ESA PetriNet: Petri net Based Tool for Reliability Analysis

Romaric Guillerm, Hamid Demmou, Nabil Sadou

ESA_PetriNet: Petri net Based Tool for Reliability Analysis

R. GUILLERM, H. DEMMOU
CNRS ; LAAS
7 avenue du colonel Roche,
F-31077 Toulouse, France
University of Toulouse ; UPS, INSA, INP, ISAE ; LAAS,
F-31077 Toulouse, France
rguiller@laas.fr
demmou@laas.fr

N. SADOU
SUPELEC - IETR
Avenue de la boulais
BP 8112
F35511 Cesson-Sevigne
nabil.sadou@supelec.fr

***Abstract*—This paper describes the critical (feared) scenarios derivation tool ESA_PetriNet (Extraction Scenarios Algorithm from Petri Net) available from : http://www.laas.fr/ESA. ESA_PetriNet allows to derive scenarios leading to critical (feared) situation in embedded systems. The system model is given by a Petri net. To derive critical scenarios and to avoid the state space explosion, the solution is to use directly the Petri net model. Linear logic (which does not appears in this paper) offers a theoretical framework to interpret the Petri net model and to extract the scenarios. ESA_PetriNet provides all minimal scenarios which contain strictly necessary and sufficient events to reach a specified state. ESA_PetriNet can be used with classical Petri net modelling or in its objects oriented version.**

***Index Terms*—Embedded systems, Feared scenarios extraction, Petri nets, reliability, Dynamic systems.**

I. Introduction

The growing complexity of embedded systems [1], including more and more ECU (Electronic Control Units), makes it difficult to perform the dependability analysis of such systems. Considering mechatronic systems in the automotive domain and focusing on dependability analysis, one way to help designers is to identify critical scenarios and define corrective actions to avoid them as early as possible in the design stage. This means that when some event affecting the safety of the system occurs, a reconfiguration action is executed in order to maintain the vehicle in a safe degraded state. If the reconfiguration fails then the system will reach a critical (dangerous) state with dramatic consequences for the passengers. So it is important to understand how the system reaches such critical states at the early design stage of the system in order to set up the reconfiguration actions.

The safety analysis [2] of dynamic embedded systems starts with a qualitative analysis, the objective of which is to determine the catastrophic or unacceptable behavior of the system.

For static systems the most popular approach for reliability analysis is based on fault trees with its associated tool [3]. Static fault trees use traditional Boolean Logic functions to represent the combination of component failures (events) that cause system failure. One interesting aspect of fault trees is that a set of minimal cutsets [4] can be derived. However, to deal with the complexity of dynamic systems, fault trees are not sufficient: safety analysis of such systems must include timing considerations and the order of events [5].

To overcome these limitations, Dugan [6] introduces a new type of gate in order to differentiate the static aspects from the dynamic ones. The dynamic gates allow Markov analysis. One drawback is that it is not possible to derive a qualitative analysis, and moreover some produced feared scenarios can lead to non-permanent feared states.

Simulation is another popular method, in particular the Monte-Carlo simulation [7]. Good results can be obtained when simulation is applied. Nevertheless, when used for dependability analysis simulation methods have to deal with the so-called ”rare events problem”.

Other methods based on the exploration of the state graph, like the Markov graph or the Petri net accessibility graph[8], are limited by the problem of state space explosion.

So, to achieve the qualitative analysis we propose an approach focused on the search for critical scenarios in order to propose a way to avoid the problem of state space explosion. The basic idea is to use a Petri net model and directly extract the critical scenarios without building the accessibility graph.

Linear logic [9], [10] offers a formal framework to interpret the Petri net model and to extract the scenarios. The key point of this approach is equivalence between reachability in the Petri net and linear logic sequent provability [11]. Linear logic thus makes it possible to analyze these cause-effect relations.

To model Embedded systems, temporal Petri nets are used. They allow to model some continuous aspects of these systems. This modelling approach has the advantage of clearly separating the continuous aspects (time modelling) from the discrete ones. It allows a logical analysis using linear logic based on the causality of events leading to a critical state.

