A Formal Model of Requirements

François Defossez[*,1,2], Simon Collart-Dutilleul[1,3] and Philippe Bon[1,2]

[1]*Univ Nord de France, F-59000 Lille, France*

[2]*INRETS/ESTAS, 20 rue Élisée Reclus BP 317 59666 Villeneuve d'Ascq CEDEX, France*

[3]*École Centrale de Lille/LAGIS, Cité scientifique BP 48 59651 Villeneuve d'Ascq, France*

Abstract: This paper introduces a methodology to analyze the safety of timed discrete event systems. Our case-study is the level crossing, a critical component for the safety of railway systems. First, our goal is to take out the forbidden state highlighted by a p-time Petri net modelling. This model deals with the requirements of the considered system and has to contain all the constraints that have to be respected. Then we describe a process identified as a solution for the system functioning. This method consists in exploring all the possible behaviors of the system by means of the construction of state classes. Finally, we check if the proposed process corresponds to the model of requirements previously built.

Keywords: Temporal and safety requirements, requirement engineering, traceability, modelling, Petri net, Railway transport, level crossing.

INTRODUCTION

This paper deals with the general problem of safety critical systems processing. The objective is to describe a methodological approach which proposes to use formal modelling in order to provide some tools for requirement engineering.

In order to fulfill some needs in regards to the safety of railway systems, European specifications and standards are introduced and materialize in different practices and cultures. Thereby, for example, Technical Specifications for Interoperability are regulation texts approved by the European Union and which recommend the application of standards such as the 50128 CENELEC standard [1]. It deals with software for railway control and protection systems and brings up notions such as requirements and tracability. The authors are convinced that formal methods give a relevant contribution to answer the problem considered. Actually, formal methods allow a mathematical expression of requirements. However requirements need to be represented as simply and non ambiguously as possible. Requirements also need to be assessed by professional specialists. We put forward the hypothesis that the UML notation, Petri nets and the method are widely used to model railway systems. Among other applications, they are used to model and validate real-time distributed railway systems. A benefit of using UML [2] is its status as an international standard and its widespread use in the software industry. It is used to identify the requirements and describe the system. Its graphical nature makes discussions easier for the different actors of a project. Nevertheless, it is only a semi-formal modelling tool. Then, the method, introduced by J.R. Abrial [3] is a formal method for the development of specifications and their refinements to an implementation, already used in the railway industry [4]. Its notation may be complex and difficult to understand. Moreover, a model analysis requires mathematical skills which are unusual in industry. Finally, Petri nets [5] combine three important features: a graphical representation, a dynamic behavior and an abstraction of the treatments.

For these reasons, we propose to use Petri nets as a graphical formal model. Graphical tools are used to express first-level requirements which are strong and mandatory constraints. Consequently, their violation is strictly forbidden. The model provided herein can highlight mathematical dependencies ensuing from first-level requirements. This aspect defines the main contribution of this paper. During the different process phases, requirements have to be sharply propagated into the downstream phases. Several types of requirements can be identified; first-level (source) requirements resulting from norms (rules) and informal specifications (expression of needs), first level dependencies coming from first level requirements propagation, technical requirements related to technical choices and then technical dependencies. In order to make traceability easier, requirement propagation has to be document for verification purposes.

As seen previously, the paper deals with requirement engineering, but more precisely, it focuses on temporal requirements.

This paper is divided into 6 parts. After this introduction, the approach is argued and the methodology is detailed by presenting the different models, their use and the state classes for the requirements assessment. The fourth part illustrates the method with a railway case study. The fifth one presents some discussions around the informal specification and our methodology efficiency. Then the last part gives some conclusions and prospects.

*Address correspondence to this author at the INRETS/ESTAS, 20 rue Élisée Reclus BP 317 59666 Villeneuve d'Ascq CEDEX, France; Tel: + 33 3 20 43 84 07; E-mail: françois.defossez@inrets.fr

1874-4478/11

2. APPROACH

In safety critical systems processing, requirement engineering is an important phase. We are convinced that graphical tools must be used to model them. In order to assess the requirement fulfillment, we also need mathematical tools. Nevertheless, when the assessment cannot be proved mathematically, a professional specialist is needed. Actually, from an ergonomic point of view, a graphical tool provides a better understanding of models.

The work described in this paper aims at assisting specification assessment through a rigorous requirement modelling. It starts when all the high level requirements have been elicited from the informal specifications, with some well-defined and identified safety critical entities. Then, a p-time Petri net is built in order to model safety temporal requirements. This model captures the set of the valid temporal behaviours.

The next step consists in building a process model by mean of a t-time Petri net. This model corresponds to a proposed solution which still has to be checked regarding requirements. In order to guaranty requirements traceability, some rules must be fulfilled for this model construction in order to allow consistency checking with the requirements model. Consistency checking is an important problem but not developed in this paper.

3. METHODOLOGY

Petri nets formalism is a well-known graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems. Fundamental properties and theoretical definitions are presented in [6].

In order to integrate the time window constraints in the Petri net model, dedicated tools named time Petri nets have been introduced. These tools have both modelling abilities and strong structural properties and therefore seem able to provide a representation of the system adapted to our goal. First we describe the two time Petri nets that are used in this paper. Then a comparison of these two models will be done in order to justify our modelling choices. Finally the state classes are described. In this section, all the steps of the methodology are illustrated by a theoretical example.

3.1. Relevant Tools for Particular Problems

Temporal requirement can be modelled by time interval associated to nodes of a discrete event system model. Nevertheless several modelling tools contain this kind of constraints: timed automata, t-time Petri nets and p-time Petri net etc. Let us recall those requirements are extracted from an informal specification where the functioning is, most of the time, only partially specified. Then, the modelling tools are divided between two classes of model: first ones have associated functioning and others have not. Actually, the introduction of a local functioning in a requirement model is not appropriate. Indeed, in this case some available solutions are rejected with no reason. Timed automata [7] and t-time Petri net [8, 9] have their own embedded functioning. Actually, they are well known efficient tool for modelling, controlling, simulating a system which already exists. More particularly, t-time Petri net are a widespread tool for temporal protocol modelling [10]. Thus, considering a system which is completely defined, this last tool is particularly relevant.

