Modeling standby redundancies in repairable systems as guarded preemption mechanisms

Pierre-Yves Piriou, Jean-Marc Faure, Jean-Jacques Lesage

Modeling standby redundancies in repairable systems as guarded preemption mechanisms

Pierre-Yves Piriou, Jean-Marc Faure, Jean-Jacques Lesage

LURPA, ENS Cachan, Univ Paris-Sud, F-94235 Cachan, France
(e-mail: {pierre-yves.piriou; jean-marc.faure; jean-jacques.lesage}@lurpa.ens-cachan.fr).

Abstract: This paper proposes an extension of the BDMP (Boolean logic Driven Markov Processes) formalism for enriching its capabilities to model replacement and resumption mechanisms in repairable systems. The implicit assumptions made by the classical BDMP to describe these mechanisms are first highlighted. An analogy between standby redundancies management in critical systems and preemption mechanisms in concurrent systems is then proposed. This permits to formally define an extension of BDMP that allows several replacement and resumption mechanisms be specified. A case study illustrates the benefits of this proposal.

Keywords: Model Based Safety Analysis, Repairable component, Standby redundancy policies, Preemption, Boolean logic Driven Markov Process

1. INTRODUCTION

Component redundancy is a widely used design concept to improve safety of critical systems. It consists in duplicating a set of components of the system in order to increase the probability to achieve the aimed functions. For a standby redundancy, all the redundant components do not work simultaneously. The so called spare components are activated only if the so called main components are not able to perform their functions. When components are repairable, such a standby redundancy carries out two switching mechanisms: the replacement and the resumption. The triggering condition of these two mechanisms depend on a dysfunctional context. Hence, several redundancy policies may be defined for a given set of redundant components.

Safety analysis of a dynamic system basically consists in determining its most critical failure scenarios and assessing their probability of occurrence. These analyses are based on a model of the possible dysfunctional behavior. Boolean logic Driven Markov Processes (BDMP), defined in Bouissou and Bon (2003), is a promising modeling formalism with regard to its ability to capture the dynamic aspects that arise with complex systems. Indeed, it is well suited for dealing with repairable components, and one of its primitives (the trigger) allows to model standby redundancies (Carer et al. (2002)). Nonetheless, a unique redundancy policy can be translated through a BDMP trigger: the replacement occurs as soon as the main components fail, and the resumption occurs as soon as they are repaired. Moreover, the components which trigger these switching mechanisms are not represented in the model, and are then considered faultless.

This paper proposes an extension of the BDMP formalism for enriching its modeling capabilities. The extension aims to allow a modeling of standby redundancies more accurate than with the BDMP formalism. It mainly impacts the definition of the trigger primitive, in order to capture several redundancy policies, that can be specified as preemption mechanisms. Indeed, for concurrent systems, preemption consists in the interruption of a process, in a particular context, generally in order to trigger another process. Given that components can be seen as processes, a switching mechanism is similar to a preemption mechanism that occurs in a particular dysfunctional context.

What is above mentioned about BDMP formalism is developed and illustrated on a case study in section 2. Preemption in concurrent systems is next defined in section 3. This section also proposes an analogy with standby redundancies in critical systems. Section 4 introduces Guarded BDMP (GBDMP) as an extension of BDMP and illustrates its benefits on the case study introduced in section 2. Finally, concluding remarks are given in section 5.

2. PROBLEM STATEMENT

This section introduces first the case study illustrating the work. Secondly, it provides a recall on BDMP formalism and its benefits for performing Model Based Safety Analysis (MBSA). Finally, the modeling of standby redundancies by triggers is discussed, and illustrated on the case study.

2.1 Motivating example

The Coolant Feeding Water System (CFWS) is a crucial system of every nuclear power plant. It aims to supply cool water to the steam generator. Figure 1 provides a simplified view of this system that performs two subfunctions. First, three extraction pumps ($Ex1$, $Ex2$ and $Ex3$) provide a sufficient flow of cool water. Two out of the three must be available to fulfill the service requirements.