Starting from a critical state, it is possible to go back through the chain of causalities and to point out only those scenarios leading to a critical situation. Each scenario is a

978-1-4244-2794-9/09/$25.00

representation of the partial order of events necessary for the occurrence of the critical outcome. The final objective is to determine minimal scenarios. Indeed, a critical scenario can contain events (the consequence of other events of the scenario) which are not strictly necessary to reach the final critical state. Such a scenario is not minimal. To deal with the concept of minimality of a scenario we introduce a formal definition of a minimal scenario. This definition has been used in our algorithm to derive automatically critical scenarios, but only the minimal ones. Another application of the proposed approach is the verification of behavioral properties like safety.

This paper is organized as follows: section 2 presents briefly the deriving scenarios extraction approach. Section 3 describes ESA_PetriNet tool. In section 4 an example illustrates the capabilities of the tool. Finally, section 5 draws a number of conclusions and future work.

II. Scenario extraction approach

A. Critical scenario

Critical scenario is a set of events (transition firings for a Petri net model) verifying a partial order and leading from one partial state corresponding to normal behavior (partial marking) to another one that represents a dangerous situation of the system.

In this part we present briefly the approach for deriving critical scenarios for dynamic reliability analysis.

B. System modelling

The system modelling is based on Petri net [12] using or not the object-oriented concepts [13], [14]. Object oriented approach can be used to facilitate the modelling and analysis of complex systems.

There are several proposals incorporating the notion of time into different components of the Petri net framework, namely tokens, transitions, places, and arcs [15]. In this work we use a temporal Petri net where the durations are associated with transitions . Let us now give the formal definition of the temporal Petri nets.

Definition [temporal Petri net]: A Temporal Petri net is a pair $N_{tl} =< N, D >$ where N is a Petri net $< P, T, Pre, Post >$ and D is a function that associates to each transition t_i a static temporal interval $d(t_i) = [dimin(t_i), dimax(t_i)]$ that describes the enabling duration.

C. Principles of the approach

The method is based on a qualitative analysis stemming from the Petri net model. The objective is to extract and clearly identify the critical scenarios (leading from one partial state corresponding to normal behavior to another one that represents a dangerous situation of the system) starting from a model that contains the necessary knowledge to make the analysis. The initial partial knowledge of the critical state is progressively enriched while analyzing the components necessary to its occurrence. This method is made up of two steps: a backward and a forward reasoning process. The backward reasoning starts from the partial critical state in order to derive the events that are necessary to reach it, and gives the last nominal states preceding the critical behavior. The forward reasoning starts from these nominal states, and determines the components at the source of the critical scenario. To determine the complete context in which the critical scenario occurs, the concept of context enrichment is introduced. The context enrichment is carried out by adding tokens to some places (empty input places of potentially enabled transitions) that can have an impact on the critical scenario that is being explored (adding one token in the place p_0 to make the transition t_0, initially potentially enabled, enabled in the example of the Figure 1)).

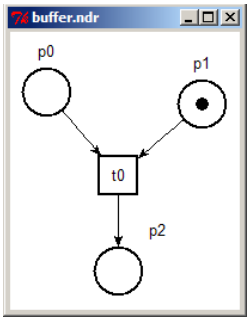

Fig. 1. Marking enrichment

D. Principles of scenario extraction algorithm

The deriving scenarios algorithm (both backward and forward reasoning) [13] can be considered as a Petri net player, but not classical Petri net player (occurrence graph). It is a player based on linear logic that guides the construction of the partial orders between events [9].

E. Steps of the method

1) Determining the nominal states
2) Determining the target states (partial critical states or states to be analyzed).
3) Backward reasoning starting from the partial critical state.
4) Forward reasoning starting from the objects that contain the conditioning state.

1) The first step is to determine the places that, when marked, represent a normal operation state. These 'nominal' places will be used as "stop criteria" for backward reasoning. This step can be achieved in two ways: by using a priori knowledge of the normal operation states of the system, or by a Monte Carlo simulation of the model (in a short temporal window) in order to determine the marking probabilities of the places of the Petri net. The places that will have a significant marking probability will be considered as nominal places.

2) The second step determines the target states to be analyzed. This target states can be either a partial critical state or another partial state with a direct or indirect link to the critical state (for example a place that represents the availability of a

resource that allows operating even with presence of a fault, and avoids the occurrence of the critical event).

