ESA_PetriNet: a Tool for Extracting Scenarios in Computer Controlled Systems

Malika Medjoudj

LAGIS, Ecole Centrale de Lille

Cité Scientifique, BP 48

Villeneuve d'Ascq, 59651, France

malika.medjoudj@ec-lille.fr

Abstract: This paper deals with the dynamic reliability of a computer-controlled system by means of deriving critical scenarios from its Petri net model. These scenarios characterize how the system leaves the normal operating to go to the feared state by determining the sequences of actions (events) and state changes leading to dangerous situation. We present a method (algorithm) that takes into account the continuous dynamic of the system by a temporal abstraction, which makes it possible to determine more precisely the exact conditions of the occurrence of the feared event. The originality is that the order of occurrence of the events is taken into account, and impossible scenarios with respect to the continuous dynamic of the system are eliminated. The automation of all the steps of this method has led to the development of ESA_PetriNet tool (Extraction Scenarios & Analyzer by Petri Net model) and was applied on real industrial systems.

Keywords: dynamic reliability, critical scenarios, computer-controlled systems, hybrid aspect, Petri nets, temporal abstraction.

Malika Medjoudj was born in Tizi-ouzou (Algeria) on February 21, 1977. She received the Engineer Diploma degree in Electronics (Control) and the Diploma of Higher Education Applied in Technical English from Mouloud Mammeri University of Tizi-ouzou (Algeria) in 2001. She obtained the Master in Industrial Systems from UPS-LAAS-CNRS of Toulouse (France) in 2002 and the PhD in Industrial Systems from the same university and laboratory in March 2006. She is actually a Post Doctorate at the Ecole Centrale de Lille after a scientific stay of six months in the nuclear metrology service of the Université Libre de Bruxelles (FNRS-Belgium). Her research is related to the reliability of hybrid and dynamic systems (computer-controlled systems, embedded systems), checking of temporal constraints, extended Petri Nets for safety (transportation systems), feared scenarios and simulation.

1. Introduction

Computer-controlled systems are energy systems (mechanical, hydraulic, electrical) ordered and controlled by one or several computers (computer science and electronics). These systems are used in the field of defense, space, nuclear (control of the nuclear power stations); car and avionic (embedded systems as mechatronic systems and computers flight, landing gear systems, ect). The software and material suppleness of these systems allowed a progressive integration of electronics in these named fields to improve both functions and services. However, this has caused an increased complexity in the design of these systems typically involving computers, which makes the control of their reliability difficult. In addition, the phase of design must be fast and inexpensive (i.e. less prototypes and at the later stages) with a level of guaranteed safety. In more cases for reasons of cost and implementation, material resources are limited and the system designers must avoid component redundancies within the system as much as possible. Reliability studies performed at the design phase have allowed a better control of the risks and reliability of the conceived systems. Indeed, the evaluation of the safety level during the systems conception allows the specification of piloting strategies and reconfiguration modes before the first tests on a real prototype. Computer-controlled systems are hybrid: continuous dynamics is applied to the power characteristics, and discrete dynamics is related to the numerical control and the existence of discrete events (failures and thresholds). The study of reliability of these hybrid and dynamic systems named dynamic reliability [1] [2] [3] or probabilistic dynamics [4] [5] must necessarily take into account the existing interactions between their physical parameters (temperature, pressure, speed, etc.) and the failure of their components.

One way to evaluate the reliability of such complex systems is the extraction of critical scenarios leading to feared states. From a qualitative point of view, this is a question of characterizing these scenarios as soon as possible in the design phase, which makes it possible to evaluate their probabilities of occurrence in order to validate the architecture of the system or to evaluate the safety level of existent systems.

Traditional methods for reliability are insufficient because they don't take into account the reconfiguration and the hybrid dynamic of the system. For example classical Failures Trees [6] are static and don't take into account the order of appearance of the events. In effect, a sequence of events can lead to a feared event while the same events occurring in a different order or in different dates can avoid it. The time separating two events is not taken into account in the Failures Trees method; therefore, reconfigurations cannot be represented. Temporary failures are not either taken into account. Several

extensions of classical methods were proposed to extend their field of application like Failure Trees with gates (A before B). These methods remain combinatory and unable to take into account the states changes and reconfigurations in the feared scenarios. Other methods were introduced as the Events Sequence Diagrams (ESD) [7] to allow a better visual presentation of the events ordered in time. Although the ESD represents in a clear way the scenarios in competition, they cannot be generated automatically and require a definition of states and transitions. All order and reconfiguration states must therefore be listed by the designer and in the case of the hybrid dynamic systems; the number of states is infinite if the energy party is taken into account. This problem is also encountered in the analytical methods based on Markov graphs. To take into account partially the dynamic of the system, methods of discretization were developed as the Discrete Dynamic Event Trees (DDET). The DDET generates the feared scenarios by failure propagation of the elementary components of the system. The limit of this method is that all the sequences of event constituting possible scenarios are generated. In order to better manage the multiple generated scenarios by the DDET, methods as DYLAM (Dynamic Logical Analytical Methodology) and DETAM (Dynamic Event Tree Analysis Method) were developed. Therefore, the time and order of execution of the events must be taken into account [8]. Limits of quantitative methods based on simulation [9] are owed to the combinative states explosion. Because of the scarcity of the feared scenarios, these methods simulate in the most part of time the normal operating. It is however necessary to mention the existence of theoretical developments and methods to resolve the problem encountered in the simulation of systems in the presence of rare events [10]. Indeed, techniques of acceleration of the simulation were developed and largely used with success, in particular in nuclear engineering. We can mention Monte Carlo Dynamic Event Tree (MCDET) [11] which is a coupling of the DDET with Monte Carlo Simulation [12] to investigate in a more efficient way the whole tree of events.

In the case of Petri nets [13], the combinative explosion affects the accessibility graph and not the original Petri net. So to avoid this combinative states explosion, a qualitative analysis method of reliability aiming to directly use the Petri net model of the system to extract the feared scenarios without generating the reachability graph was developed by [14]. Unfortunately this method based on Linear Logic [15] operated only on the discrete aspect of the system and lot of impossible scenarios is generated. To determine more precisely the exact conditions of the occurrence of the feared event, i.e what has led the system to leave its normal operation and to evolve into the feared state, a method taking into account the continuous aspect and the temporal specifications of the system is developed by [16] [17]. The originality of this approach, automated to result ESA_PetriNet tool [18], is that the order of occurrence of the events is taken into account, and impossible scenarios with respect to the continuous dynamic and the temporal specifications of the system are eliminated. ESA_PetriNet tool has been interfaced with TINA tool (Time Petri Net Analyzer) [19].

