Safety assessment of computerized railway signaling equipment supported by formal techniques

Tadeusz CICHOCKI [1], Janusz GÓRSKI [2]

[1] Adtranz Zwus, ul. Modelarska 12, 40-142 Katowice, Poland,
e-mail: tadeusz.cichocki@plsig.mail.abb.com
[2] Department of Applied Informatics, Technical University of Gdańsk,
ul. Narutowicza 11/12, 80-952 Gdańsk, Poland,
e-mail: jango@pg.gda.pl

Abstract. One of possible ways to achieve a very high level of confidence in the system is to develop its adequate model and then to analyze the properties of this model. An application of formal modeling method in the area of safety assessment of computerized railway signaling systems is presented.

1. Introduction

As a result of the progress in technology development the increasing number of applications include a software component which can directly affect risks associated with the system. In some applications, e.g. rail and air transportation, medical, nuclear, a very high level of confidence that the system will not do any harm has to be achieved before the system is commissioned to use. The criteria to be fulfilled and the guidance how to do this are provided by numerous national, international and sectoral standards and regulations, e.g. [1, 2, 3].

One of possible ways to achieve a very high level of confidence in the system is to develop its adequate model and then to analyze the properties of this model. If the model is a valid representation of the problem under consideration, the properties of the model can be understood as the properties of the real system. To achieve a very high confidence in the analytical results, the model and the associated analytical framework are often formal, i.e. are represented in terms of mathematical objects.

In this paper we show how the formal method, namely Z [11], is used to model the problem related to railway signaling. The model is then used to support various analyses and to drive the design process. One of its possible uses which we are working on now is to support a (formal) FMEA (Failure Mode and Effect Analysis) [5, 6, 7].

2. The method

We are following the modeling method introduced in [8, 9]. The method assumes hierarchical specification of properties of a system and verification of consistency between subsequent layers of the hierarchy. The method comprises the following stages.

Stage 1: Specification. This includes the following steps:

- Application of object-oriented modeling to find representation of the system that includes objects (classes), attributes and associations of interest;
- Building a formal specification of the object model.

Stage 2: Validation. In this stage the specification is analyzed in order to ensure that it adequately captures the considered problem. This step is not subjected to formalization.

Step 3: Analysis. This stage concentrates on the consequences of the specification. It may involve a range of approaches, from informal reading and interpretation of (formal) specifications to proving theorems from the axioms included in the specification. The specifications can be checked against a list of criteria including: type consistency, syntax correctness, definition of all types, functions, operations and correctness constraints, completeness of all logical cases, completeness of specifications of initial values and state changes (compare specification agendas and validation criteria in [10]).

Step 4: Extension. Here we go through a chain of steps where each step extends the model with new objects or decomposes some objects from the previous model into objects, attributes and relations in the subsequent model. As the result we get more detailed (less abstract) representations of the problem under consideration. Each extension step consists of the following substeps:

- extend the present specification,
- define the mapping between the abstract (higher level) and specific (lower level) state spaces,
- verify the new specification against the previous one.

3. Line Block System specifications

The main goal of any railway system is to maintain the required railway traffic level through a specific line or region with a required safety guarantees. Years of experience lead to a definition of adequate traffic rules to form a functional safety specifications. Each railway system must perform according to the rules both in the case of normal operation as well as in case of a component failure. The Line Block System (LBS) is supposed to control railway traffic on a rail-track between two stations. It uses light signals shown on semaphores, each protecting a line segment (block) of the rail-track. Each segment is tested in order to determine its occupation or its availability to move a train toward the next segment. The main goal is then to achieve separation of trains on the line and to allow them smooth passing in a required direction.

3.1. Level 0 – object model

At the highest abstraction level a system is represented by the model shown in Fig. 1. The only attribute of the system is *Safe*. It assumes Boolean values (possible states of the system are classified as being either safe or unsafe).

Fig. 1. LINE_BLOCK_SYSTEM (Level 0 structure).