3) Backward reasoning: The aim of this step is to determine the ”conditioning states” that correspond to the last normal operation state of the system. It generates the sets of paths that lead to the partial critical state. Backward reasoning is carried out on the reversed Petri net model. In this reversed Petri net, the initial marking corresponds to each minimal Cutset of the Boolean function that represents a critical state. We search for all the minimal scenarios (only the necessary transitions are fired) that lead from the initial marking to a final marking, containing only places that are associated with normal operation. During this step, starting from the initial context that concerns one or more objects, in most cases we have to enrich the initial context.

4) Forward reasoning: This step is carried out on the initial Petri net with the ’conditioning states’ as initial marking. The goal is to determine the reachability of the critical state and to identify the scenarios that lead to it, or to prove that the critical state will never be reached. It aims also to identify the bifurcations between the normal behavior and the critical one and also to set the conditions (marking of some places and introduced objects) associated to these bifurcations. The analysis of these bifurcations (that correspond to transitions) gives information about the occurrence of events which are the causes of the critical state.

III. Tool presentation

A. Objectives of ESA_PetriNet tool

ESA_PetriNet (Figure 2) implements the deriving critical scenarios algorithm. It allows to generate critical scenarios leading the system from normal working to critical situation.

For designers the interesting scenarios are minimal ones. Minimal scenario [16] means that it contains only necessary events to reach one marking from another one. So minimality analysis [16] is implemented in to ESA_PetriNet. It allows to derive only consistent scenarios.

Another functionality of ESA_PetriNet is the verification of behavior properties. Indeed, it is possible to prove that one scenario respects some temporal constraints. In this paper the approach for verification is not presented. The principles of the approach can be found in [17].

B. Input files

1) Petri net models and structural analysis: The input files of the tool correspond to a textual description of the temporal Petri net model of the system. To edit these files, the tool TINA (TIme Petri Net Analyzer) [18], [19](Figure 3) is used. The definitions of the nominal states and the target states (referring to step 1 and 2 of the method) are also done in these files, through the labels associated to the places of the Petri nets.

The Petri net models can be modified directly from ESA_PetriNet by editing TINA files in their textual or graphical description.

More specifically, the input files corresponding to the textual description of the Petri nets are the result files of the structural analysis (Figure 5) offered by TINA. These files are used because they contain both the descriptions of the Petri nets and the marking invariants necessary for marking enrichments. Indeed, marking enrichments are carried out during the feared scenario research algorithm, but they are not always possible. And to preserve the system consistency, that’s the marking invariants which determine if the marking enrichments are possible or not.

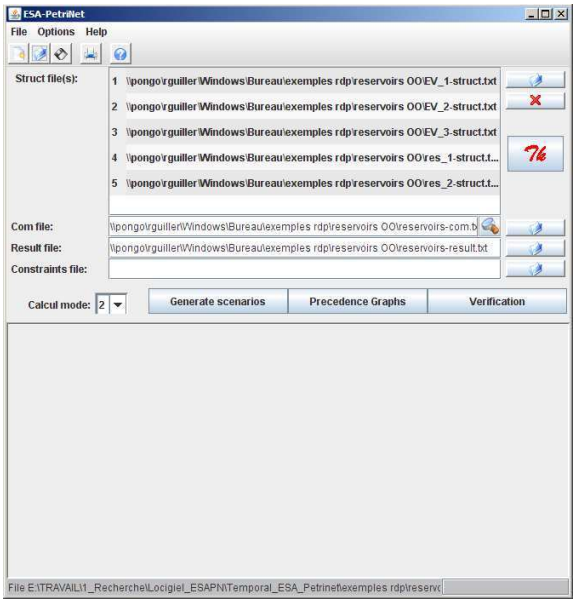

Fig. 2. ESA_PetriNet Screen snapshot

2) Communication file: For the object oriented approach, each object is modelled by one Petri net and the communication between objects is specified. Figure 4 shows communications between objects of the case study. It will be commented later.

C. output files

1) Result File: The single output file provided by an analysis with ESA_PetriNet is the result file. It contains a textual description of the generated feared scenarios.

But a common way to represent scenarios is in the form of graphs, because they clearly show the sequence of events, partial orders and the parallelism between some events. Indeed, a precedence graph is a directed acyclic graph defined by a set of events (those of the scenario) and a precedence relationship that corresponds to the partial order of the scenario created by the various links between the events. Consequently, ESA_PetriNet offers the possibility to display the scenarios in the forme of precedence graphs starting from the result file by clicking on the ”Precedence Graphs” button.