We will present the method and the basic of the algorithm in section 2, the ESA_PetriNet tool in section 3, the selected case study and the scenarios generation in section 4, and we will end by a conclusion.

2. Method of Extraction of Feared Scenarios

We call a scenario, a set of events (here transitions firing) leading from one partial state (here partial marking) to another one and verifying a partial order. We assume that the system is made up of a set of components. A partial state is the conjunction of the states of a subset of these components.

Definition 1**:** *A partial order is defined by a directed graph (E, A) where the nodes E are a set of transitions firing and the arcs A are pairs (ti, tj) such that ti precedes tj (ti and tj are transitions firing).*

The application of this method requires the modeling of the system by a Petri Net model and identifying the places of nominal behavior. The appropriate Petri net modeling of computer controlled systems is a Predicate Transitions Differential Stochastic Petri net (PTDS Petri net) [14] as they are generally hybrid (discrete and continuous dynamics) and there reliability analysis require taking into account the failures. This modeling approach that associates Petri nets and differential equations [20] has the advantage to clearly separate the continuous aspect from the discrete one; the Petri net model describes the operation modes, the failures and the reconfiguration mechanisms. The differential equations represent the evolution of the continuous variables of the energetic part of the system. A temporal abstraction is necessary to translate this model to a time Petri net by associating to the transitions a temporal interval of firing corresponding to the time which the system can spend to reach the state in question. A preliminary analysis will refine the fields of variables according to various accessible marking by reasoning on the invariants of places. Indeed, the invariants of places determine the possible dynamics, and which other places can be simultaneously marked when a token is present in a given place.

2.1 Principal

The method of extraction of feared scenarios is made up of two steps [16]: a backward reasoning and a forward reasoning. The backward reasoning takes as an initial marking in the reversed Petri net model (the initial Petri net in which all the arcs are reversed), the only target state (feared) and seeks exhaustively all the scenarios making it possible to consume the initial marking (feared state since forward reasoning) and reach a final marking composed only of places associated to the normal operation. The forward reasoning takes as an initial state these places of normal operation in the initial Petri net model. The objective is to locate the junctions between the feared behavior and the normal operation of the system as well as the conditions implied in these junctions. Thus we have not only the explanation of the dangerous behavior but also of strategies allowing its avoidance. A significant point of the method is that the context in which occurred the feared event is enriched gradually.

Definition 2*: let us consider a marked Petri net R. A potentially enabled transition is a transition that has at least a marked input place (contains a token at least) and at least a non marked input place (lacks a token at least).*

The process of enrichment of marking consists in adding the missing tokens to the potentially enabled transition to become enabled. This makes evolve the system to generate the scenarios. It is necessary to verify that all components composing the system are not in two different configurations with this enrichment what would be contradictory with the reality of the physical system. Indeed the new added tokens are removed if they are contradictory with the structure of the system (for example a component can not be in its activated and deactivated mode at the same time). The invariants of places are used as a mechanism to verify the coherence of the enrichment of marking. Each scenario is given in the form of a partial order between the events necessary to the appearance of the feared event what differs from a failures tree, which gives a whole of static combinations of the partial states necessary for obtaining the feared state.

2.2 Dealing with continuous dynamic by temporal abstraction

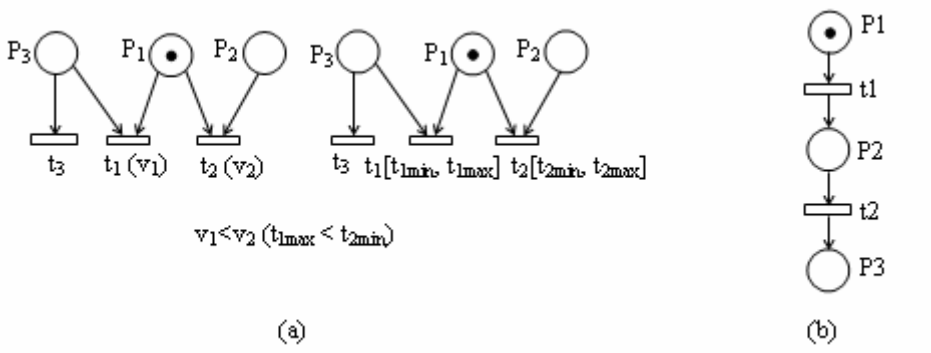


Figure 1. Temporal abstraction and direct and indirect causality

This method takes into account conditions associated to the firing of certain transitions. These conditions are thresholds involving continuous variables. By temporal approximation of the hybrid dynamic, these thresholds are transformed to durations, which correspond to the time that the system puts to reach when the transitions are enabled. From a qualitative point of view, the objective is to determine the firing order of the transitions. Thus, when we enrich the marking, we can find situation where two transitions *t1* and *t2* are enabled if only the ordinary Petri net is considered, but whose are such as *t1* will be always fired before *t2* if the temporal abstraction is also considered. In the generation of the scenarios only the firing of *t1* will be considered since that of *t2* before *t1* would be in fact incoherent with the continuous dynamic. This appears in the form of a priority of firing: if *t1* and *t2* are enabled, only the case of *t1*, priority, is examined. The taking into account of these precedence relations coming from the continuous dynamic and not specified by the ordinary Petri net allows reducing the number of generated scenarios by eliminating a certain number of incoherent scenarios with respect to the continuous dynamic.

Let us consider an example. In the Figure 1a we suppose that the differential-algebra system associated to the place *P1* guarantees that the variable *x* is increasing. We associate to the transition *t1* the threshold $x = v1$ and to the transition *t2* the threshold $x = v2$ with $v1 < v2$. Finally, we suppose that when the token

arrives in the place *P1* we have always $x < v1$. So, if the place *P3* is marked, the transition *t1* will be fired before *t2* since the threshold associated to *t1* is lower than that of *t2*. In this case we don't consider the scenario associated to the firing of *t2*. On the other hand, if *t3* is already fired for example if we consider that *t1* is a stochastic transition corresponding to a failure (place *P3* empty) and if the place *P2* is marked, *t1* cannot be fired and then *t2* will be fired.

2.3 Precedence and direct and indirect causality

In the example above, finally only one type of scenarios is examined, those for which the transition *t2* is fired after *t3*. So, there is a precedence relation between the firing of *t3*, which empties the place *P3* and that of *t2*, however there is no place connecting *t3* to *t2*. This precedence relation is so, a consequence of continuous dynamic and thresholds associated to transitions *t1* and *t2*. We are talking in this case about indirect precedence relation and about indirect causality. The direct precedence relation and causality are those that are highlighted by the only Petri net, i.e. by the only discrete aspect. For example place *P2* in the Figure 1b leads to a direct causality relation between the firing of t*1* and that of *t2*. It is necessary to have produced a token in the place *P2* by firing *t1* to be able to fire *t2*.