Second, two redundant Feeding Turbo Pumps ($FTP1$ and $FTP2$) pressurize the cool water. At least one of these pumps must be faultless.

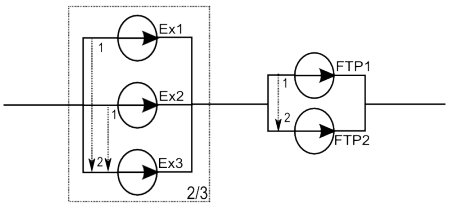

Fig. 1. A simplified view of the CFWS

Each component of this system can fail and be repaired, either in active or inactive mode. Furthermore, two different redundancy policies have been defined:

- When the main component ($Ex1$ or $Ex2$ for extraction pumps, and $FTP1$ for turbo pumps) is faulty and the spare component ($Ex3$ for the extraction pumps and $FTP2$ for the turbo pumps) is faultless, the main is replaced by the spare in both cases.
- The resumption conditions of the main component are not the same however. When an extraction pump is activated, it remains active until it fails; this implies that a main extraction pump is reactivated only if it has been repaired and the spare pump has failed. On the other hand, as $FTP1$ has better performances than $FTP2$, it is reactivated after a failure as soon as it has been repaired, even if $FTP2$ has not failed.

The objective of this work is to perform a MBSA of this system, that takes into account these two redundancy policies.

2.2 Recall on BDMP formalism

BDMP is a formalism defined in Bouissou and Bon (2003) to perform safety analysis of complex systems, and in particular repairable systems. A BDMP model can be implicitly defined as a multi-top coherent tree structure whose leaves are triggered Markov Processes. It is used to capture the failure scenarios of a critical system. To build such a model, the dysfunctional behavior of each component has first to be specified. It must be modeled by a trigger Markov process, that is defined as two Markov chains and two transfer matrices. One Markov chain specifies the dysfunctional behavior of the component in the active mode, and the other specifies its dysfunctional behavior in the inactive mode. A component can be activated or deactivated through the transfer matrices which give a probability distribution from the states of one chain to the states of the other. Figure 2 shows a representation (adapted from Bouissou and Bon (2003)) of a trigger Markov Process that specifies the behavior of a standard BDMP leaf (called SF leaf). λ_S, λ are failure rates and μ is a repair rate. It has been assumed that the failure rates are different in the two modes, and the repair rates are similar.

The system architecture is then translated into a fault tree. Activation and deactivation of a component are provoked by triggers that are defined from an origin node (leaf or logic gate output) to a destination one:

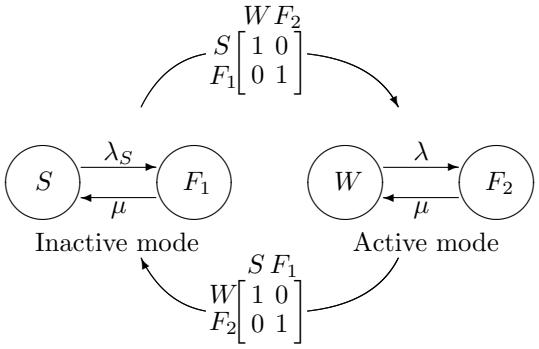

Fig. 2. Trigger Markov Process of the SF leaf

the destination is activated when the origin fails, and deactivated when the origin is repaired. A trigger is mainly used to model standby redundancies from the origin node to the destination one (cf. Carer et al. (2002)). The fault tree structure and the triggers determine the mode of each leaf, and the corresponding trigger Markov process determines its faulty status. Finally, the faulty status of each component is propagated through the fault tree to determine the occurrence of the top event.