D. User interface

In order to obtain the critical scenarios with ESA_PetriNet, we must indicate some files through the user interface (Figure 2). Firstly, all the Petri net objects that we need must be given to the software. Each Petri net that models one

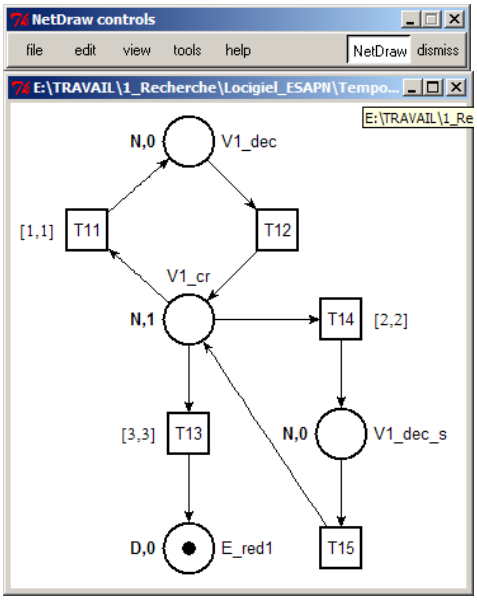

Fig. 3. TINA snapshot

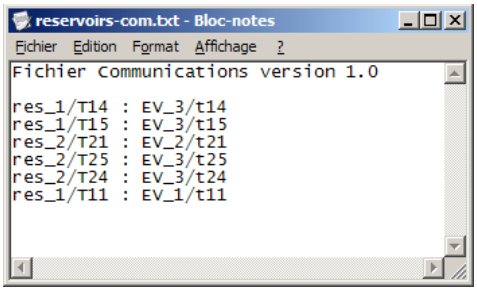

Fig. 4. Communication file

object is specified. We have already seen that these files are created and obtained with another tool: TINA. Next, we must define a communication file in order to keep the consistency of the model, eventually by using the editor provided with ESA_PetriNet. The last file that must be defined trough the interface is the result file.

Once all the needed files have been specified, we run the extraction of the critical scenarios by clicking on the dedicated button: "Generate Scenarios". Finally, the precedence graphs are displayed by using the "Precedence Graphs" button.

E. Installation

ESA_PetriNet can be downloaded from www.laas.fr/ESA. ESA_PetriNet is coded in Java language, so a Java virtual machine is needed. Windows OS version is available. The Linux version will be developed later.

IV. Example and Application

The case study (inspired from automotive industy) is based on a volume regulation system of two tanks (Figure 6). It consists of a computer, two pumps, three electrovalves, two

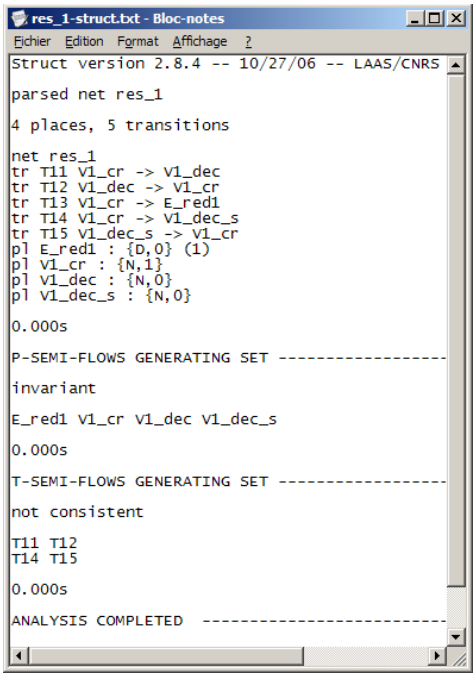

Fig. 5. Structural analysis results file

volume sensors, the two regulated tanks (Tank1 and Tank2) and a third tank for draining. The two regulated tanks are used on demand of a user. This demand is described by a function of time. The volume of each tank must be kept inside 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 electrovalve.