P-time Petri nets belong to another class of modelling tool. They have no default temporal functioning. When a duration belongs to a time interval, no assumption is locally made on the effective value of this duration. Considering this particularity, p-time Petri net are a good tool for high level requirement modelling [11]. As an illustration, performance evaluation can be performed taking only into account requirements. The immediate drawback of the previous singularity is that there is no dedicated efficient temporal simulation tool. Nevertheless the -time Petri net models all the available behaviors.

Finally, we propose p-time Petri nets for requirement modelling and t-time Petri nets to model a completely defined system. Actually, two different tools are used to model two different kinds of problem, whereas they are really closed from a semantic point of view. This last aspect is important for the checking step between the requirement and candidate process models.

3.2. P-Time Petri Nets for Requirements

The formal definition of a p-time Petri net [12] is given by a pair (R, I) where:

- R is a marked Petri net,
- $I : P \rightarrow Q^+ \times (Q \cup +\infty)$ where $p_i \rightarrow I_i = [a_i, b_i]$ with $0 \leq a_i \leq b_i$

The interval I_i defines the static interval of staying time of a mark in the place p_i belonging to the set of places P. When $b_i = +\infty$, it means there is no upper bound specification for the associated place p_i. A mark in the place p_i is taken into account in transition validation when it has stayed in p_i for a duration of at least a_i and no more than b_i. After the duration b_i, the token will be "dead". This last aspect is used for temporal requirement specification. In fact, the death of a token represents a non-respect of the specifications: it corresponds to a class of forbidden states. Contrary to the t-time extension, the evolution of a p-time Petri net is characterized by both the firing of transitions and the death of tokens. This tool has been introduced to model the behavior of production systems submitted to strong synchronization constraints. For example, in an industrial context of a production line where some products have to be soaked in chemical baths, the death of a token represents a product that stayed in a tank for too long [13]. One of its limitations is that p-time Petri net cannot easily model time-outs. This model is also relevant in a context of temporal safety requirements modelling.

In order to illustrate the different steps of our methodology, the following theoretical example is given:

- Let us consider a production process with a transport phase,
- There is a deadline of consumption date, named *max*, for the product.

Fig. (**1**) gives the p-time Petri net for this specification. Transition *production* (resp. *consumption*) represents the production (resp. the consumption) phase of the process.

Place p_3 represents the deadline of consumption constraint, p_2 the transport duration (at least d time units) and p_1 allows to reinitialize the process.

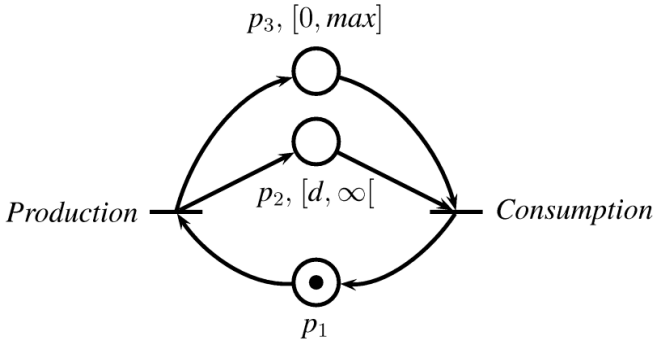

Fig. (1). Requirements modelling illustration.

3.3. T-Time Petri Nets for Process Modelling

The formal definition of a t-time Petri net [19] is given by a pair (R, I) where:

- R is a marked Petri net
- $I : T \rightarrow Q^+ \times (Q \cup +\infty)$ where $t_i \rightarrow I_i = [a_i, b_i]$ with $0 \leq a_i \leq b_i$

The static interval function I associates with each transition t_i a temporal interval $[a_i, b_i]$ that represents the set of its possible firing dates counting from its enabling date (Q^+ is the set of positive rational numbers). In a t-time Petri net, the events to consider are the enabling date of a transition, the beginning and the end of the temporal interval associated to the considered transition and finally the effective firing date of the transition. The belonging of the firing date to the interval can be expressed with simple temporal constraints. Thus, the notions of enabling and validating transitions are not still equivalent: a transition can be fired only if both conditions of marking and time are verified. This tool has been introduced to model and analyze communication systems, particularly to describe and validate telecommunication protocols. Moreover, it can be generalized to model procedures to be followed. Indeed, its ability to model uncertainties, by means of the intervals, appears as very useful in such a case. Finally, one of the last advantages of this tool is to be able to model watchdogs and time-outs. A time-out is a system able to verify if a given event happens before a given time, otherwise it indicates an error.

In order to check that a process model corresponds to a requirements model, a projection Pr is defined as:

- Pr is a relation,
- Sol is the set of process model nodes,
- Req is the set of requirements model nodes,
- Pr (Sol) = Req.

It means that all the nodes of the requirements model have to possess a corresponding node in the process model. In order to respect that, a guideline to build the process model is recommended. Fig. (**2**) gives the t-time Petri net which models a process answering to the specifications given by Fig. (**1**).

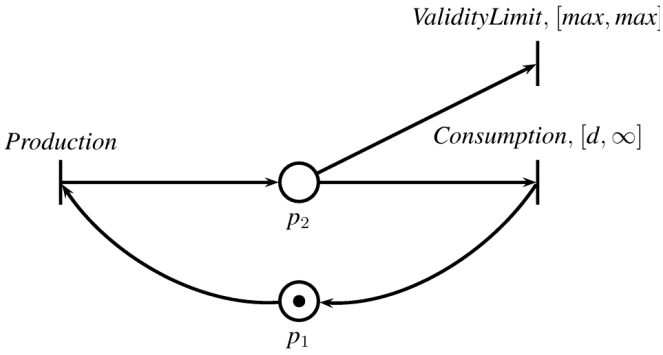

Fig. (2). Process modelling illustration.

We can notice that transitions (resp. places) keep the same name except the place modelling the deadline of consumption constraint (Table **1**). It is transformed into a transition, fired only if the limit max is reached. In such a case, the specification is no more fulfilled.

Let us note that several nodes in the process model can be projected on a single node on the requirement model. It can particularly happen in the case of a watchdog, built with several transitions and places in the process model, which corresponds to a temporal constraint modelled by a timed place in the requirement model.

