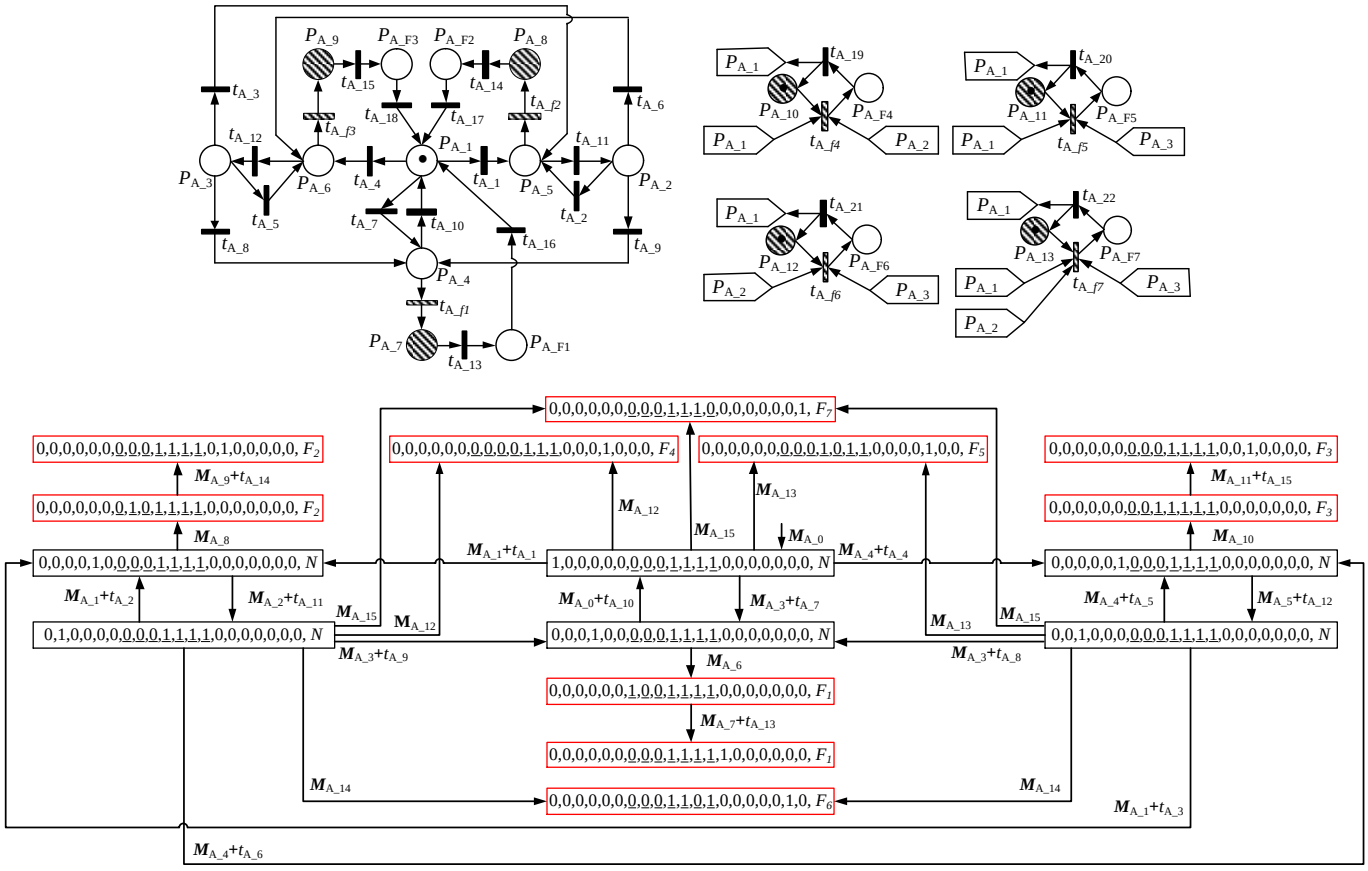


Fig. 9 *PN* model of the signal A (3) and its diagnoser

V. Conclusion

Since systems (or its subsystems) where occurrence of a fault may cause severe harm or a great number of death to human beings are considered as *safety-critical systems* (e.g. railway systems, an aircraft or nuclear power station control system), the development steps of such safety-critical software must be carried out very carefully. The recommendations of the international safety standards and the national rules have to be considered by the designers, developers and engineers to satisfy the required safety level and fulfill the software requirements.