2.4 Case of priority between transitions firing

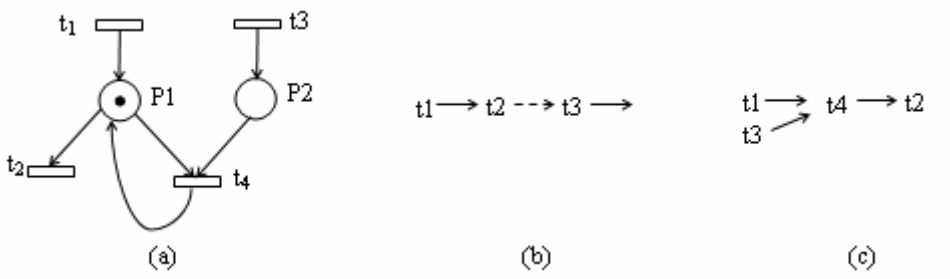


Figure 2. Case of direct and indirect causality

We have seen above that continuous dynamic could lead to precedence relations between the firing of transitions. A similar phenomenon can occur if we introduce rules of priority between the firing of transitions. Although we do not introduce rules of priority explicitly between firing transitions, they can be necessary to represent strategies of order and of reconfiguration in a simple way. Besides, because of what precedes, they will be taken into account in a simple way in our algorithm.

Let consider the Petri net of Figure 2a. It represents a temporal window (place *p1* contains a token between the firing of transition *t1* and *t2*) during which the treatment of *t4* can not only be performed, but must be performed without waiting if there is a request (presence of a token in the place *P2* resulted by the firing of the transition *t3*). In a classical way, it can be expressed by returning firing of *t4* priority in comparison with that of *t2*. Regarding precedence relations, only two types of scenarios will be generated. In the first scenario (Figure 2b), *t3* is fired after *t2*. In the second (Figure 2c), *t4* is fired before *t2*. The precedence relation and causality between *t2* and *t3* in the Figure 2b is an indirect relation as before because it does not correspond to a place linking up both transitions.

3. ESA_PetriNet Tool

ESA_PetriNet tool has been developed in JAVA to have a better portability (use on various material platforms and under various operating systems). The current version of ESA_PetriNet uses two output files of TINA tool version 2.8.4.

3.1 Algorithm

The temporal abstraction of the continuous dynamic allows the identification of the precedence relations and indirect causality between certain transitions firing. This is expressed in the algorithm in the form of rules of priority (after the enrichment of the marking, a certain transition is not fired if another is enabled). In the expression of the results (scenarios), this appears in the form of indirect precedence/causality relations between transitions which are not related with a place. So we restrain the number of generated scenarios and for each scenario the set of sequences of transitions firing is consistent

with the strict partial order associated to the scenario. We note that only one execution of the algorithm generates automatically several scenarios. All the possible and coherent scenarios with respect to the continuous dynamic and the temporal constraints of the system are generated. The principal dada structures and functions are illustrated in the appendixes (more details are given in [17]).

3.2 TINA tool

Although TINA tool is dedicated to the ordinary Petri nets and the t-temporal Petri nets and not to Petri nets associated with differentials-algebra equations, it has several advantages. First, its graphic editor permits to describe the Petri net model of the system: the transition time interval of TINA is used to express the rule of priority between transitions firing, the label of places is used to define the nominal behavior (N) and the label of transitions to define the feared events (red) and forbidden transitions (F), ect. Then we generate two input files of ESA_PetriNet: the first file corresponds to a textual description of the Petri net model of the system and the second contains the invariants of places (structural analysis).

3.3 Principal Functions

Operating mode: extraction of feared scenarios for reliability needs a backward and a forward reasoning (mode 2). Our approach (tool) is extended towards the checking of certain properties of the computer-controlled systems as it will be shortly mentioned in the conclusion. In this case, the generation of scenarios is done only by a simple backward research (mode 1).

Extraction of scenarios: it is the principal function. After analysis of the input files, the tool extracts the necessary data structure for the algorithm and generates the scenarios.

Recording results: generated scenarios are memorized in a textual file. We extract all the scenarios (normal operation, reconfigurations and feared scenarios) to obtain precise information concerning the dynamic of the system.

Precedence graph: we have chosen precedence graph to present the generated scenarios. Direct and indirect causality relations are illustrated in different color.