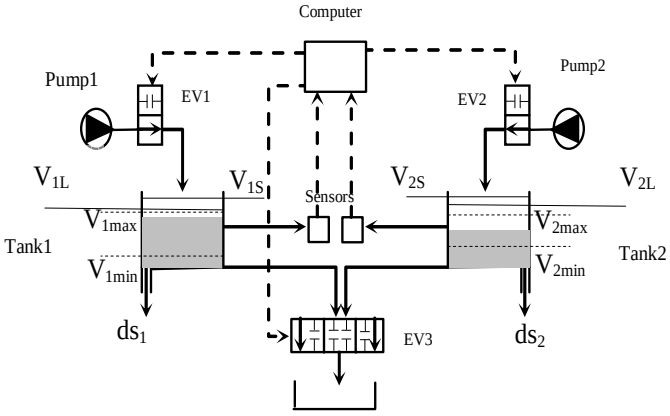

Fig. 6. Case study

The control strategy of the computer is such that the electrovalve is closed whenever the volume of the controlled tank exceeds the upper limit Vimax (in the conjunction phase). On the other hand, the computer commands the opening of the electrovalve each time the value of the volume in the controlled tank is lower than the limit Vimin (in the disjunction

phase). We distinguish two normal phases of the system, corresponding to the state of the electrovalve:

- A conjunction phase when the electrovalve is open. The volume in the tank is increasing during this phase, no matter what is the value of the outgoing flow to the user (the ingoing flow to the electrovalve is much higher than the outgoing flow).
- A disjunction phase when the electrovalve is closed. The volume in the tank is decreasing during this phase.

This system must supply the user while avoiding the overflow of the tanks. A relief electrovalve is added to the system in order to drain the tanks in case of overflow. This third electrovalve is viewed as a shared resource between the two main tanks, and it can be used for only one tank at a time. When the volume of one tank exceeds the high security limit (ViL), the computer commands the opening of the relief electrovalve. As we focus our study on critical scenarios, and in order to simplify the problem we consider that only the electrovalves can have failures. A typical failure of the electrovalves one and two corresponds to a stuck open (or closed) state in which the electrovalve does not react to a closure (or opening) command of the computer. These two electrovalves can be repaired after failure occurrence. When the electrovalve 3 has a failure it is definitively out of service.

A. Modelling

We propose 3 object classes for the modelling of this system: one tank class and two electrovalve classes. We define two objects (tank1 and tank2) instances of tank class, two objects (EV1 and EV2 electrovalves) instances of the first electrovalve class and one object (relief electrovalve) instance of the third class. In order to deal with the continuous aspects of the system, the volume thresholds (Vimin, Vimax, ViL, Vis) that correspond to the enabling functions of the Differential Predicate Transition Petri Nets are replaced by temporal thresholds (Timin, Timax, TiL, Tis). These temporal thresholds are obtained by temporal abstraction of the continuous dynamic associated to places Vi_cr and Vi_dec.

1) Model of the tanks: (Figure 3) shows the model of Tank1 (the model of the tank2 is identical). Place V1_dec represents the disjunction phase (the volume is decreasing), and the place V1_cr represents the conjunction phase in which the volume is increasing. When the volume exceeds V1max (temporal interval [1,1]) the tank calls a 'close electrovalve_1' method (associated with transition T11) provided by the electrovalve EV1. The method 'Open electrovalve_1' (associated with transition T12) is called by the Tank1 when the volume becomes lower than V1min (interval [1,1]). When the volume in the Tank1 exceeds the high security limit V1L (interval [2,2]), the 'open electrovalve_3' method (transition T14) provided by the relief electrovalve is called in order to drain Tank1. This phase lasts the time necessary for the volume to reach the low threshold V1min. A conjunction phase is started again (place V1_cr is marked) by firing transition T15.

There is overflow on the Tank1, when the volume in this tank exceeds V1S (V1S reached in the interval [3,3] is higher than V1max and V1L). In this case, transition T13 is fired and place E_red1 is marked. The firing of T13 is considered as the critical event.

2) Model of the electrovalve 1 (EV1): The electrovalve 1 (EV1) when opened supplies the Tank1 in the conjunction phase (Figure 7).

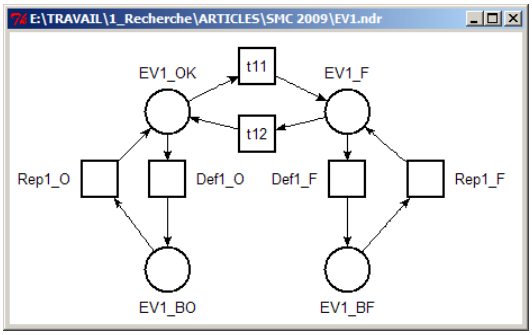

Fig. 7. Petri net model of the electrovalve 1

The EV1 is closed (firing of transition t11) when the Tank1 calls the 'close electrovalve_1' method. The transition t12 represents the opening of the electrovalve. The two methods provided by the EV1 ensure that the volume in the Tank1 is kept inside a given interval [V1min, V1max].