3.2. Level 0 – formal specification

```
┌─ SYSTEM ──────────────────────
│ Safe: Boolean
├───────────────────────────────
│ Safe=TRUE                        -- Safety requirement Level 0.
│                                  -- Only safe states are allowed.
└───────────────────────────────
```

3.3. Level 1 – object model

In the next level safety is achieved by physical separation of trains on the track. Disjoint units of the rail-track are defined and then the safety requirement demands that each unit (except the utmost ones) is occupied by at most one train. In the object model, the system is composed of instances of TRAIN and UNIT classes and the occupation of units by trains is represented by the OCCUPATION association. It is assumed that a train is either completely included in a block or spans over two adjacent blocks.

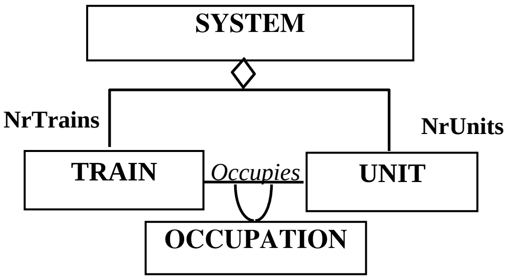

Fig. 2. LINE_BLOCK_SYSTEM (Level 1 structure).

3.4. Level 1 – formal specification

```
    NrTrains : N                          -- Number of trains in the system.

  ┌──────────────────────────────────
  │ NrUnits : N
  ├──────────────────────────────────
  │ NrUnits >= 3                       -- Number of units in the system
  └──────────────────────────────────   -- (at least one block and two stations).

Unit_identifier : {1 .. NrUnits}          -- Identifier of a unit in the system.
Train_identifier : {1 .. NrTrains}        -- Identifier of a train in the system.

-- Class TRAIN

  ┌─  TRAIN[Train_identifier] ─────────────────
  │ Train_front : {1 .. NrUnits}
  │ Train_end : {1 .. NrUnits}
  ├────────────────────────────────────────────
  │ (Train_front - Train_end) ∈ {0, 1}           -- The length of any train must be less
  └────────────────────────────────────────────   -- then the size of one unit (block).

  ┌─  InitTRAIN[Train_Identifier]   ────────────
  │ TRAIN’
  ├────────────────────────────────────────────
  │ Front_Train = End_Train ∈ {1, NrUnits}       -- Initially a train is entirely included in
  └────────────────────────────────────────────   -- some block.
```

-- Class UNIT

UNIT[Unit_identifier]

Occupation : {YES, NO }

U_Traffic_direction : {-1, 0, 1}

Each unit between the first and the last represents a rail-track segment. Units number 1 and NrUnits represent two stations connected by a direct rail-track. Each unit has a traffic direction label associated with it (it tells the intended direction of traffic through this unit). We use the convention that “1” means “left-to-right”, “-1” means “right-to-left” and “0” means “undefined”. Additionally, each unit has the indicator of its occupation by a train.

InitUNIT[Unit_identifier]

UNIT’[Unit_identifier]

Occupation’ = NO

U_Traffic_direction’ = 0

function Id : UNIT[Unit_identifier] → {1 .. NrUnits}

Id(u) = Unit_identifier

function occupied : TRAIN×UNIT → BOOL

occupied(t, u) = (t.Train_front = Id(u) ∨ t.Train_end = Id(u))

OCCUPATION

TRAIN
UNIT

$\forall\ t : \mathrm{TRAIN} \bullet$ -- (1)
$(t.\mathrm{Train\_front} - t.\mathrm{Train\_end}) \in \{0, 1\}$

$\forall\ u : \mathrm{UNIT} \bullet$ -- (2)
$(\ \exists\ t : \mathrm{TRAIN} \bullet \mathrm{occupied}(t, u)\ ) \Rightarrow u.\mathrm{Occupation} = \mathrm{YES}$

$\forall\ t : \mathrm{TRAIN} \bullet$ -- (3)
$(t.\mathrm{Train\_front} = 1 \wedge t.\mathrm{Train\_end} = 1) \Rightarrow$
$t.\mathrm{Drive\_direction} \in \{0, 1\}$

$\forall\ t : \mathrm{TRAIN} \bullet$ -- (4)
$(t.\mathrm{Train\_front} = \mathrm{NrUnits} \wedge t.\mathrm{Train\_end} = \mathrm{NrUnits}) \Rightarrow$
$t.\mathrm{Drive\_direction} \in \{-1, 0\}$