4. Case Study

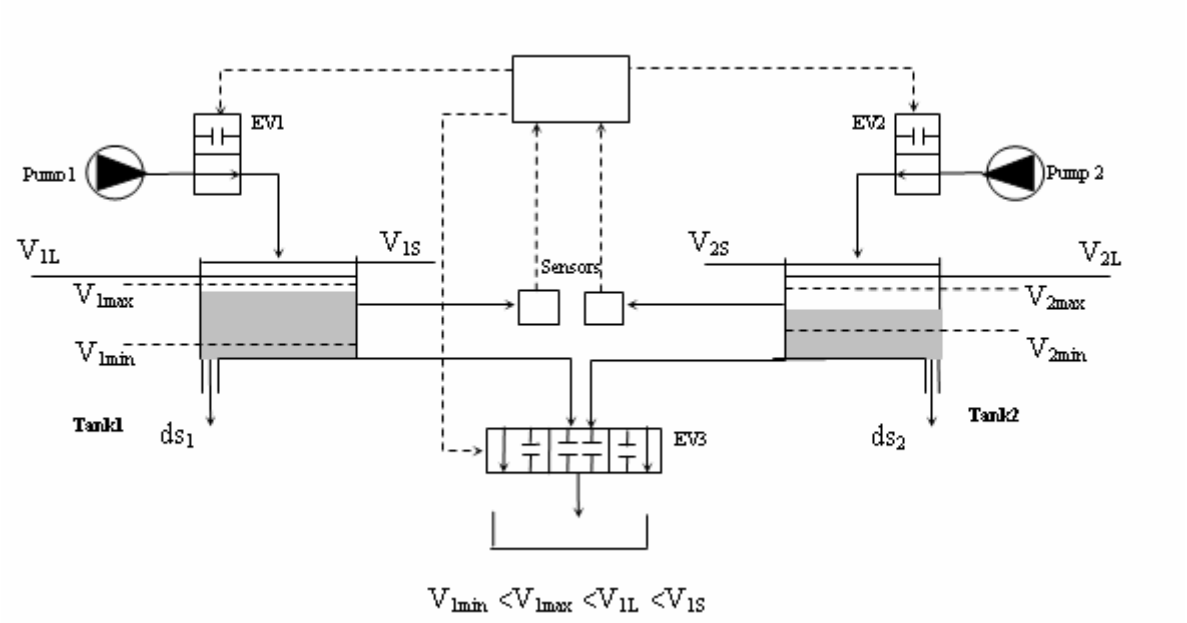


Figure 3. Case study

4.1 Presentation

It concerns a volume regulation system of two tanks (Figure 3) that was presented in [21]. The operating of this simple academic case study can be similar to a reel industrial hybrid system. It is made up of a computer, two pumps, three electro-valves, two volume sensors, the two regulated tanks (tank1 and tank2) and a third tank for draining. The demand is specified by a function of time (outgoing flowrates *ds1(t)* and *ds2(t)*). The volume of each tank *i* must be kept within a given interval *[Vimin, Vimax]*. The volume is controlled by the computer, which decides, according to the values given by the volume sensors, to fill (or not) the concerned tank by opening (or not) the concerned electro-valve. The control law of the computer is such that the electro-valve is closed when the volume of the controlled tank oversteps the upper limit *Vimax*. In the other hand, the computer commands the opening of the electro-valve each time the value of the volume in the controlled tank is lower than the limit *Vimin*. We distinguish two normal phases of the system, corresponding to the state of the electro-valve:

- A conjunction phase when the electro-valve is open. The volume in the tank is going up; no matter what is the value of the outgoing flowrate (the pump flowrate is much higher than the outgoing flowrate).
- A disjunction phase when the electro-valve is closed. The volume in the tank is decreasing.

This system must avoid the overflow of the tanks. A backup electro-valve is added to the system in order to drain the tanks in case of overflow. This third electro-valve is viewed as a shared resource between the two tanks, and it can be used to drain a unique tank at a time. When the volume of one tank oversteps the security limit *ViL*, the computer commands the opening of the backup electro-valve until the volume becomes lower than *Vimin*. As we focus our study on critical scenarios, and in order to simplify the problem we consider that only the electro-valves can have failures. A typical failure of the electro-valves *EV1* and *EV2* corresponds to a blocked open state in which the electro-valve does not react to a closure command of the computer. These two electro-valves can be repaired after a failure occurrence. When the electro-valve *EV3* has a failure it is considered to be definitively out of service

4.2 Petri Net Modeling

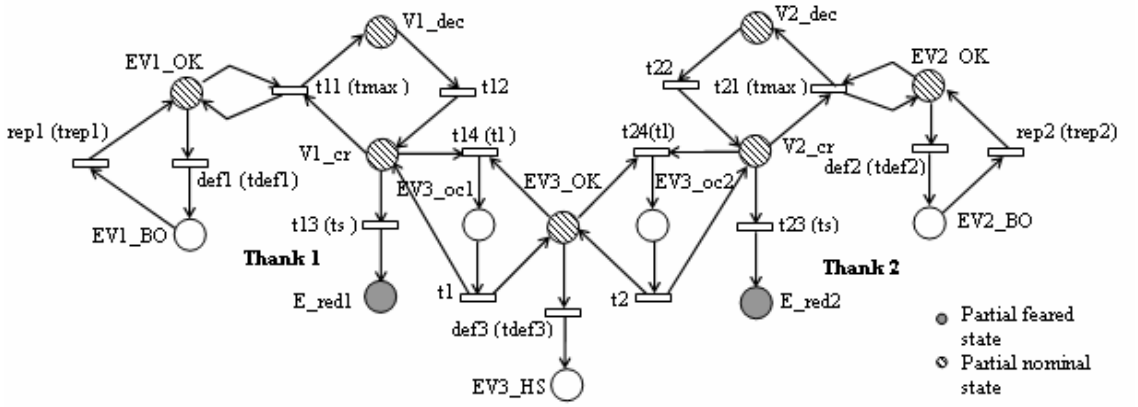


Figure 4. Petri net model of the regulation system

We have used the same Petri net model of [21] and we have added the temporal abstraction on the transition [16] as presented in Figure 4. Place *V1_dec* of the net represents the disjunction phase (the volume is decreasing); place *V1_cr* represents the conjunction phase in which the volume is increasing. Place *EV1_OK* corresponds to a state where the electro-valve *EV1* works. Transition *t11* represents the closing command of the electro-valve *EV1* when the volume oversteps *V1max*. Transition *t12* represents the opening command of the same electro-valve when the volume becomes lower than *V1min*. Transitions *def1* and *rep1* represent the fact that the electro-valve can stay blocked in an open state *(def1)*, and can be repaired *(rep1)*. Tank2 is modeled in the same way. When the volume in the tank1 oversteps the high security limit *(V1L)*, and the backup electro-valve is available (place *EV3_OK* is marked) then *t14* becomes enabled and the draining process of tank1 can start via the backup electro-valve by marking

place *EV3_oc1*. The backup electro-valve is no longer available for use to drain tank2; this corresponds to the place *EV3_OK* empty. This phase last the time that it takes for the volume to reach the low threshold *V1min*. Then, the electro-valve *EV3* is released (place *EV3_OK* is newly marked), and a conjunction phase is started again (place *V1_cr* is marked) by firing transition *t15*. The electro-valve *EV3* can have a failure (modeled by transition *def3*). In this case, place *EV3_HS* is marked and the electro-valve is set out of order. The system contains the following invariants of places:

$$M(EV1\_BO) + M(EV1\_OK) = 1$$

$$M(EV2\_BO) + M(EV2\_OK) = 1$$

$$M(EV3\_oc1) + M(E\_red1) + M(V1\_cr) + M(V1\_dec) = 1$$

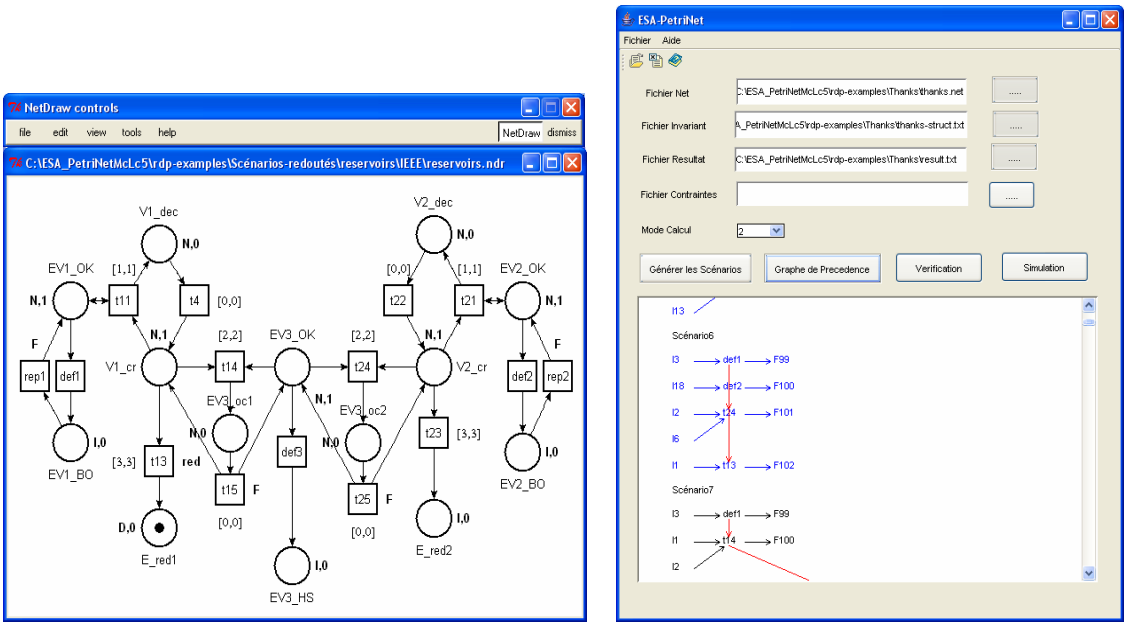
$$M(EV3\_oc2) + M(E\_red2) + M(V2\_cr) + M(V2\_dec) = 1$$

$$M(EV3\_HS) + M(EV3\_oc1) + M(EV3\_oc2) + M(EV3\_OK) = 1$$

We have chosen the following parameters:

tmax = [1, 1], tl = [2, 2], ts = [3, 3], tdefi = trepi = [0, ∞[(time unite)

4.3 Extraction of feared scenarios



TINA ⟶ Input files ⟶ ESA_PetriNet ⟶ Scenarios

Figure 5. Screen shots of TINA and ESA_PetriNet tools

A general view of ESA_PetriNet and TINA tools is given in Figure 5. Places labeled with *N* represent normal operating and transitions labeled with *F* are forbidden to avoid loops due to reparations. To use ESA_PetriNet, we first edit the Petri net model of the system on the graphic editor of TINA tool to generate two input files: a descriptive file of the Petri net model and a file containing the invariants of places. Generated scenarios can be illustrated in the form of a precedence graph (feared scenarios are illustrated with a different color to facilitate their identification among those of normal operating and reconfiguration). ESA_PetriNet generates a total of 12 scenarios (nominal, reconfiguration and feared) in which 8 are feared (Figure 6a). Note that the actual version of ESA_PetriNet generates non minimal scenarios, so most of the generated scenarios are redundant. This explains the important number of the

generated scenarios. Note also that this version takes into account continuous dynamic and temporal constraints and an important number of impossible scenarios are yet eliminated. Indeed because the continuous dynamic of the system is not taken into account in [14], not only the number of the generated scenarios is important (29 feared scenarios among a total of 51 scenarios) but also the order of appearance of the events is not respected. This order is not respected also by the failure trees.

The 8 feared scenarios (overflow) correspond to the following situations: 1) Failure of the electro-valve *EV1* (firing of the transition *def1*) and the failure of the backup electro-valve *EV3* (firing of the transition *def3*): *sc1*, *sc2*, *sc4* and *sc6*. These scenarios are represented by *{def1, def3, t13}*. 2) Failure of the electro-valve *EV1* (firing of the transition *def1*) and the use of the backup electro-valve *EV3* by the second tank (firing of the transition *t24*): *sc7*, *sc9*, *sc11*, *sc12*. These scenarios are represented by *{def1, t24, t13}*. As a matter of fact the consequence of the thresholds associated with transitions *t11*, *t14* and *t13*, is that the transition *t13* will be fired only if *t11* and *t14* are not enabled. This means that the feared scenarios are composed by fragments containing transitions in conflict with *t11* and *t14*, and by the firing of *t13*. For example the following scenario *{t13, def1, t23, def2}* given by an old version [14] (that does not take into account the continuous aspect of the system) is not produced in this new version because transition *t14* that is in conflict with *t13* has an inferior threshold so it is fired before and forbids the firing of *t13*. In the precedence graph of scenario *cs1* (Figure 6b), the indirect precedence relation is presented with the red color. The blue color corresponds to the indirect precedence relation of the second feared state *(E_red2)*. Note that the minimal scenario (only the necessary events to reach the feared state) of *sc1* is *{def1, def3, t13}*.

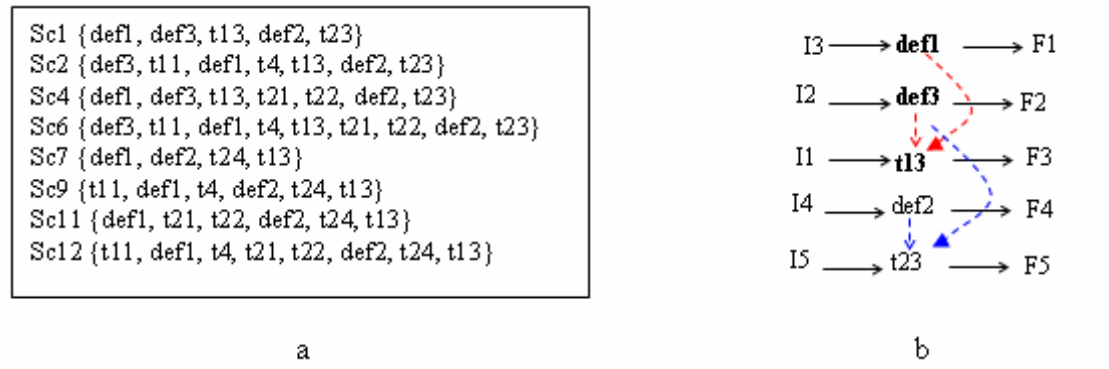


Figure 6. The generated feared scenarios

5. Conclusion

We have presented in this paper a method automated to result ESA_PetriNet tool (Extraction & Scenarios Analyzer by Petri Net model) developed for extraction of feared scenarios from the Petri net model of computers-controlled systems. The taking into account of the continuous dynamic of these systems by temporal abstraction allows the elimination of a significant number of incoherence scenarios (relating to the continuous dynamic) and the respect of the order of appearance of the events. The computing time takes only few seconds. This tool has been used to generate feared scenarios from reel industrial systems of significant size: a *Rafale* landing gears control system of *Dassault Aviation* [18] and a decentralized radio-based railway level crossing control system [22]. The aim in the last system, taken from a realistic specification of a new radio-based train control system [23] developed for the German Railways, was the evaluation of the safety [24] level to avoid collision. Note that we have improved some functions of ESA_PetriNet in [22] relating to the exploration way of the Petri net.