Table 1. Corresponding Nodes Between the Two Models

Nodes Fig. (1)	Nodes Fig. (2)
Production	***Production***
Consumption	***Consumption***
P_1	P_1
P_2	P_2
P_3	Tp_3

3.4. State Classes

The analysis of Petri net can be done thanks to two methods: the structural analysis and the enumerative one [20]. The latter is used in order to check the dynamic properties of the system. This approach is based on the construction of the coverability tree. The checking that forbidden states are not reachable or the analysis of temporal constraints between events of a given scenario require the exhaustive search of all the states of the system. However, in case of temporal constraints, the states of the system are in infinite number, which makes impossible their direct exploration by enumeration because of the state-space explosion. Therefore, it is necessary to cover these states by a finite number of classes which are abstractions for symbolic states. As a result a class graph can be built in order to explain the states and their associated temporal constraints, which allow the transition from a class to another. In fact, the notion of state classes allows an analysis similar to the coverability tree method for the time models. Overall, the final purpose of this work is to check if the projection of the set of the state classes generated from the t-time Petri net on the requirement space is included in the set

of state classes generated from the p-time Petri net. We can express this purpose in a formal way. Let us denote:

- *Sr*: set of state classes generated from the p-time net modelling the requirements,
- *Ss*: set of state classes generated from the t-time net modelling the proposed solution as process,
- *Req*: the requirement space, i.e. the set of requirements model nodes,
- *R(A,B)*: an application which represents the projection of *A* on *B*.

As a result, we want to check if:

- R(Ss,Req) ⊂ Sr.

3.4.1. State Classes Construction

In a p-time Petri net, as a classic Petri net, a state can be reached from the initial state by a firing sequence, named *s*. For each transition of this sequence, its firing instant, named *u*, is associated. Let us consider the set of the reachable states from the initial one by firing of all feasible *u* corresponding to *s*: the set of all reachable state from the initial one by firing *s* is now define. This set defines the state class associated to *s*. The initial class, C_0, contains only one state, the initial one. Then, a state class is a pair *C= (M, D)* where:

- *M* is the class marking,
- *D* is the potential temporal firing domain.

Let us assume that the transition t_i is fireable at the instant θ_{t_i} from the class *C= (M, D)*. This fire generates the following class *C'= (M', D')* which is computed as follow:

1. *M'* is computed using usual incidence matrix,
2. *D'* is computed from *D*.

T he firing domains *D* of p-time Petri net state classes can be defined as a solution of these inequations:

1. $q b_i^j \geq \theta_i^j \geq a_i^j, \forall p \in P \wedge j \in J,$

   where *P* is the set of place and *J*, the set of marking.

2. $\theta_l^p - \theta_k^q \leq c_{pq}, \forall p_l, p_k \in P \wedge \forall p, q \in J$,

   where $c_{pq} = \max(\theta_l^p - \theta_k^q), \forall \theta \in D.$

The computation of *D'* from *D* is not detailed but just illustrated by Table **2**. The formal algorithm can be found in [14]. Fig. (**3**) gives the graph of state classes ensuing from p-time Petri net represented in Fig. (**1**).

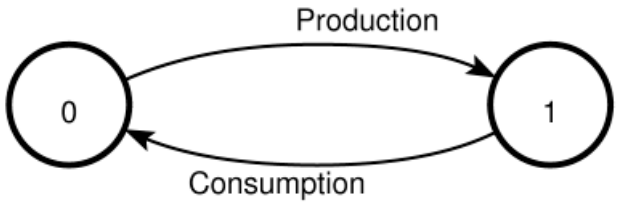

Fig. (3). Graph of the state classes of the p-time Petri net theoretical example.

Table 2. Markings Corresponding to the State Classes of Fig. (1).

Class	Marking	Domain
C_0	1	$\theta_{production} \in [0, \infty[$
C_1	2, 3	$\theta_{consumption} \in [d, max[$

The all description of t-time Petri net state classes construction is not defined here. [15] defines the theoretical definition of this construction and the TINA[1] (TIme petri Net Analyser) software tool is used in order to generate the t-time state classes [8]. Fig. (**4**) gives the graph of state classes ensuing from t-time Petri net represented in Fig. (**2**). Its marking is illustrated by Table **3**.

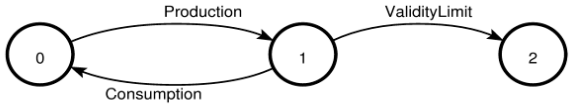

Fig. (4). Graph of the state classes of the t-time Petri net theoretical example.

Table 3. Markings Corresponding to the State Classes of Fig. (2)

Class	Marking	Domain
C0	1	$\theta_{production} \in [0, \infty[$
C1	2	$\theta_{consumption} \in [d, max[$ $\theta_{ValidityLimit} \in [max, max[$
C2	ø	ø

3.4.2. Projection and Consistency Checking

As shown in paragraph III.C, structural consistency is applied by construction. The behavioral consistency has to be now checked using the corresponding state classes graphs. Let us recall to mind that the evolution from a state class to another ensues from a transition firing. Let us denote by *Pe(Ss,Req)*, the subset of *Ss*, with *Ss* the set of classes generated from the t-time net modelling, the set of state where all selected state classes are:

1. Input state classes of a transition *t* firing with $t \in Req$, or
2. Output state classes of a transition firing of *Req*.

We can claim the consistency when:

$Pe(Ss, Req) \subset Sr$

Obviously, the illustrating example respect the conditions mentioned below. The consistency of the two models (Figs. **1**, **2**) can be checked.

4. CASE STUDY

4.1. Level Crossing Case Study

To illustrate the approach, a radio-based level crossing control system is designed. This example is inspired by [16]