$\forall\ t1, t2 : \mathrm{TRAIN} \bullet$ -- (5)
$\forall\ u : \mathrm{UNIT} \bullet$ -- ***Safety requirement Level 1***
$((t1 \neq t2) \wedge \neg \mathrm{Id}(u) \in \{1, \mathrm{NrUnits}\}) \Rightarrow$
$(\ \mathrm{occupied}(t1, u)\ \mathrm{XOR}\ \mathrm{occupied}(t2, u)\ )$

The following properties are stated by OCCUPATION:

(1) a train can span over at most two adjacent units,
(2) the units pointed to by the front and the end of the train are *occupied,*
(3),(4) being at a station the train goes toward the other station,
(5) a unit (except a station) can be occupied by at most one train (this is the safety requirement for the system).

3.5. Level 2 – object model

In the next level we add the LPC (*Local Point of Control)* class to the system and we specialize the UNIT class by explicit differentiation between the BLOCK and STATION units. The relationship among those classes is given by the LBC_Control_Rules association. The next specification level may add new classes like SEMAPHORE, CONTROLLER, LINE_SEGMENT and WHEEL_DETECTOR. In Fig.3. the specification levels down to Level 3 are shown. For sake of simplicity we do not show class associations.

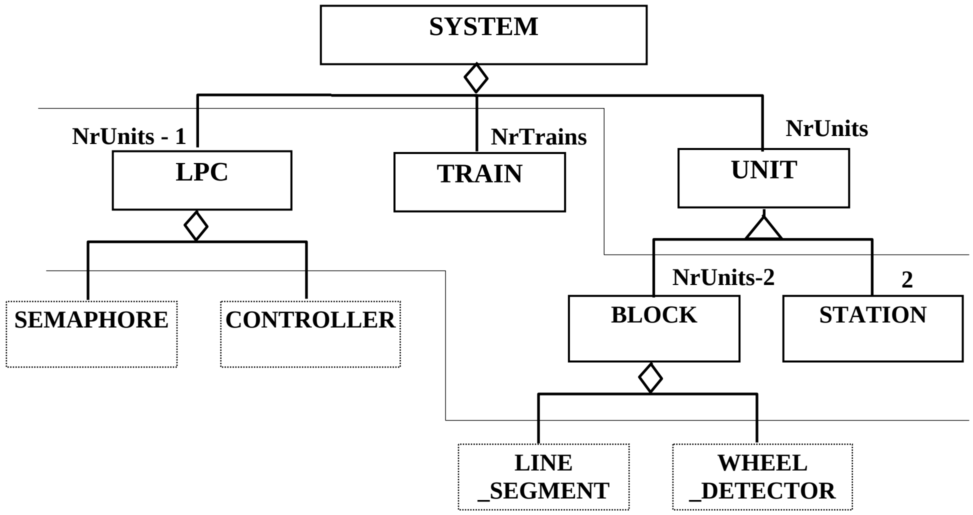

Fig. 3. LINE_BLOCK_SYSTEM specification levels.

3.5. Level 2 – formal specification

We define the correspondence between *Unit_identifier* and *Station_identifier* and *Block_identifier* values as follows.

```
Unit_identifier
Station_identifier
Block_identifier
________________________________________
Station_identifier = -1
⇔ Unit_identifier = 1

Station_identifier = 1
⇔ Unit_identifier = NrUnits

Unit_identifier ∉ {1, NrUnits}
⇔ Block_identifier = Unit_identifier
```

-- Class BLOCK

Block_identifier : {2 .. NrUnits-1}

```
┌─ BLOCK[Block_identifier] ──────────
│ UNIT[U_Traffic_direction /                -- The attribute renaming.
│       B_Traffic_direction]
│
├────────────────────────────────────
│
└────────────────────────────────────
```

```
┌── InitBLOCK[Block_identifier] ─────
│ InitUNIT'[Unit_identifier]
│
├────────────────────────────────────
│
└────────────────────────────────────
```