As it is mentioned in section 4, ESA_PetriNet was adapted to the checking of some properties (determining if the system satisfies certain properties like the duration of a scenario or accessibility between two states). A simple back exploration is enough to generate all the scenarios leading to the target state. Then a temporal abstraction is used to obtain temporal constraints networks. ESA_PetriNet has been used in the precedent landing gears system to check that the duration of a scenario is lower than certain limit [25] [17]. It is important to note that we have implemented Monte Carlo simulation [12] in this tool to quantify the probability of occurrence of these scenarios [26]. To improve this tool, we have to take into account the minimality of the scenarios to eliminate the unnecessary events and redundancy. We

have also to improve the mechanism of enrichment of the marking (in some complex systems the invariants of places is not sufficient). The checking part can be supported by algorithms of research longest ways [27] [28].

Appendix 1: Principal data structure

1.1 Input data: they are composed of the list of initial tokens *(Li)* and of the list of the normal tokens *(Ln)* which will be used as one of the two stops criteria of the algorithm. This one corresponds to the presence of only nominal tokens those are not initials *(LjnInit)* in the current list. A list of forbidden transitions *LintEntree* used to block certain transitions of the Petri net from the beginning of the reasoning.

1.2 Output data: the algorithm generates a set of partial orders; some corresponds to the nominal operating, others to the reconfigurations or to the different feared scenarios. Each partial order is defined by a triplet of the form *(E, A, B)* such as *E* is the list of firing transitions, *A* and *B* are lists of precedence relation generating partial orders. *A* is the list of arcs linking up elements of *E* having direct causality relations, *B* the list of the arcs linking up elements of *E* having indirect causality relations. The algorithm generates also a list of enrichment tokens *(Le)* created when we enrich the marking.

1.3 Internal data: they are composed of the list *C {C0, C1,...,Ci,...,Cn}* containing *n* elements of the context *Ci (Lc, Lint1, Lint2, Lint3, E, A, B, Le, LeInt, LjnInit)*.

- *Lc* is the current list. It contains the set of the current tokens which is updated after each transition firing. The consumed tokens are taken away and the produced one are added. To generate the list *Lc*, each token is represented in the form of a couple *(e, p)*; *e* is the event which produced this token and *p* the place which contains it.
- *Le* is the list of the tokens of the marking enrichment. A couple *(e, p)* is added to this list, after (each) coherent enrichment.
- *LeInt* is the list of forbidden tokens because of conflict in the enrichment of the marking.
- *LjnInit* is the list of nominal but not initial tokens. It contains the tokens of *Ln* those are not initial. For each transition firing *tk*, the couple *(tk, p)* is added to this list. After checking, if the place $p \in Ln$, it will be added to *LjnInit* (used as one of the two stop criteria).
- *Lint1* is the list of forbidden transitions of the first level. It contains the list of transitions that would not be fired from a current given stage. It allows managing conflicts of transitions. When a transition in conflict with other one of the same priority, is fired, it is added in this list in order to not be fired a second time; what avoids the generation of the same partial order more then once.
- *Lint2* is the list of forbidden transitions of the second level. It contains the list of enabled or potentially enabled transitions that would not be fired because they are in conflict with an enabled transition that has an inferior firing threshold.
- *Lint3* is the list of forbidden transitions of the third level. It contains the list of transitions which cannot be enriched to avoid loops by trying to enrich them.
- Other lists of internal data are generated from the current list *Lc*. These data concern all enabled and potentially enabled transitions knowing that enabled transitions are priority then those potentially enabled. The priority of treatment of transitions of these lists is:
  - *tfcEsc* is the list of enabled transitions in conflict or without conflict.
  - *Tpfc* is the list of potentially enabled transitions in conflict either with enabled transitions, or with potentially enabled transitions.
  - *Tpfsc* is the list of potentially enabled transitions without conflict.

Appendix 2: Principal functions

2.1 Fire transition *tk*: in this function, the current list is updated after the firing of the transition *tk* by removing the consumed tokens from the list *Lc* and by adding the produced one. Events are memorized in the list *E* and all arcs corresponding to a precedence relation between two events in the list *A*. As each

token is linked to the event having produced it, the correspondent precedence relation is directly obtained when it is consumed. The procedure is identical to the labeling of a proof tree in Linear Logic [6].

Fire_transition_tk
Add tk to E;
For each token (ti, P) necessary to fire tk remove (ti, p) from Lc and add (ti, tk) to A;
For each output place Ps of tk, add a token (tk, Ps) to Lc;
If Ps is a nominal place, add it to LjnInit;

2.2 Enrich marking *tk*: it consists in adding tokens to the list of current tokens *Lc* to enable the potentially enabled transitions. This corresponds to the taking into account of the state of new components of the studied system because their interaction with the components already taken into account can provide state changes. We consider two types of enrichment. The first noted *Enrich_marking1* allows the enrichment of transition *tk* of the list *Tfcpf*. The second noted *Enrich_marking2* allows the enrichment of the others potentially enabled transitions *tk*.

Enrich_marking1_tk
L: internal list of tokens initially empty;
For each transitions tj, potentially enabled in conflict with tk;
For each input place pl of tj add a token (ek, pl) to the list L;
Verify the coherence of the marking;

Enrich_marquage2_tk
L: internal list of tokens initially empty;
For each input place pl of tj add a token (ek, pl) to the list L;
Verify the coherence of the marking;

2.3 Marking coherence: it verifies the coherence of the marking. It uses the list *L* of the enrichment tokens *(ek, pl)*, the list of invariants of places and the conservative component of each invariant of places those are input dada.

Marking_coherence_tk
For each token pl of L;
For each invariant i, verify if it contains pl;
Verify the number of tokens in the places composing the invariant i, If the number is upper then the corresponding conservative component, remove the place pl from L;
If the list L is not empty, the enrichment is coherent, add the tokens of L to Lc and Le;
Else add tk to the list Lint3 to avoid loops;

2.4 Enrichment conflict: during the enrichment, we can meet situation in which we can not add tokens to all the places that need enrichment because of the incoherence of the marking. So, we add gradually the tokens in these places and we verify each time the coherence. In the case of incoherence because of an enrichment token in a current state, we memorize the context and we continue the enrichment. This procedure allows the consideration of all possible context of the system

Enrichment_conflict_pk
Add an enrichment token to the place pk;
Verify the marking coherence;
Add gradually tokens to the others places that need enrichment;
If incoherence because of pk, Memorize_context_conflict_enrichment_pk;

Memorize_ context_conflict_enrichment_pk
Add the token of place pk to the list of forbidden tokens LeInt;
Add a new element Ci (Lc, Lint1, Lint2, Lint3, E, A, B, Le, LeInt LjnInit) to C. Lint2 and Lint3 are initially empty in this new context;

After each context memorization, the token is added to the list *IntEnrich* to avoid the memorization of the same context enrichment. *IntEnrich* is a global variable of the algorithm.