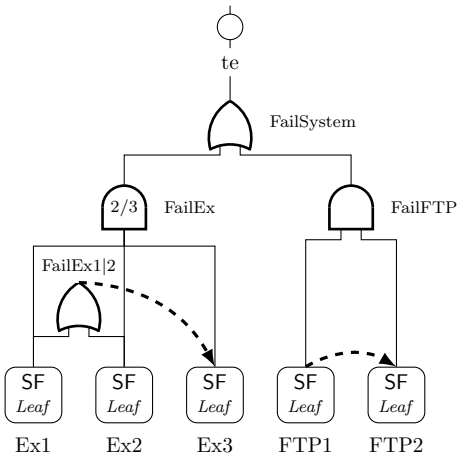

Fig. 3. BDMP model of the CFWS

The BDMP model of the CFWS is shown in Figure 3. It expresses that the system fails if 2 out of 3 extraction pumps or the 2 turbo pumps fail. The dysfunctional behavior of each pump is modeled by a SF leaf. The standby redundancies are modeled by the triggers:

- $Ex3$ is activated when $Ex1$ or $Ex2$ fails and is deactivated when the failed pump is repaired.
- $FTP2$ is activated when $FTP1$ fails and is deactivated when it is repaired.

BDMP has many benefits to perform MBSA. It is a tree-oriented formalism, easy to understand through its graphical representation. Moreover, trigger is an appropriate primitive to model standby redundancies.

BDMP formalism was initially developed for power plant applications, but is also used in other domains like computer security (Pietre-Cambacedes and Bouissou (2010), Kriaa et al. (2012)). Moreover Chaux et al. (2012) proposes a formalization of the BDMP semantics using finite state automata, in order to extract minimal cut sequences.

2.3 Discussion

The BDMP semantics is not sufficient to address all the issue introduced in subsection 2.1. Indeed, the behavior captured in BDMP is constrained by three implicit assumptions in its definition. These assumptions are formulated below.

First, the switching mechanisms handled by triggers refer to a unique redundancy policy: the spare is activated whenever the main is faulty. In other words, the replacement occurs as soon as the main fails, and the resumption occurs as soon as the main is repaired. Moreover, both replacement and resumption are not truly switching mechanisms, because the main is not deactivated. Because of this choice, in particular, the redundancy policies described in subsection 2.1 cannot be translated into the model.

Second, the switching mechanisms handled by triggers cannot be lost. However in reality, these mechanisms cannot be triggered without a "controller" (human operator, electronic devices...). This "controller" can also be faulty, what would prevent the achievement of the mechanisms.

Finally, BDMP gates focus only on the dysfunctional status of their inputs. Then a non-faulty component is always considered as available even if it is inactive. Nevertheless, there are two ways for a component to be unavailable: either it is faulty or it cannot be activated. This assumption is not more discussed in this section, because the difficulties it implies will appear only when the two previous other assumptions are relaxed.

Because of these three assumptions, BDMP does not provide a mean to model relevantly all the possible management policies of standby redundancies. Hence it is not modeled, what leads to approximations in the results of the analysis. This claim is exemplified next.

In a previous paper (Piriou et al. (2014)), the authors tried to consider the trigger's controller in a BDMP model, as an additional condition to achieve the function. In order to recall this approach, let us extend the example introduced in subsection 2.1. As depicted Figure 4, we consider now that standby redundancies are managed by two control functions called R_A and R_B. These functions are each implemented on a couple of active redundant Programmable Logic Controller (PLC), that communicate through two active redundant communication buses. These devices should be integrated in the BDMP model as controllers of the triggers in the BDMP.

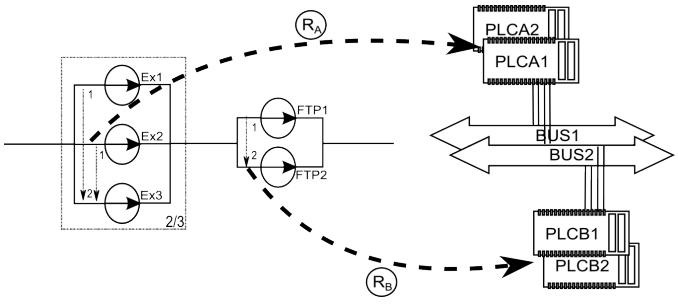

Fig. 4. The controlled CFWS (process on the left, control on the right)

For the considered example, as shown by Figure 5, a group of controlled pumps are considered faulty if the pumps fail or the control function that manage the standby redundancy is lost (because of failures of the control devices necessary to its achievement). The part of BDMP that models the loss of the control functions is darkened. But this modeling provides a too pessimistic knowledge concerning the role of the controller in the redundancy management. Indeed, a control of the trigger is required only when a switching mechanism should occur.