The failures of the electrovalve are represented by the transitions Def1_O (stuck open) or Def1_F (stuck closed). The transitions Rep1_O and Rep1_F represent the repair of the electrovalve. Duration is associated with the firing of these transitions.

3) Model of the relief electrovalve 3: The relief electrovalve is shared between the two tanks and its model is presented in (Figure 8). The place EV3_OK represents the availability of the valve. The transition t14 (respectively t24) is the 'open electrovalve_3' provided method that can be called by the Tank1 (respectively Tank2) when the volume exceeds V1L (respectively V2L). The electrovalve 3 can fail (firing of the transition def3). In this case, place EV3_HS is marked and the electrovalve is out of order.

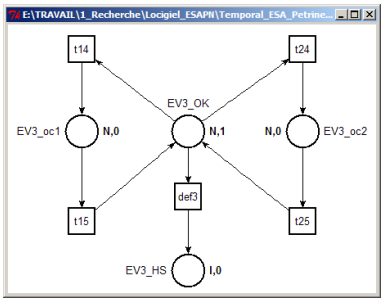

Fig. 8. Petri net model of the relief electrovalve.

Remark:

1) We can see in figure 4 (textual file which represents the communications between objects). For example, the transition T11 of the tank1 calls the transition t11 of the electrovalve 1.

2) In figure 5 we can see the results of the structural analysis. For example, for the object tank1, there is a P-invariant that corresponds to the places : E_red1 V1_cr V1_dec V1_dec_s. So when one of these places is marked, it is not possible to enrich the other places.

B. Application of the method and results

ESA_PetriNet tool derives 2 minimal critical scenarios (Figure 9) leading to the critical state E_red1 (overflow of the tank1).

The first scenario is composed by the events: failure of electrovalve 1, failure of the relief electrovalve (EV3), followed by the overflow of tank1 (figure 9).

The second scenario is composed by the events: failure of electrovalve 1 and failure of electrovalve 2, the use of the relief electrovalve (EV3) to drain tank2, followed by the overflow of tank1.

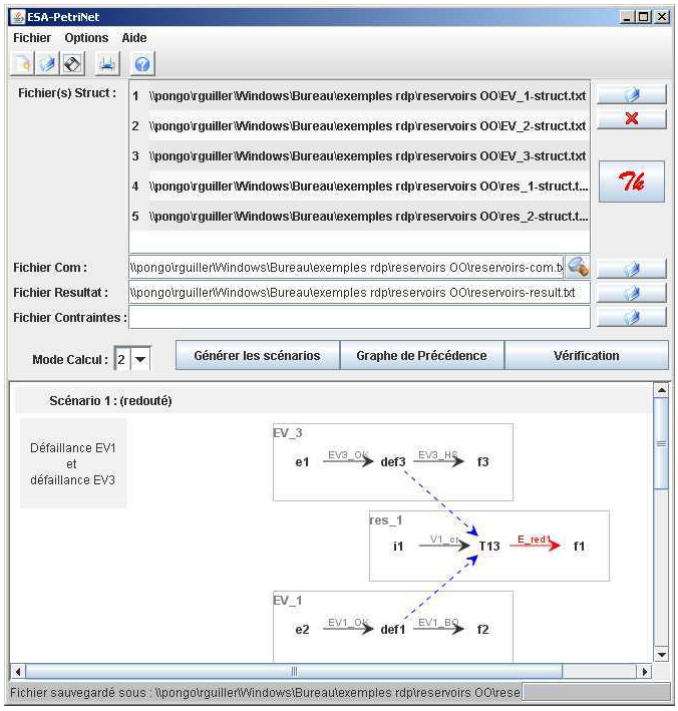

Fig. 9. Generated scenarios

V. Conclusion

In this paper we presented ESA_PetriNet tool that implements our algorithm for deriving critical scenario in embedded systems.

System modelling is based on temporal Petri net. The objects oriented approach allows to facilitate system modelling and scenario analysis.

Another important aspect is the minimality of derived scenarios. The tool allows generating only pertinent scenarios. It takes into account the notion of minimal scenario [16] which is the relevant information for designers and facilitates analysis.