2.5 Memorize context conflict transitions: as it was mentioned before, each time transitions conflict (with the same priority) is encountered during the construction of the scenario; this last is split to the same number of different scenarios as that of transitions implicated in the conflict. This procedure allows the memorization of all the necessary information for the construction of another scenario corresponding to the firing of another transition in conflict with this last.

> ***Memorize_Context_tk***
> *Add the transition tk to the list Lint1;*
> *Add a new element Ci (Lc, Lint1, Lint2, Lint3, E, A, B, Le, LeInt LjnInit) to C. Lint2 and Lint3 are initially empty in this new context;*

Others functions are used like: verifying if transition *tk* is in conflict with *tj* (they are in a structural conflict if they have at least a same input place), sorting transitions following the increasing threshold, construction the list of forbidden transitions *Lint2*, verifying if two conflict transitions have separated firing threshold to choose the priority of transitions.

Appendix 3: Different steps of the algorithm

0. Initial step

0.1. Initialize *C* with *C0(Lc, Lint1, Lint2, Lint3, E, A, B, Le, LeInt)* to generate the first partial order. *Lint1, Lint2, Lint3, LjnInit, Le, A* and *B* are empty, *Lc = Li* (set of initial tokens), *E{I1}* such as *I1* is an initial event, the natural number *Inc = 1, C{C0}*;

0.2. Define the priority of transitions firing due to the continuous aspect by associating to each transition *tk* a temporal threshold of firing *[dmin, dmax]*. The transition *tk* will be fired at the moment $\tau k \in [dmin, dmax]$.

1. Build new partial order

Let *Ci(Lc, Lint1, Lint2, Lint3, A, B, E, Le, LeInt, LjnInit)* the current context initially empty;

1.1. If *C* is empty Goto step 8;

1.2. Else

1.2.1. Memorize the first element of *C* in *Ci*;

1.2.2. Delete this element from *C*;

1.2.3. Goto step 2;

2. Different transitions lists

2.1. Generate from *Lc* all enabled *(tfcEsc)* and potentially enabled transitions *(Ltpf)*;

2.2. Delete from these lists: transitions of *E, Lint1, Lint3* and *LintEntree*;

2.3. Generate the lists: *Tpfsc* and *Tpfc*;

2.4. Goto step 3;

3. Stop criteria of construction of a partial order

3.1. If *Lc* contains only tokens of *Ln* whose are not initial *(LjnInit)* or the lists *tfcEsc*, *Tpfsc,* and *Tpfc* are all empty Goto step 7;

3.2. Else Goto step 4;

4. tfcEsc

The transitions are sorting in priority. This step solves the transitions conflict by memorizing the necessary information for the construction of the other partial orders relating to the firing of the other transitions implied in the conflict. An enrichment of the marking will be carried out if necessary.

4.1. If *TfcEsc* is empty Goto step 5;

4.2. Else

4.2.1. Sort *tfcEsc* according to the increasing threshold;

4.2.2. Let *tk* the first transition of *tfcEsc*;

4.2.3. Add to *Lint2* any enabled transition *tj* of higher threshold in conflict with *tk* (temporal intervals of firing are disjoined);

4.2.4. *Enrich_marking1_tk* if it is necessary and *Memorize_ context_conflict_enrichment_pk* if necessary;

4.2.5. If the enrichment is not coherent;

4.2.5.1. If *tk* is not in conflict with an enabled transition of lower threshold;

4.2.5.1.1. If *tk* is in conflict with a transition which does not belong to *Lint1, Lint2, Lint3, E* or *LintEntree* (intersection of temporal interval of firing);

4.2.5.1.1.1. *Memorize_context_tk*;
4.2.5.1.1.2. Add if necessary the indirect causality relation *(tj, tk)* to *B*;
4.2.5.2. *Fire_transition_tk*;
4.3. Goto step 2;
5. Tpfc
5.1. If *Tpfc* is empty do step 6;
5.2. Else
5.2.1. Sort the *Tpfsc* transitions according to the increasing threshold;
5.2.2. Choose the first transition *tk* in *Tpfc*;
5.2.3. Put the other transitions of higher threshold in structural conflict with *tk* in *Lint2*;
5.2.4. *Enrich_marking2_tk*;
5.2.5. If the enrichment is coherent;
5.2.5.1. If *tk* is not in conflict with an enabled transition of lower threshold;
5.2.5.1.1. Add if necessary the indirect causality relation *(tj, tk)* to *B*;
5.2.5.1.2. *Fire_transition_tk*;
5.3. Goto step 2;
6. Tpfsc
The marking of a potentially enabled transition without conflict is enriched and the transition is fired.
6.1. If *Tpfsc* is empty Goto step 7;
6.2. Else
6.2.1. Sort the *Tpfsc* list according to their increasing threshold;
6.2.2. Choose the first transition *tk* in *Tpfsc*;
6.2.3. *Enrich_marking2_tk*;
6.2.4. If the enrichment is coherent;
6.2.4.1. *Fire_transition_tk*;
6.3. Goto step 2;
7. Generate scenario
Memorize the constructed partial order and return to step 1;
7.1. For each atom *(ti, p)* of *Lc*, add *(ti, F)* to *A*; *F* is the final event;
7.1.1. Memorize the derived scenario *Inc* such as: *E(Inc) = E, A(Inc) = A, B(Inc) = B;*
7.1.2. Increment *Inc*;
7.2. Goto step 1;
8. Final step
Post all the generated scenarios;

Acknowledgment

The author would like to thank Hamid Demmou, Assistant Professor at Paul Sabatier University of Toulouse (France), Robert Valette, Directeur de recherche 2ème classe at LAAS-CNRS of Toulouse (France) for supervising this work during my PhD thesis at LAAS-CNRS. To access this tool, contact the author (malika.medjoudj@ec-lille.fr, actually a post doctorate under the support of the pole ST2 and the region Nord-Pas de Calais at LAGIS- Ecole Centrale de Lille) or Hamid Demmou (hamid@laas.fr).