[1] http://www.laas.fr/tina/

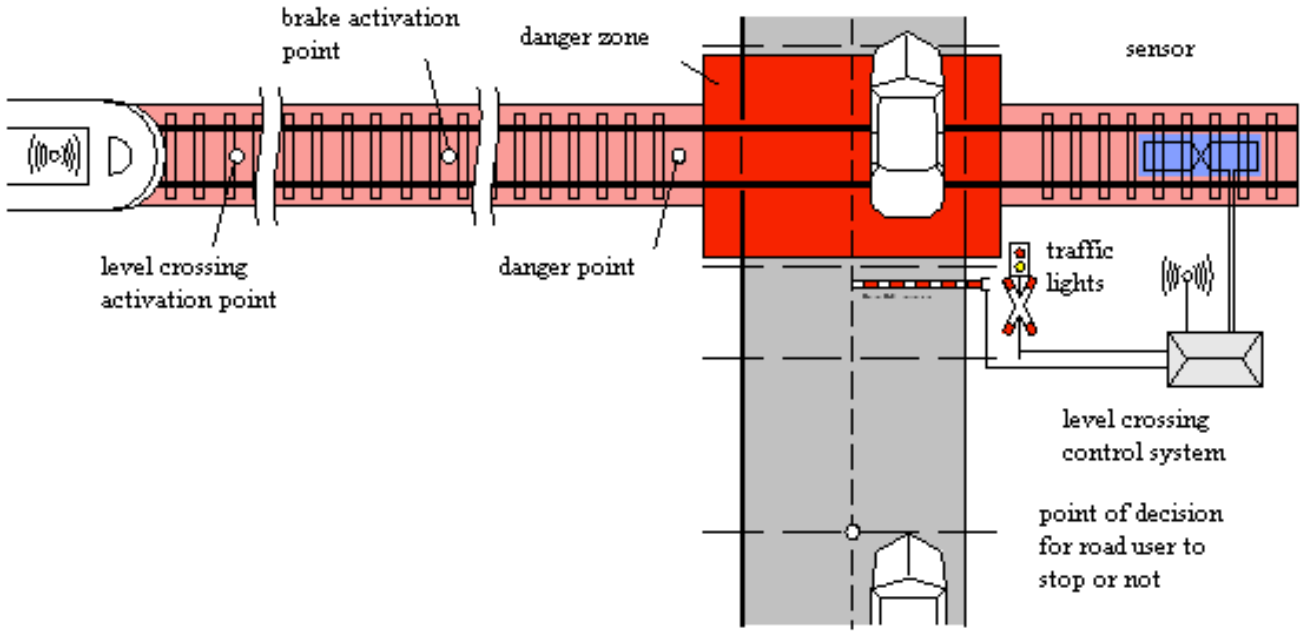

Fig. (5). Level crossing case study.

and is composed of a single-railway track which crosses a road at the same level (Fig. **5**). Our purpose is not to discuss about the specifications relevancy, but to propose a methodology which helps to assess that the process respects the specifications.

This theoretical case study is intended to scientific research and its specifications have both advantages of being more realistic and sizeable than theoretical traditional examples and less complex than industrial projects. For example, it has been chosen as an application case study in [17]. The crossing zone is named danger zone. The most important security rule is to avoid collision by prohibiting road and railway traffic simultaneously on level crossing. The railway crossing is equipped with barriers and road traffic lights to forbid the car passage. When they are switched off, road users (drivers, pedestrians,) can cross. In the other case, the level crossing is closed and railway traffic has priority. Half barriers are used in order to permit the evacuation of vehicles from the danger zone after the level crossing closing.

The main difference between this technology and the traditional control of level crossings is that signals and sensors on the route are replaced by radio communication systems embedded in the train and in the level crossing. This offers cheaper and more flexible solutions, but also shifts safety critical functionality from hardware to software. Instead of being detecting by a sensor, the train computes the position where it has to send a signal to secure the level crossing. Therefore the train has to know the position of the level crossing, the time needed to secure the level crossing, and its current speed and position, which are measured by an odometer. When the level crossing receives this command, it switches on the traffic lights, first the yellow lights, then the red lights, and finally closes the barriers. When they are closed, the level crossing is considered as safe for a given period of time. The stop signal, indicating an insecure crossing, is also substituted by computation and communication. The train requests the status of the level crossing. Depending on the answer the train brakes or passes the crossing. The level crossing performs self-diagnosis and automatically informs the central radio office about defects and problems.

4.2. Temporal Requirements

Since there are no barriers for the exit lanes, road users possibly may enter the crossing area on the opposite lane. Although this behavior constitutes a severe contravention of the traffic regulations, it can be frequently observed due to long waiting times at closed level crossings. This has to be taken into account for the level crossing control by respecting a maximum closure time.

In order to avoid long waiting times for road users before a closed level crossing, the case study specifications lay down that the train has to pass the level crossing before a maximum arrival time of 240 seconds, from having sent the activation order to the level crossing. If the train detects that it cannot arrive at the level crossing within the specified time and is still able to stop before the danger point, it has to cancel the activation order by sending a deactivation order to the level crossing. In this case, the train applies a braking curve ending at the danger point. The level crossing will be open upon receipt of the deactivation order. The passing of the unclosed level crossing requires the driver to confirm the safe state of the level crossing.

In order to take into account such temporal constraints, we need a formal modelling tool able to provide a representation of the level crossing adapted to our purpose. Consequently, we have chosen time Petri nets to model the system.

4.3. The Requirements Model

This section illustrates, on the level crossing case study, the first step of the proposed methodology. It contains a formal description of the requirements model using p-time

Petri nets. Then the state classes of this model are built in order to extract all the reachable states of the model.

4.3.1. P-Time Model

We focus on the needs of the system in building a requirements model that contains every constraint that must be fulfilled. This model, built with a p-time Petri net, takes particularly into account the expression of temporal constraints taken from the specifications.

This requirements model based on p-time Petri nets has already been presented in [11] and [18]. These papers aim at building safe control specifications. Namely, they propose some methods to avoid forbidden states associated with non respects of the temporal requirements. In this section, we use the same requirements model, which can be found in Fig. (**6**).

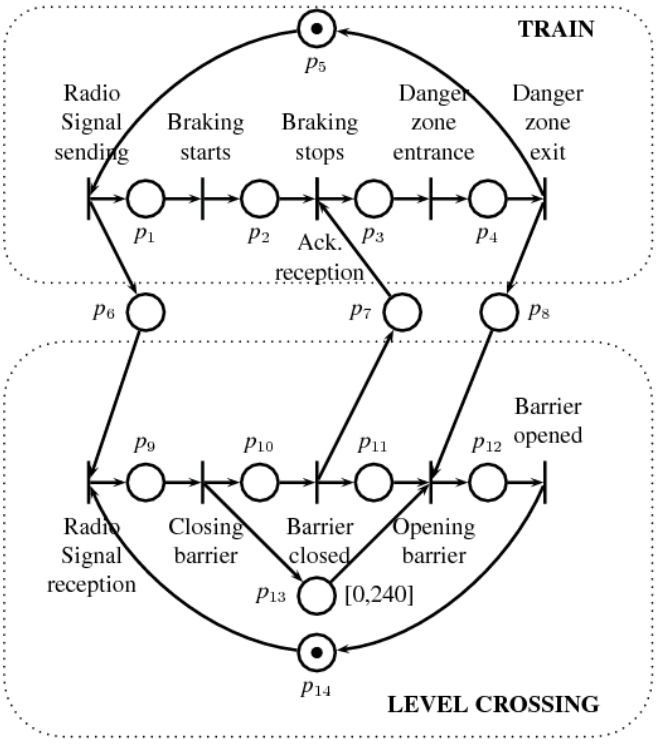

Fig. (6). Requirements model using p-time Petri net.