In order to deal with complex dynamic (hybrid systems) and overcome the limitation of temporal abstraction (used for continuous dynamic approximation), future works will introduce the possibility of integrating complex differential equation (one way is to connect the tool to a dynamic solver).

Another extension we have to work on is the quantitative analysis. Monte Carlo simulation appears as the technique that will be used to achieve quantitative analysis and will be implemented in ESA_PetriNet.

Current software development concerns improving the interface tool specially its English version. An English web site will be dedicated for it.

References

[1] Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science 4085, Springer, 2006, pp. 1-15.

[2] F. Dufour, Y. Dutuit, "Dynamic Reliability: A new model", 13-ESREL2002 European Conference, Lyon - France - 18 au 21 Mars 2002.

[3] M. Sinnamon and J. D. Andrews, "Fault trees and binary decision diagrams," Proceedings of the Annual Reliability and Maintainability Symposium, 1996, pp. 215-222.

[4] A. Rauzy. Mathematical Foundation of Minimal Cutsets. IEEE Transactions on Reliability, 50(4):389-396, december 2001.

[5] Chris J. GARRET, Sergio B. Guarro, George E. APOSTOLAKIS, "The Dynamic Flowgraph Methodology for Assessing the Dependability of Embedded Software Systems", IEEE Transactions On Systems, Man, and Cybernetics, Vol. 25, No. 5, May 1995.

[6] J.B Dugan, T. Assaf, " Diagnostic expert systems from dynamic fault tree", In Annual Reliability and Maintainability Symposium 2004 Proceedings, LA, January 2004.

[7] P.E. Labeau: "A Survey on Monte Carlo Estimation of Small Failure Risks in Dynamic Reliability". In International Journal of Electronics and Communications, Vol. 52, pp. 205-211, 1998.

[8] D. Codetta-Raiteri1 and A. Bobbio. Stochastic Petri Nets Supporting Dynamic Reliability EvaluationStochastic Petri Nets Supporting Dynamic Reliability Evaluation. International Journal of Materials and Structural Reliability Vol.4, No.1, March 2006, 65-77.

[9] H. Demmou, S. Khalfaoui, N. Riviere, E. Guilhem,"A method for deriving critical scenarios from mechatronic systems", *Journal Europeen des Systemes Automatises*, volume 36 - n7/2002. pages 987 999.

[10] J.Y Girard, "Linear Logic ", Theoretical Computer Science, 50, 1987, p.1-102.

[11] B. Pradin-Chzalviel, R. Valette, L.A. Knzle: "Scenario duration characterization of t-timed Petri nets using linear logic", IEEE PNPM'99, 8th International Workshop on Petri Nets and Performance Models, Zaragoza, Spain, September 6-10, 1999, p.208-217.

[12] T. Murata, "Petri Nets: Properties, Analysis and Applications," Proc. IEEE, vol. 77, no. 4, pp. 541-580, 1989.

[13] N.Sadou, H.Demmou, J.C.Pascal, R.Valette, "Object oriented approach for deriving feared scenarios in hybrid system," *2005 European Simulation and Modeling Conference*, Portugal, 24-26 Octobre 2005, pp.572-578.

[14] Booch, G., et al, (1998). "The Unified Modelling Language User Guide". Addison-Wesley Longman, Inc. Harlow, England

[15] C. Ghezzi, D. Mandrioli, S . Morasca, P. Mauro. A General Way to Put Time into Petri Nets. ACM SIGSOFT Engineering Notes, 14(3)-60-67, Pittsburgh, Pennsylvania, 1989.

[16] N. Sadou, H. Demmou, "Minimality of critical scenarios in Petri net model," *2006 IEEE International Conference on Systems Man and Cybernetics (SMC'06),* Taipei (Taiwan), 8-11 October 2006, 8p.

[17] N. riviere, H.demmou, R. valette, M.medjoudj. Symbolic temporal constraint analysis, an approach for verifying hybrid systems. 16th IFAC World Congress, Prague, 3-8 Juillet 2005, 6p.

[18] TINA: http://www.laas.fr/tina/.

[19] B. Berthomieu, F. Vernadat, "Time Petri Nets Analysis with TINA," *In Proceedings of 3rd Int. Conf. on The Quantitative Evaluation of Systems (QEST 2006)*, IEEE Computer Society, 2006.