REFERENCES

1. DUFOUR, F. and DUTUIT Y., **Dynamic Reliability: A New Model**, 13-ESREL2002 European Conference, Lyon-France, 18 au 21 Mars 2002.
2. DEVOOGHT, J., **Dynamic reliability**, Advances in Nuclear Science and Technology 25, pp. 215-278, 1997.
3. LABEAU, P.E., SMIDTS C., SWAMINATHAN S., **Dynamic Reliability: Towards an Integrated Platform for Probabilistic Risk Assessment**. Reliability Engineering and System Safety 68, pp. 219-254, 2000.
4. DEVOOGHT, J., SMIDTS C., **Probabilistic Reactor Dynamics - I: The Theory of Continuous Event Trees**, Nuclear Science and Engineering, Vol. 111, pp. 229-240, 1992.

5. DEVOOGHT, J., SMIDTS C., **Probabilistic Reactor Dynamics - II: A Monte-Carlo Study of a Fast Reactor Transcient** , Nuclear Science and Engineering, Vol. 111, pp. 241-256, 1992.

6. LEE, W.S., GROSH, D.L., TILLMAN, F.A., LIE, C.H., **Fault Tree Analysis, Methods, and Applications - A Review**, IEEE Transactions on Reliability, ISSN 0018-9529; r-34, pp. 194-203, August 1st 1985.

7. SWAMINATHAN, S., SMIDTS C., **The Event Sequence Diagram Framework for Dynamic PRA**, Reliability Engineering and System Safety 63, 1999, pp. 73-90.

8. GARRET, C.J., GUARRO S.B., **The Dynamic Flow graph Methodology for Assessing the Dependability of Embedded Software Systems**, IEEE Transactions On Systems, Man, and Cybernetics, Vol. 25, No. 5, May 1995.

9. MONCELET, G., CHRISTENSEN S., DEMMOU H., PALUDETTO M., PORRAS J., **Qualitative an Quantitative Dependability Evaluation of a Simple Mechatronic System Using Colored Petri Nets**, Workshop on practical use of colored Petri nets and DesignCPN, Aarhus, Denmark, June 98.

10. VILLÉN-ALTAMIRANO, M., VILLÉN-ALTAMIRANO J., **RESTART: A Straightforward Method for Fast Simulation of Rare Events**, Proceedings of the 1994 Winter Simulation Conference, 1994, pp. 282-294.

11. HOFER, E., KLOOS M., KRZYKACZ-HAUSMANN B., PESCHKE J., SONNENKALB M., **Method enentwicklung zur simulativen Behandlung der Stochastik in probabilistischen**, Sicherheitsanalysen der Stufe 2, Abschlußbericht, GRS-A-2997, Gesellschaft für Anlagen- und Reaktorsicherheit, Germany (2001).

12. KALOS, M.H. and WHITLOCK P.A., **Mont Carlo Methods**, Vol. 1: Basics, John Wiley and Sons, New York, 1986.

13. MURATA, T., **Petri Nets: Properties, Analysis and Applications**, IEEE Proc, Vol. 77, pp. 541-580, April 1989.

14. DEMMOU, H., KHALFAOUI S., RIVIERE N., VALETTE R., **Extracting Critical Scenarios from a Petri Net Model Using Linear Logic**, Journal Européen des Systèmes Automatisés (APII-JESA), Vol. 36, N7, 2002, pp. 987-999.

15. GIRARD, J.Y., **Linear Logic**, Theoretical Computer Science, Vol. 50, 1987, pp. 1-102.

16. MEDJOUDJ, M., KHALFAOUI S., DEMMOU H., VALETTE R., **A Method for Deriving Feared Scenarios in Hybrid Systems,** Probabilistic Safety Assessment and Management (PSAM7-ESREL04), Berlin-Germany, 14-18 June 2004.

17. MEDJOUDJ, M., **Contribution à l'analyse des systèmes pilotés par calculateurs: Extraction de scénarios redoutés et vérification de contraintes temporelles**, Thèse doctorale de l'Université Paul Sabatier, Toulouse-France, Mars 2006.

18. MEDJOUDJ, M., DEMMOU H., VALETTE R., **ESA_PetriNet tool: Extraction Scenarios & Analyzer by Petri Net Model : Application to the Extraction of Feared Scenarios in a Landing Gear System**, European Simulation and Modeling Conference (ESM2006), LAAS-Toulouse-France, 23-25 October 2006, pp. 375-382.

19. BERTHOMIEU, B., RIBET P.O., VERNADAT F., **The tool TINA - Construction of Abstract State Spaces for Petri Nets and Time Petri Nets**, International Journal of Production Research, Vol. 42, No. 14, 15 July 2004, pp. 2741-2756.

20. CHAMPAGNAT, R., ESTEBAN P., PINGAUD H., VALETTE R., **Modeling and Simulation of a Hybrid System Through Pr/Tr PN DAE Model**, ADPM'98 3rd International Conference on Automation of Mixed Processes, Reims-France, 19-20 March 1998, pp. 131-137.

21. DEMMOU, H., KHALFAOUI S., GUILHEM E., VALETTE R., **Critical Scenarios Derivation Methodology for Mechatronic Systems**, Reliability Engineering & System Safety, Vol. 84, No. 1, April 2004, pp.33-44.

22. MEDJOUDJ, M. and YIM P., **Extraction of Critical Scenarios in a Railway Level Crossing Control System**, International Journal of Computers, Communication and Control (IJCCC) Vol. II, No. 3, 2007, pp. 252-268.

23. JANSEN, L. and SCHNIEDER E., **Traffic Control Systems Case Study: Problem Description and a Note on Domain-based Software Specification**, Technical rapport, Technical University of Braunschweig, 2000.

24. LAPRIE, J.C., **Dependability: Basic Concepts and Terminology**, Vol. 5, Springer, 1992.

25. RIVIERE, N., DEMMOU H., VALETTE R., MEDJOUDJ M., **Symbolic Temporal Constraint Analysis, an Approach for Verifying Hybrid Systems**, 16th IFAC World Congress, Prague-République Tchèque, 3-8 July 2005.

26. MEDJOUDJ, M. and LABEAU P.E., **Estimation Monte Carlo de la probabilité d'atteindre des états redoutés basée sur la prédétermination de ces scénarios**, 12P, PENTOM, Mons-Belgique. 9-10 juillet 2007.

27. FLOYD, R.W., **Algorithm 97: Shortest Path**, Communications of the ACM Vol.5 Issue 6, page 345, June 1962.

28. WARSHALL, S., **A Theorem on Boolean Matrices**, Journal of the ACM Vol. 9 Issue .1, pp. 11-12, January 1962.