-- Class STATION

Station_identifier : {-1, 1}

```
┌─ STATION[Station_identifier] ──────────────
│ UNIT[U_Traffic_direction /
│       S_Traffic_direction]                      -- The attribute renaming.
│
│ Station_view :                                  -- As compared to line block
│ {S, S0, S1, ..., S13, S10A, S11A, S12A, S13A}   -- signals, Set of station signals
│                                                 -- is extended {S6, ..., S13A}.
│
│ Decision :
│ {Set_up_1, Set_up_-1, Stop_1, Stop_-1, Restart,
│ No_decision}
├────────────────────────────────────────────
│ S_Traffic_direction ∉ {0, Station_identifier}  -- Decisions on the Line block
│ ⇒ Decision = No_decision                       -- state are taken by the station
│                                                -- which is a target of the rail
│                                                -- traffic on the line or both
│                                                -- stations if the direction is not
└────────────────────────────────────────────-- defined.
```

InitSTATION[Station_identifier]

STATION'[Station_identifier]
InitUnit [Unit_identifier]

Station_view' = S1
Decision' = Restart

-- Class LPC
LPC_identifier : {(i, i+1): i ∈ {1, ..., NrUnits}} -- LPC identifier.

We think of LPC[(*i*, *i+1*)] as an object situated between the blocks with Block_identifier *i* and *i+1* in order to show proper control signals according to the rules of the line block system.

LPC[LPC_identifier]

L_Traffic_direction : {-1, 0, 1}
Signal_for : { {-1, 1} → {S, S0, S1, ..., S13, S10A, S11A, S12A, S13A} }

Signal_for(- L_Traffic_direction) = S0

InitLPC[LPC_identifier]

LPC'

L_Traffic_direction' = 0

Signal_for' = {1 → S0} ⊕ {-1 → S0}

The class TRAIN is extended with some new attributes which indicate the driving direction and speed of a train. We will use the same name for a redefined class.

-- Class TRAIN

TRAIN[Train_identifier]

TRAIN[Train_identifier] -- *The class of level 1 structure.*
Drive_speed : $\mathbf{R}_+ \cup \{0\}$ -- *Drive speed of the train.*
Drive_direction : $\{-1, 0, 1\}$ -- *0 stands for "direction is not defined".*

Drive_direction = 0 $\Rightarrow$ Drive_speed = 0 -- *Undefined direction is allowed for*
-- *stopped trains only.*
(Train_front – Train_end) $\in \{0, 1\}$ -- *The length of any train must be less*
-- *then the length of one unit (block).*

InitTRAIN[Train_Identifier]

TRAIN' -- *The class of the last definition.*
InitTrain[Train_Identifier] -- *The initialisation for the level*
-- *1 structure*

Drive_speed' = 0
Drive_direction' = 0

The OCCUPATION scheme of Level 2 inherits properties from the OCCUPATION scheme of Level 1.

In order to show how the control is effected we introduce the indexed sequence of LPC (*Local Point of Control*) objects. The system structure of level 2 includes a LINE_BLOCK_CONTROL_RULES association which embodies the safety requirements as stated in the user defined requirements [4].

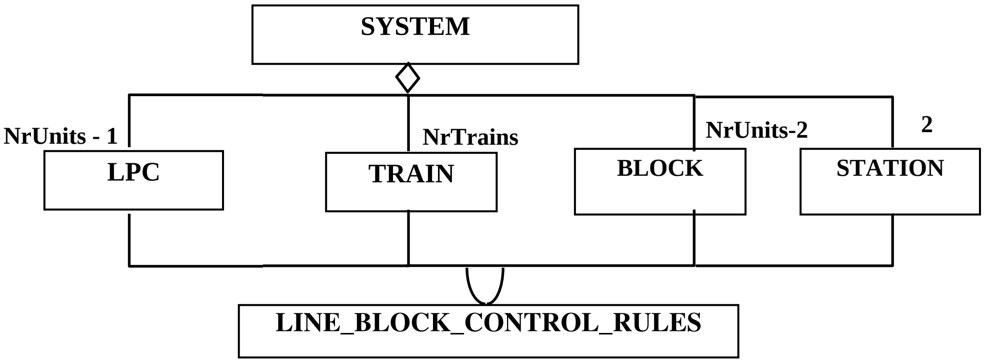

Fig. 4. LINE_BLOCK_CONTROL_RULES (Level 2 structure).