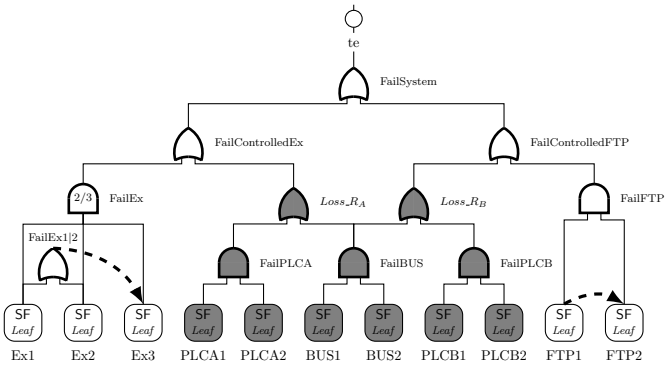

Fig. 5. BDMP model of the controlled CFWS

BDMP formalism is particularly important for MBSA because it is the only tree-based formalism that deals with the replacement, the reparation and the resumption of the components. Nevertheless, the implicit assumptions made in its definition prevent to capture in the model every redundancy policies, and their possible loss. This paper proposes to extend this formalism in order to enrich its modeling capabilities. This extended formalism will permit in particular to take into account the standby redundancy management. Next section proposes an analogy between standby redundancy and preemption, in order to reach this goal.

3. PREEMPTION MECHANISMS FOR MODEL BASED SAFETY ANALYSIS

This section presents the notion of preemption in concurrent systems, and discusses the analogy with standby redundancy in critical systems.

3.1 Preemption in concurrent systems

Concurrent systems deal with processes whose application depends on shared resources. The access of the processes to the resources is managed by coordination mechanisms. Preemption mechanism is one of them. It consists in interrupting a process in a particular context, generally in order to trigger another process. It is a key concept for many applications. For example, it is widely used in real-time programming in order to schedule the execution of processes sharing a unique processor (Baruah et al. (1990)). In a similar way, the consequences of an emergency stop on a distributed reactive system can be described by preemption mechanisms (Andre (1996)). It is also a fundamental concept in traffic management. For example it allows to specify strategies for easing the locomotion of emergency vehicle (Hall et al. (1996)). Let us remark that control is

the common idea behind these applications of preemption mechanisms.

A formal definition and characterization of preemption is proposed in Berry (1993). This work is based on the zero-delay paradigm, which is used by synchronous languages. In this framework, a concurrent system is described as a set of processes that interact through flows of events. A process receives input signals and emits output signals. An input (resp. output) event is a subset of all input (resp. output) signals. So the subset of received inputs signals at a given time is an input event. A process is then define as a mapping from the sequences of input events to the sequences of output events. The zero-delay hypothesis asserts that parallel processes always see the same inputs on the signals they share. Preemption mechanism is then seen as a way to control the right to work to a process, depending on the emitted signals. It distinguishes two kind of preemption (p is a process and s is a signal in what follows):

- abortion: "denying the right to work to a process permanently" (**abort** p **when** s)
- suspension: "denying the right to work to a process temporarily" (**suspend** p **whenever** s **is present**)

Next subsection explains how a standby redundancy can be seen as a preemption mechanism.

3.2 Analogy between standby redundancy and preemption

A standby redundancy manages the activation and deactivation of two parts of the process that perform the same function (called *main* and *spare*). A standby redundancy between non repairable components can easily be seen as an abortion:

- **abort** *main* **when** *main is faulty* **and trigger** *spare*.