To sum up, there are two sequences synchronized by event occurrences. Let us introduce the time constraint on the place p_{13}: it models the limitation for the level crossing to be closed. A process must be proposed in order to avoid that a token becomes "dead" after having stayed more than 240 seconds in p_{13}. Indeed, this corresponds to a non respect of the temporal specifications. Such a process can be found in the case study.

Our goal is to build the state classes of this model in order to compare them with the projection of the state classes of the process model on the requirement space.

4.3.2. State Classes of the p-Time Model

The chosen approach consists in obtaining a finite representation of the reachable states by the construction of a coverability graph [14].

A state is reachable from the initial state by the execution of a firing sequence (s). For each transition of this sequence, a firing time (u) is associated. So, a class state associated to the sequence (s) is a set of reachable states from the initial state by the firing of the sequence (s). Therefore it is a couple $C= (M, D)$ with:

- M: the marking of the class,
- D: the potential firing domain of the class.

Fig. (**7**) represents the graph of classes of the p-time Petri net described by Fig. (**6**). It is composed by classes C_i and the arcs connecting them. An arc (C_i, C_j) represents the firing of the transition leading from C_i to C_j. For example, the arc connecting C_0 to C_1 corresponds to the firing of the Radio Signal Sending transition.

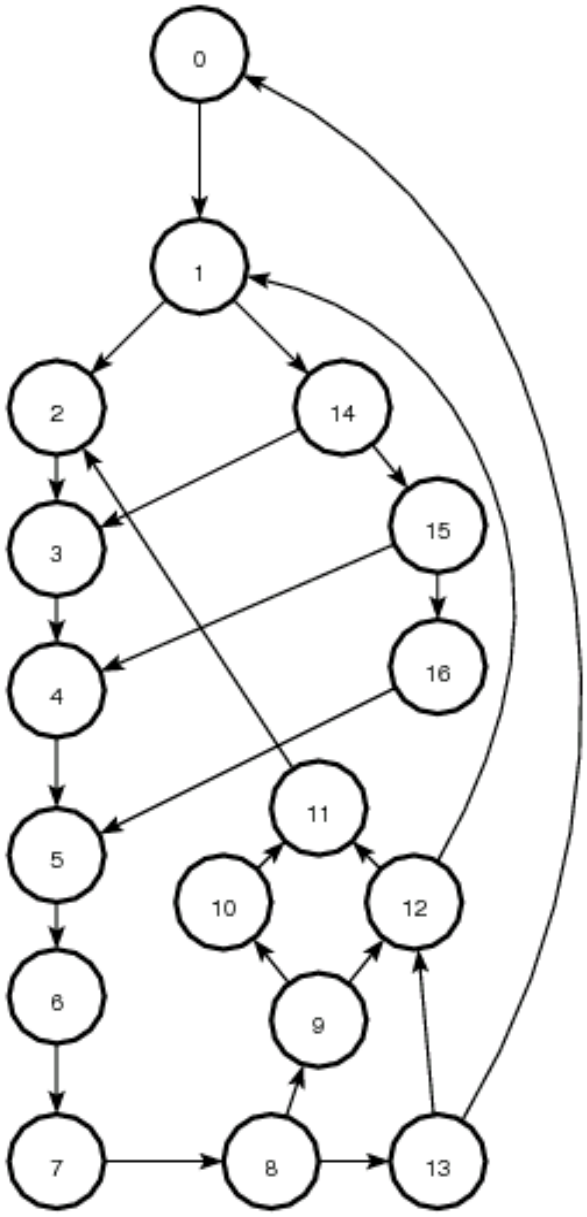

Fig. (7). Graph of the state classes of the p-time Petri net.

In Table **4**, the potential firing domains are represented. Actually, a domain is associated to each mark of a given class. For example, the temporal constraints for C_1 are:

- $0 \leq \theta_1^1 \leq 249$

Table 4. Markings Corresponding to the State Classes of the p-Time Model

Class	Marking	Domain	Class	Marking	Domain
C0	5,14	$\theta_5, \theta_{14} \in [0, \infty]$	C9	1,6,8,11,13	$\theta_1 \in [0,249]$ $\theta_6, \theta_8 \in [0,0]$ $\theta_{11}, \theta_{13} \in [0,240]$
C1	1,6,14	$\theta_1 \in [0,249]$ $\theta_6 \in [0,0]$ $\theta_{14} \in [0, \infty]$	C10	2,6,8,11,13	$\theta_2 \in [0,249]$ $\theta_6, \theta_8 \in [0,0]$ $\theta_{11}, \theta_{13} \in [0,240]$
C2	2,6,14	$\theta_2 \in [0,249]$ $\theta_6 \in [0,0]$ $\theta_{14} \in [0, \infty]$	C11	2,6,8	$\theta_2 \in [0,249]$ $\theta_6 \in [0,0]$ $\theta_{12} \in [0,240]$
C3	2,9	$\theta_2 \in [0,249]$ $\theta_9 \in [9,9]$	C12	1,6,12	$\theta_1 \in [0,249]$ $\theta_6 \in [0,0]$ $\theta_{12} \in [0, \infty]$
C4	2,10,13	$\theta_2 \in [0,249]$ $\theta_{10}, \theta_{13} \in [0,240]$	C13	5,12	$\theta_5, \theta_{12} \in [0, \infty]$
C5	2,7,11,13	$\theta_2 \in [0,249]$ $\theta_7 \in [0,0]$ $\theta_{11}, \theta_{13} \in [0,204]$	C14	1,9	$\theta_1 \in [0,249]$ $\theta_9 \in [9,9]$
C6	3,11,13	$\theta_3, \theta_{11}, \theta_{13} \in [0,240]$	C15	1,10,13	$\theta_1 \in [0,249]$ $\theta_{10}, \theta_{13} \in [0,240]$
C7	4,11,13	$\theta_4, \theta_{11}, \theta_{13} \in [0,204]$	C16	1,7,11,13	$\theta_1 \in [0, \infty 249$ $\theta_7 \in [0,0]$ $\theta_{11}, \theta_{13} \in [0,240]$
C8	5,8,11,13	$\theta_5 \in [0, \infty]$ $\theta_8 \in [0,0]$ $\theta_{11}, \theta_{13} \in [0,240]$			

- $0 \leq \theta_6^1 \leq 0$
- $0 \leq \theta_{14}^1 \leq \infty$

The upper bound of θ_1^1 is due to temporal requirements for lights (9 s) and barriers (240 s). This is an example of temporal requirements expression through a couple of parallelism and synchronization transition [18, 19]. The description of the 17 classes of the graph of Fig. (**7**) can be found in Table **4**. The notation in Table **4** is simplified, as there is only one token per place, the exponent is not specified.