The informal description of the requirements is given below.

Table 1. Line Control Rules as may be found in PKP regulations [4].

Safety requirement 1: Drive direction of any train must conform to the traffic direction set up in the line block system. If the traffic direction in the line block system is not set up then any train should stop (or continue its driving according to the rules outside the line block system).

Safety requirement 2: Permission signal of the LPC protecting the line block may be shown only in the case when this segment is not occupied, and under the condition that the next segment is protected by "STOP" signal or is not occupied.

Safety requirement 3: Each occupied line block must be protected by "STOP" signal on its protecting LPC.

Safety requirement 4: The system does not allowed to change a traffic direction on the line if at least one of the line blocks is occupied.

The formal expression of those requirements is given below.

LINE_BLOCK_Control_Rules

BLOCK[2] ∧ .. ∧ BLOCK[NrUnits-1]
LPC[2] ∧ .. ∧ LPC[NrUnits]
TRAIN[1] ∧ .. ∧ TRAIN[NrTrains]
STATION[-1] ∧ STATION[1]
LB_Traffic_direction : {-1, 0, 1}

(LB_Traffic_direction ∈ {-1, 1} ∧ -- *Safety requirement 1.*
∀ i : 1 .. NrBlocks • ∃ Tra:TRAIN • Tra.Train_front = i)
⇒ (1)

(Tra.Drive_direction = LB_Traffic_direction
$\wedge$
Tra.Drive_direction = BLOCK[i].B_Traffic_direction)

(LB_Traffic_direction = 0 $\wedge$
$\forall$ i : 1 .. NrBlocks $\bullet$ $\exists$ Tra:TRAIN $\bullet$ Tra.Train_front = i) (2)
$\Rightarrow$ Tra.Drive_speed = 0

$\forall$ i : 0 .. NrBlocks-1 $\bullet$ *-- Safety requirement 2.*
LPC[i, i+1].Signal_for(1) $\in$ { S2, S3, S4, S5 }
$\Rightarrow$ (3)
(BLOCK[i+1].Occupation = NO $\wedge$
(BLOCK[i+2].Occupation = NO $\vee$
LPC[i+1, i+2].Signal_for(1) = S1))

$\forall$ i : 1 .. NrBlocks $\bullet$
LPC[i, i+1].Signal_for(-1) $\in$ { S2, S3, S4, S5 }
$\Rightarrow$ (4)
(BLOCK[i].Occupation = NO $\wedge$
(BLOCK[i-1].Occupation = NO $\vee$ LPC[i, i-1].Signal_for(1) = S1))

$\forall$ i : 1 .. NrBlocks $\bullet$ *-- Safety requirement 3.*
BLOCK[i].Occupation = YES
$\Rightarrow$ (5)
((LB_Traffic_direction = 1 $\Rightarrow$ LPC[(i-1, i)].Signal_for(1) = S1)
$\wedge$
(LB_Traffic_direction = -1 $\Rightarrow$ LPC[(i, i+1)].Signal_for(1) = S1))

$\exists$ i $\in$ {i .. NrBlocks} $\bullet$ *-- Safety requirement 4.*
(BLOCK[i].Occupation = YES $\wedge$ LB_Traffic_direction = 1)
$\Rightarrow$ (6)
STATION[1].Decision $\in$ {Set_up_1, Stop_1, No_decision}

$\exists$ i $\in$ {i .. NrBlocks} $\bullet$
(BLOCK[i].Occupation = YES $\wedge$ LB_Traffic_direction = -1)
$\Rightarrow$
STATION[-1].Decision} $\in$ {Set_up_-1, Stop_-1, No_decision} (7)

4. The FMEA technique

Identification of the system architecture, modeling of functional dependencies between components, developing the inventory of potential faults of different components, and identification of error propagation scenarios initiated by these faults are the basic activities of the FMEA (*Failure Mode and Effect Analysis*) technique [6].