By adding the DES-based diagnoser design and the diagnosability check in to the V-model enables the designers to verify their models before passing to the testing phase and additionally, it become possible to detect the model deficiencies in the design phase.

Modification of the V-model by adding the application of the DES-based fault diagnosis can be seen as a time-consuming and extra workload to the project work group but provides a cross check between the obtained models and the software requirements, decrement in the cost of fixing the faults in the software development process and an increment in the readability of the code.

References

[1] *IEC 61508-7, Functional safety of electrical/electronic/programmable electronic safety-related systems, Part 7: Overview of techniques and measures*, European Committee for Electrotechnical Standardization, Brussels, 2010.

[2] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "Diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 40(9), pp. 1555-1575, 1995.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Transactions on Control Systems Technology*, vol. 4(2), pp. 105-124, 1996.

[4] *EN 50128, Railway Applications, Communications, signalling and processing* systems, Software for railway control and protection systems, European Committee for Electrotechnical Standardization, Brussels, 2011.

[5] C.G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, 2nd ed., Springer, 2008.

[6] T. Ushio, I. Onishi and K. Okuda, "Fault detection based on Petri net models with faulty behaviors," in *Proceedings of the IEEE International Conference on System, Man and Cybernetics*, San Diego, California, USA, 1998, pp. 113-118.

[7] T. Murata, "Petri nets: Properties, Analysis and Applications," *Proceedings of IEEE*, vol. 77, pp. 541-580, 1989.

[8] M.S. Durmuş, S. Takai and M.T. Söylemez, "Fault Diagnosis in Fixed-Block Railway Signaling Systems: A Discrete Event Systems Approach," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 9(5), pp. 523-531, 2014.

[9] P. Rook, "Controlling Software Projects," *Software Engineering Journal*, vol. 1(1), pp. 7-16, 1986.

[10] *IEC 61508-3, Functional Safety of Electrical/Electronic/Programmable electronic safety-related systems, Part 3: Software requirements*, European Committee for Electrotechnical Standardization, Brussels, 2010.

[11] A. Ratcliffe, "Software Development with the V-model," in *SAS Global Forum 2011, Coders' Corner*, Las Vegas, Nevada, USA, 2011, pp. 1-9.

[12] B.W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," *IEEE Transactions on Software Engineering Journal*, pp. 75-88, 1984.

[13] B.W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering Journal*, vol. SE-10(1), pp. 4-21, 1984.

[14] J.M. Stecklein, J. Dabney, D. Brandon, B. Haskins, R. Lovell, G. Moroney, "Error Cost Escalation Through the Project Life Cycle," in *14th Annual International Symposium*, Toulouse, France, 2004.

[15] S. Hall, *Modern Signalling Handbook*. 3rd ed. Ian Allan Publishing, England, 2001.

[16] M.S. Durmuş, U. Yıldırım and M.T. Söylemez, "The Application of Automation Theory to Railway Signaling Systems: The Turkish National Railway Signaling Project," *Pamukkale University Journal of Engineering Sciences*, vol. 19(5), pp. 216-223, 2013.

[17] DB Netz AG. (2016). Richtlinie 301 - Signalbuch (9th ed.) [Online]. Available: http://www1.deutschebahn.com/file/fahrweg-de/10223642/EiWOeEUgs03R-yZ9ic_wCK5yUrc/10402350/data/rw_301_aktualisierung_9.pdf

M. S. Durmuş (M'07) received the B.S. and M.S. degrees from Pamukkale University (PAU), Turkey, in 2002 and 2005, respectively, and the Ph.D. degree from Istanbul Technical University (ITU) in 2014. From 2007 to 2014, he was a Research and Teaching Assistant with the Department of Control and Automation Engineering, ITU. He is currently a Research and Teaching Assistant with the Department of Electrical and Electronics Engineering, PAU. He also received a second Ph.D. degree from the Division of Electrical, Electronic and Information Engineering, Osaka University, as a Ronpaku fellow of the Japan Society for the Promotion of Science. His research interests include railway signaling systems, functional safety, linear control theory, fault diagnosis of discrete event systems.

İlker Üstoğlu received a B.Sc. degree in electrical engineering from Istanbul Technical University (ITU), Turkey, in 1997 and a M.Sc. degree in control and computer engineering from ITU, in 1999. He completed his Ph.D. in control and automation engineering at ITU in 2009. Since September 2010, he has been working at the Control and Automation Engineering Department of Yildiz Technical University, Turkey. His research areas include linear control theory, computer algebra, fuzzy sets and systems, railway systems and functional safety.