We built the graph of classes of the requirements model. Likewise, the next step is to build the graph of classes of the proposed process model in order to compare both of them and thus to check if the process fulfils the requirements model.

4.4. The Process Model

The second step of the proposed methodology is now presented. Its purpose is to describe a process identified as a solution of the system functioning. This process model corresponds to a solution described in [16]. It is modelled by means of t-time Petri net. The method consists in exploring all the possible behaviors of the system by means of the construction of state classes [20].

4.4.1. T-Time Model

As shown in the part III.C, a static interval function I associates, with each transition t_i, a temporal interval $[a_i, b_i]$

that represents the set of its possible firing dates counting from its enabling date.

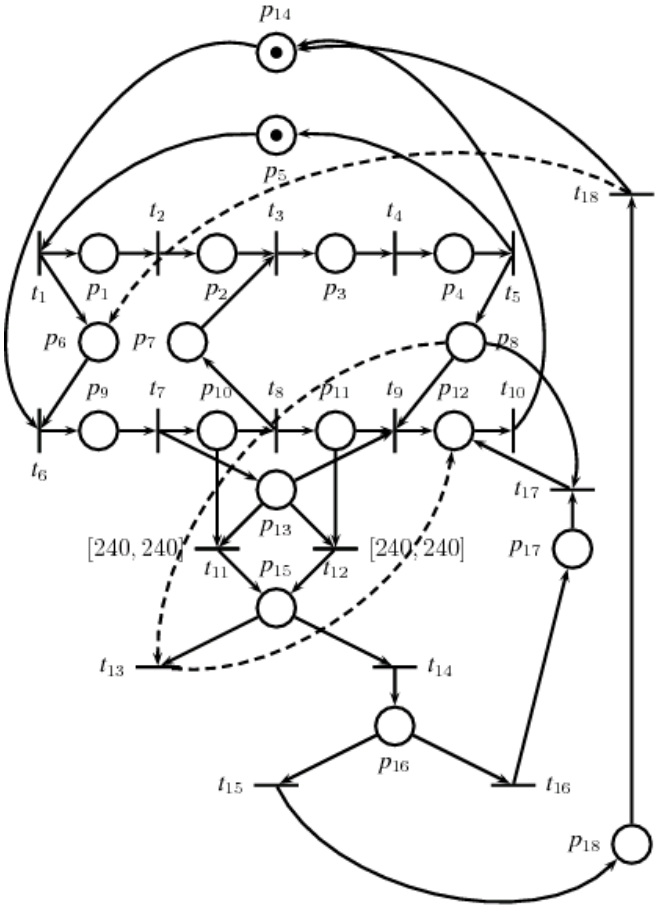

Fig. (8). Process model using t-time Petri net.

Fig. (**8**) models a given solution to fulfill the requirements that have been described in section IV.B.

The two sequences representing the behavior of the train and the level crossing can be found on this model. From p_1 to p_4, this model is similar to the model of Fig. (**6**), except that the place p_{13}, which was a timed one on the p-time model, is replaced with a watchdog in this model. Indeed, this t-time model deals with a scenario that is proposed to fulfill the requirements. Table **5** illustrates the relation.

If a token stays more than 240 seconds in p_{10} (respectively p_{11}), the transition t_{11} (resp t_{12}) is fired and the operating procedure aiming at avoiding the opening of the barriers when a train is in the level crossing is engaged. This procedure can be considered as an answer to the death of a token in the timed place p_{13} in the p-time model. So the next paragraph describes the behavior of the system in such a case.

First there are two possibilities of evolution: either the train which has sent a radio signal to the level crossing is still in the crossing area (that means that a token is in one of the places p_1 to p_4) or it has gone past it (that means that there is a token in p_8).

In the first case (that means that t_{12} is fired), the operating centre has to decide if the train is stoppable or not. If it is stoppable, the train stops before the danger point and the procedure of radio signal sending is reinitialized (t_{15} is fired).

Otherwise, the level crossing control system has to wait that the train has passed on the exit sensor (t_{16} is fired).

In the second case, the operations centre orders the opening of the barriers (t_{11} is fired).

Table 5. Corresponding Nodes Between the Two Models

Nodes of Fig. (6)	Nodes of Fig. (8)
$p_i, i \in [1,12]$	$p_i, i \in [1,12]$
p_{13}	$p_i, i \in 13 \cup [15,18]$
p_{14}	p_{14}

The nodes corresponding to the place p_{13} in the process model are out of the specifications because they represent a recovery mode not demanded in the specification.

4.4.2. State Classes of the t-Time Model

In [15], the building of the state classes for the t-time model can be found. Fig. (**9**) represents the graph of classes of the p-time Petri net described by Fig. (**8**) and in order to keep figure readable, transitions don't label arc as in Fig. (**7**).

Table **6** doesn't list the potential firing domains in order to keep table readable. The building of these domains can be found in the literature. Moreover, a tool called TINA can be used to build the state classes for the t-time model. For example, the class C_0 has the following temporal constraints:

- $0 \leq \theta_5^1 \leq \infty$
- $0 \leq \theta_{14}^1 \leq \infty$

And for:

- $0 \leq \theta_1^1 \leq \infty$
- $0 \leq \theta_6^1 \leq \infty$

The underlined classes of this model correspond to those which are directly included on the projection of the classes of the p-time one. They correspond to a recovery process if the temporal requirement of 240 seconds is not respected.

We notice that the state classes dealings with the watchdog don't correspond to the requirements. The projection excludes all the classes that can be only reached by t_{11} or t_{12}, which are transitions of failure mode management.