FMEA starts from the lowest (base) level of the components. Those are the components which internal structure is not interesting from the point of view of the analysis. Scenarios of failure development in the system initiated by the anticipated components faults are identified and documented. Effects of the components faults are interpreted in terms of higher levels of the functional structure, and finally by their impact on the system external interfaces. Links between the scenarios and the proposed actions aiming at risk elimination or reduction are documented as well. The analysis is used to drive the design process.

Ability of safe running of trains is the basic *functional safety* goal of the Line Block system in its normal behavior. However, for a railway control system there is an additional requirement of *structural safety* that postulates that the system remains safe even when failures of the components occur. Analysis of possible failures and their impact has to be done in order to define proper procedural and architectural solutions (*safety reactions* of the system).

FMEA is based on a system component partitioning structure. The question for the technique to be answered is: is there any (unwanted but possible) change of relations and dependencies between components with the consequences contradicting the specified safety goals?

We distinguish the following groups of faults:

1. changes of attribute values outside their type,
2. changes of object invariants,

and look for those causing undesired states of the system and transitions to these states.

Each specification level gives an interpretation of the faults arising at previous step and extends the list to the dependencies of that level.

For the second group of faults at the levels 1 and 2 specifications of the LBS system we receive the following tables summarizing the specification of failure behavior invariants:

-- Class TRAIN

Condition	**Consequence**
$\text{Drive\_direction} = 0 \wedge \text{Drive\_speed} > 0$	*Physical impossibility.*
$(\text{Train\_front} - \text{Train\_end}) > 1$	*Physical impossibility.*

-- Class OCCUPATION

Condition	**Consequence**
$\exists\, t : \text{TRAIN} \bullet$ $(t.\text{Train\_front} - t.\text{Train\_end}) > 1$	*Physical impossibility*
$\exists\, u : \text{UNIT} \bullet$ $(\exists\, t : \text{TRAIN} \bullet \text{occupied}(t, u)) \wedge u.\text{Occupation} = \text{NO}$	Hazard Train detection fault
$\exists\, t : \text{TRAIN} \bullet$ $(t.\text{Train\_front} = 1 \wedge t.\text{Train\_end} = 1) \wedge$ $t.\text{Drive\_direction} = -1$	*Physical impossibility*
$\exists\, t : \text{TRAIN} \bullet$ $(t.\text{Train\_front} = \text{NrUnits} \wedge t.\text{Train\_end} = \text{NrUnits}) \wedge$ $t.\text{Drive\_direction} = 1$	*Physical impossibility*
$\exists\, t1, t2 : \text{TRAIN} \bullet$ $\exists\, u : \text{UNIT} \bullet$ $((t1 \neq t2) \wedge \neg \text{Id}(u) \in \{1, \text{NrUnits}\}) \wedge$ $(\text{occupied}(t1, u) = \text{occupied}(t2, u) = .T.)$	Hazard

-- Class STATION

Condition	**Consequence**
$\text{S\_Traffic\_direction} \notin \{0, \text{Station\_identifier}\}$ $\wedge\ \text{Decision} \neq \text{No\_decision}$	Hazard

-- Class LPC

Condition	**Consequence**
$\text{Signal\_for}(-\text{L\_Traffic\_direction}) \neq \text{S0}$	*Misleading information*