5. DISCUSSION

Fig. (**8**) describes a given solution proposed by [16] in order to fulfill its requirements model. The underlined classes in Table **6** correspond to the requirements model. For these classes, we have the same marking than in Table **4** but not the same temporal domains. Indeed, the proposed solution gives the control of temporal requirements to the train controller. Therefore, in normal mode, the proposed solution fulfils the specifications.

Furthermore, the given solution proposes to manage the recovery process, even if it is outside the informal specifications. The requirements model doesn't precise what the specifications become in a failure mode. Thereby, our

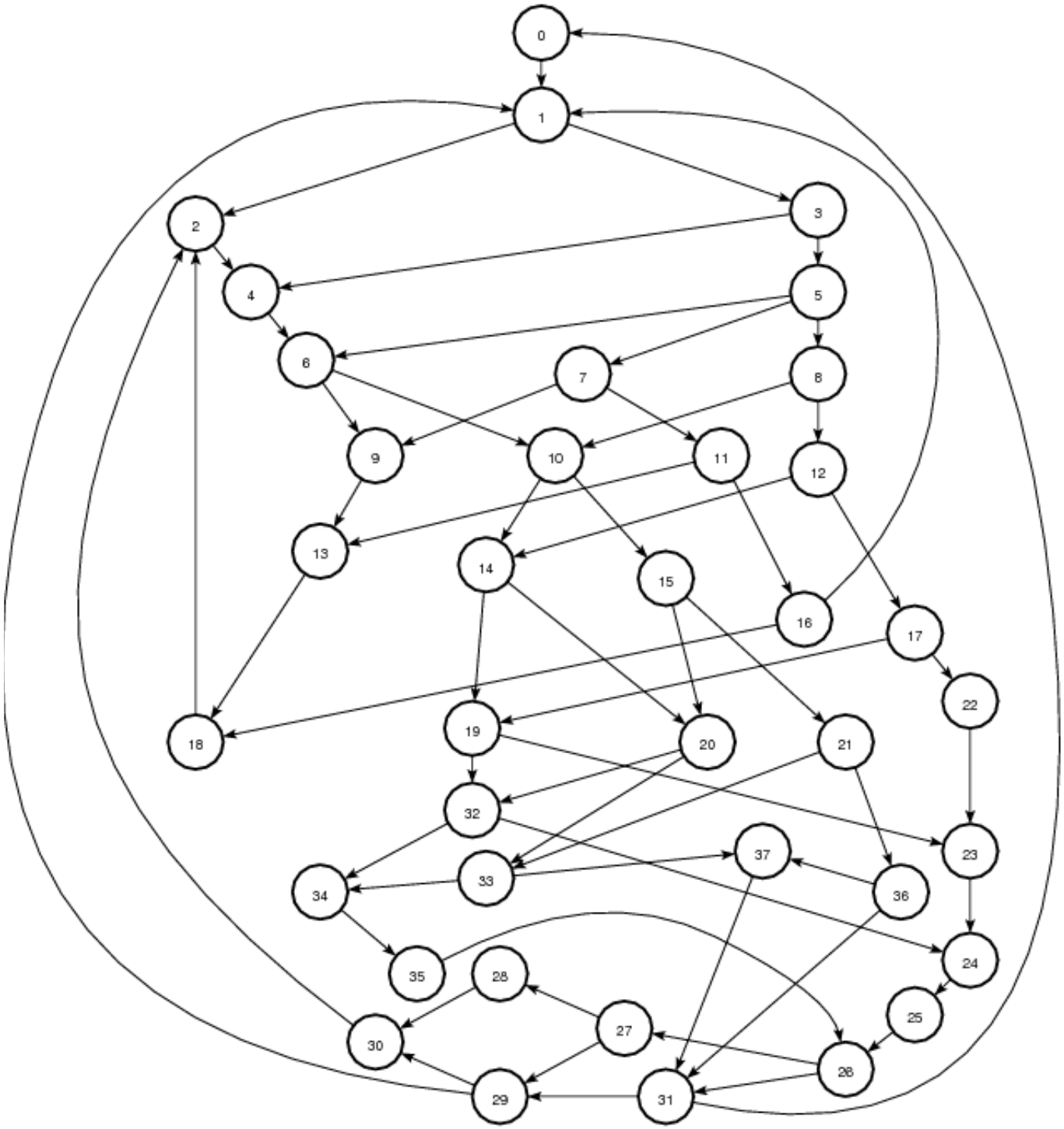

Fig. (9). Graph of the state classes of the t-time Petri net.

work extracted an explicit temporal requirement: the maximum closing time of 240 seconds for the level crossing. Nevertheless, we noticed that this creates a conflict with an implicit but essential requirement: it is strictly forbidden to open the level-crossing barrier if a train is in the danger zone. We aren't able to certify unwritten implicit rules. In such a case, an expertise is necessary to clarify the specifications.

Actually, the automatic assessment of requirements depends on what is effectively written in the informal specifications: when there are implicit (non written) requirements, a professional specialist is needed in order to express them. Moreover, extracting requirements from the informal specification is only one step of the process of solution construction. In fact, explicit identification of critical entities which are directly relevant to high level requirements is also needed at each step. To assess a solution, the traceability chain has to be not broken. Otherwise, the assessor is not able to certify that the solution

Table 6. Markings Corresponding to the State Classes of the t-Time Model

Class	Marking	Class	Marking	Class	Marking	Class	Marking
C0	5,14	C10	2,7,11,13	C20	3, 15	C30	2, 6, 12
C1	1, 6,14	C11	1, 16	C21	4, 11, 13	C31	5, 12
C2	2, 6,14	C12	1, 7, 15	C22	1, 7, 17	C32	3, 16
C3	1, 9	C13	2, 16	C23	2, 7, 17	C33	4, 15
C4	2, 9	C14	2, 7, 15	C24	3, 17	C34	4, 16
C5	1, 10, 13	C15	3, 11, 15	C25	4, 17	C35	5, 8, 16
C6	2, 10, 13	C16	1, 18	C26	5, 8, 17	C36	5,8,11,13
C7	1, 15	C17	1, 7, 16	C27	1,6, 8,17	C37	5, 8, 15
C8	1, 7, 11, 13	C18	2, 18	C28	2,6,8,17		
C9	2, 15	C19	2, 7, 16	C29	1, 6, 12		

model fulfils requirements without involving an independent professional specialist.

Another interesting aspect is that our methodology can naturally be connected with a global conception process based upon graphical tool. For instance, using UML notation, the functional analysis may be performed with use case diagrams, structural analysis with class diagrams, behavior analysis with state diagrams.

In the case of UML modeling, the state diagrams are particularly important and are the connecting points with our methodology. Anyway, whatever modelling tool, the different system states and transitions which are safety critical have to be identified and well defined in order to be taken into account in the next steps.

CONCLUSIONS AND FUTURE WORK

In this paper, a methodological approach based upon time Petri nets, in order to manage temporal requirements, was presented. To this end, the requirements model was built from the informal specification by means of p-time Petri nets. Then, the state classes of the model have to be built in order to capture the behavioural validity domain. In order to ensure traceability, the critical nodes have to be highlighted and explicitly transmitted from the requirements model to the process model. The process model is then made by means of t-time Petri nets. The next step consists in building state classes graph in order to check that the process model behavior respects the requirements. A correct implementat-ion of the transmission of critical nodes has allowed an automatic consistency check.

Our methodology was illustrated on a level crossing case study. This example is particularly relevant, because it provides an illustration for the two different scenarii of the solution construction process. The high level requirements are totally fulfilled by our proposed solution in normal mode, but an implicit requirement was highlighted in failure mode. Our methodology assesses the consistency between the requirements model and the solution model in normal mode but is unable to assess failure mode without the help of a professional specialist.

In order to integrate this methodology in the global issue of safety requirements, we propose an approach based on abstract high level Petri nets, of which p and t-time Petri nets are a subclass. In future works, we suggest that the use of a process allowing the complementary strengths of the *B* formal method [3] and the Petri net model to be used together in order to promise increased reliability of a railway traffic control application [21]. The aim is to provide some sharp results concerning time parameters, requirements traceability and formal validation, and a global approach including all functional aspects during all the conception process.

REFERENCES

[1] Railway applications. “Communications, Signalling and Processing Systems - Software for railway control and protection systems”, 2001.

[2] Unified modelling language version1.4. Technical report, OMG, 2001.

[3] J.R. Abrial,. The B Book - Assigning Programs to Meanings. Cambridge University Press, August 1996.

[4] P. Behm, P. Benoit, A. Faivre, and J.M. Meynadier. METEOR : A successful application of B in a large project. In J. M. Wing, J. Woodcock, and J. Davies, Eds, FM’99 - Formal Methods, number 1709, pages 369-387. Springer Verlag, September 1999.

[5] C.A. Petri, Communicating with Automata. Thèse de doctorat, Université de Darmstadt, 1962.

[6] T. Murata, “Petri nets: Properties, analysis and applications”, Proceedings of the IEEE, vol. 77, no. 4, pp 541-557, 1989.

[7] R. Alur, and D. Dill, “A theory of timed automata”,. Theoretical Computer Science, vol. 126, pp.183-235, 1994.

[8] B. Berthomieu, P.-O Ribet, and F Vernadat, “The tina tool: Construction of abstract State space for petri nets and time petri nets,”, International Journal of Production Research, vol. 42, no. 14, pp. 2741-2756, 2004.

[9] B. Berthomieu, and F. Vernadat, “State class constructions for Branching Analysis of Time Petri Nets”, In H. Garavel and J. Hatcliff, Eds. TACAS, of Lecture Notes in Computer Science, Springer USA, vol. 2619, pp. 442-457, 2003

[10] P. Merlin, A Study of the Recoverability of Computer Systems. PhD thesis, University of California, Irvine, California, 1974.

[11] F. Defossez, P. Bon, and C.S. Dutilleul, “Formal Methods and Temporal Safety Requirements: a Level Crossing Application. In: E. Schnieder and G. Tarnai, Eds, Formal Methods and Automation and Safety in Railway and Automotive Systems, FORMS/ FORMAT’2007, Braunschweig, Germany, Jan PP. 219-230, 2007

[12] W. Khansa, P. Aygalinc, and J. P. Denat, Structural Analysis of P Time Petri Nets. In CESA ’96, Lille, France, 1996.

[13] N. Jerbi, S. Collart Dutilleul, E. Craye, and M. Benrejeb, Robust control of multi-product job-shops in repetitive functioning mode. In IEEE Conference on Systems, Man, and Cybernetics (SMC’04), The Hague, Netherlands, vol. 5, pages 4917-4922, October 2004.

[14] Khansa W, Réseaux de Petri p-temporels : contribution à l'étude des systèmes à événements discrets. thèse de doctorat, Université de Savoie, France, 1997.

[15] B. Berthomieu and M. Diaz, "Modeling and verification of time dependant systems using time petri nets", IEEE Transactions on Software Engineering, vol. 17, no. 3, pp. 259- 273, 1991.

[16] L. Jansen and E. Schnieder. "Traffic Control Systems Case Study: Problem Description and a Note on Domain-Based Software Specification". Technical report, Technical University of Braunschweig, 2000.

[17] J.-L. Boulanger and P. Bon, "BRAIL Requirement Analysis". In E. Schnieder and G. Tarnai, Eds, Formal Methods and Automation and Safety in Railway and Automotive Systems, FORMS/FOR MAT'2004, pages 221-229, Braunschweig, Germany, December, pp. 221-229 2004

[18] P. Bon S. Collart-Dutilleul, F. Defossez, Safety requirements and p-time Petri nets: a level crossing case study. In IMACS Multiconference on Computational Engineering in Systems Applications (CESA 2006), Pékin, Chine, October, pp. 1118-1123. 2006.

[19] W. Khansa, J. P. Denat, and S. Collart Dutilleul, "P-time Petri Nets for Manufacturing Systems". In IEE International Workshop On Discrete Event Systems (WODES'96), pp. 94-102, Edinburgh, Scotland, August 1996.

[20] F. Defossez, S. Collart Dutilleul, and P. Bon, Temporal requirements checking in a safety analysis of railway systems. In E. Schnieder, and G. Tarnai, Eds., Formal Methods and Automation and Safety in Railway and Automotive Systems, FORMS/FOR MAT, 2008, Braunschweig, Germany, October 2008, pp. 273-280.

[21] F. Defossez , P. Bon, and S. Collart Dutilleul, "Taking advantage of some complementary modeling methods to meet critical system requirement specifications". *Computer Railway*, vol. XI, PP. 153-162, 2008.

Received: May 25, 2010 Revised: August 23, 2010 Accepted: September 29, 2010