-- Class LINE_BLOCK_Control_Rules

Condition	**Consequence**
(LB_Traffic_direction ∈ {-1, 1} ∧ ∃ i : 2 .. NrUnits-1 • ∃ Tra:TRAIN • Tra.Train_front = i) ∧ (Tra.Drive_direction ≠ LB_Traffic_direction ∨ Tra.Drive_direction ≠ BLOCK[i].B_Traffic_direction)	Hazard Project decision: Allow LB_Traffic_direction communication to the train
(LB_Traffic_direction = 0 ∧ ∃ i : 2 .. NrUnits-1 • ∃ Tra:TRAIN • Tra.Train_front = i) ∧ Tra.Drive_speed > 0	Hazard Project decision: Allow LB_Traffic_direction communication to the train
∃ i : 2 .. NrUnits-2 • LPC[i, i+1].Signal_for(1) ∈ { S2, S3, S4, S5 } ∧ (BLOCK[i+1].Occupation = YES ∨ (BLOCK[i+2].Occupation = YES ∧ LPC[i+1, i+2].Signal_for(1) ≠ S1))	Hazard
∃ i : 2 .. NrUnits-1 • LPC[i, i+1].Signal_for(-1) ∈ { S2, S3, S4, S5 } ∧ (BLOCK[i].Occupation = YES ∨ (BLOCK[i-1].Occupation = YES ∧ LPC[i, i-1].Signal_for(1) ≠ S1))	Hazard
∃ i : 2 .. NrUnits-1 • BLOCK[i].Occupation = Yes ∧ ((LB_Traffic_direction = 1 ∧ LPC[(i-1, i)].Signal_for(1) = S1) ∨ (LB_Traffic_direction = -1 ∧ LPC[(i, i+1)].Signal_for(1) ≠ S1))	Hazard
∃ i : 2 .. NrUnits-1 • (BLOCK[i].Occupation = Yes ∧ LB_Traffic_direction = 1)	Hazard

$\wedge$ STATION[1].Decision $\notin$ {Set_up_1, Stop_1, No_decision}	
$\exists$ i : 2 .. NrUnits-1 $\bullet$ (BLOCK[i].Occupation = Yes $\wedge$ LB_Traffic_direction = -1) $\wedge$ STATION[-1].Decision $\notin$ {Set_up_-1, Stop_-1, No_decision}	Hazard

5. Lessons learned

One of the project aims is to improve the techniques presently used in the company in the context of safety analysis. The project is still in progress and is performed in parallel to traditional forms of analysis and documentation started earlier. However what is to be noticed about the "formal way" now is:

- a significant concentration on details and strong developer guidance to the problems and solutions at the first stages of the project; formal notation forces designers to concentrate on system structure and properties details early leading to specification completeness;
- formal specifications support systematic search for possible deviations, inconsistencies and their consequences;
- the method and formal notation used significantly improves precision and compactness of the specifications. Although it needs some training in formal notation, it may be used in an everyday engineering practice.

In further work we introduce formal descriptions of dynamic behavior of system components and extend the formal FMEA to include analysis of faults related to the properties different than state invariants.

References

1. EN 50126: Railway applications. The Specification and Demonstration of Dependability, Reliability, Availability, Maintainability and Safety (RAMS), CENELEC, Final Draft version, May 1998.
2. EN 50128: Railway applications. Software for railway control and protection sytems, CENELEC, Final Draft version, July 1998.
3. ENV 50129: Railway applications. Safety Related Electronic Systems for Signalling, CENELEC, May 1998.
4. "Wytyczne Techniczne Budowy Urzadzen Sterowania Ruchem Kolejowym w Przedsiebiorstwie PKP", WTB-E10, Polish State Railways, Warszawa 1996.

5. Cichocki, T., Górski, J., Application of FMEA to Safety Analysis of Industrial Computer Applications, (*in Polish*) III Konferencja Naukowo-Techniczna “Diagnostyka Procesów Przemyslowych”, Jurata k/Gdańska, 7-10 of September, 1998.
6. Cichocki, T., Górski, J., Safety assessment of computerized railway signalling equipment, Proc. of CENELEC Workshop SC9XA/WGA10, Münich (Germany), May 11, 1999.
7. Cichocki, T., Górski, J., Safety Assessment of a Line Block System, (*in Polish*) VI Konferencja “Systemy Czasu Rzeczywistego”, Zakopane 27-30 September 1999.
8. Górski, J., Formal System-Level Safety Analysis, IEEE publication, 1995.
9. Górski, J., Specification and Analysis of Safety Related Systems - Application of Temporal Logic, ELEKTRONIKA Nr 68, 1990, Technical University of Gdańsk (in Polish).
10. Heisel, M., Methodology and Machine Support for the Application of Formal Support for the Application of Formal Techniques in Software Engineering, Habilitation Thesis, Technische Universität Berlin, Berlin, 1997.
11. Spivey, J. M., The Z Notation: A Reference Manual, First published by Prentice Hall International (UK) Ltd., 1992 (Second edition), ISBN 0-13-629312-3.