However when components are repairable, two switching mechanisms have to be considered: the replacement and the resumption. The replacement can only occur after the failure of the main part, and the resumption can only occur after its reparation. A redundancy policy defines the additional conditions required for applying these switchings. Both replacement and resumption can be applied as soon as possible, i.e. without additional required conditions. But they also can wait for a specific context to be applied. Next it is considered that redundancies focus only on a Boolean dysfunctional status of *main* and *spare*. Let us denote respectively F_M and F_S these Boolean dysfunctional status (F_M (respectively F_S) is $True$ when *main* (respectively *spare*) is faulty). Only four particular policies can then be defined:

- P_{ee} (earliest replacement, earliest resumption): replacement occurs as soon as F_M is $True$, and then resumption occurs as soon as F_M is $False$.
- P_{le} (latest replacement, earliest resumption): replacement occurs when F_M is $True$ and F_S is $False$, and then resumption occurs as soon as F_M is $False$.
- P_{el} (earliest replacement, latest resumption): replacement occurs as soon as F_M is $True$, and then resumption occurs when F_M is $False$ and F_S is $True$.
- P_{ll} (latest replacement, latest resumption): replacement occurs when F_M is $True$ and F_S is $False$, and then resumption occurs when F_M is $False$ and F_S is $True$.

Each of these policies can be seen as a suspension. In what follows, let $s(C_1 \rightarrow C_2)$ denotes the signal that becomes present when the Boolean C_1 becomes $True$ and stays present until the Boolean C_2 becomes $True$. Let us remark that a policy is then exactly specified by the two contexts C_1 and C_2.

- P_{ee}: **suspend** *main* **whenever** $s((F_M = True) \rightarrow (F_M = False))$ **is present and trigger** *spare*.
- P_{le}: **suspend** *main* **whenever** $s((F_M = True) \wedge (F_S = False) \rightarrow (F_M = False))$ **is present and trigger** *spare*.
- P_{el}: **suspend** *main* **whenever** $s((F_M = True) \rightarrow (F_M = False) \wedge (F_S = True))$ **is present and trigger** *spare*.
- P_{ll}: **suspend** *main* **whenever** $s((F_M = True) \wedge (F_S = False) \rightarrow (F_M = False) \wedge (F_S = True))$ **is present and trigger** *spare*.

For the sake of clarity, such redundancy policies can be illustrated with timing diagrams (Figure 6). An arbitrary evolution of F_M and F_S is represented on this diagram. The demanded part of process according to this evolution is also represented for the two most different policies: P_{ee} and P_{ll}. The dashed line expresses that the part of process is demanded but cannot perform the function (because it is faulty). Let us explain this diagram:

(0) Initially, for the two policies, the function is performed by *main*.
(1) A failure of *spare* occurs without consequences.
(2) A failure of *main* occurs: the replacement switching occurs for the policy P_{ee}, and does not occur for the policy P_{ll} (because *spare* is faulty). For the two policies, the function is not achieved.
(3) A reparation of *spare* occurs: the replacement switching occurs for the policy P_{ll} (because *main* is faulty and not *spare*). For the two policies, the function is performed by *spare*.
(4) A reparation of *main* occurs: the resumption switching occurs for the policy P_{ee}, and does not occur for the policy P_{ll} (because *spare* is not faulty, then still able to perform the function). The function is performed by *main* for the policy P_{ee}, and by *spare* for the policy P_{ll}.
(5) A failure of *spare* occurs: the resumption switching occurs for the policy P_{ll} (because *spare* is faulty and not *main*). For the two policies, the function is performed by *main*.
(6) A reparation of *spare* occurs without consequences.

Let us remark that in the considered scenario, the achievement of the expected function does not depend on the policy. Nevertheless, it can even though have a significant impact on safety because the dysfunctional behavior of a component is generally different when it is active or not (such is the case for the SF leaves, see Figure 2: the failure rate changes). Then the activation of one part or another according to particular dysfunctional context can have

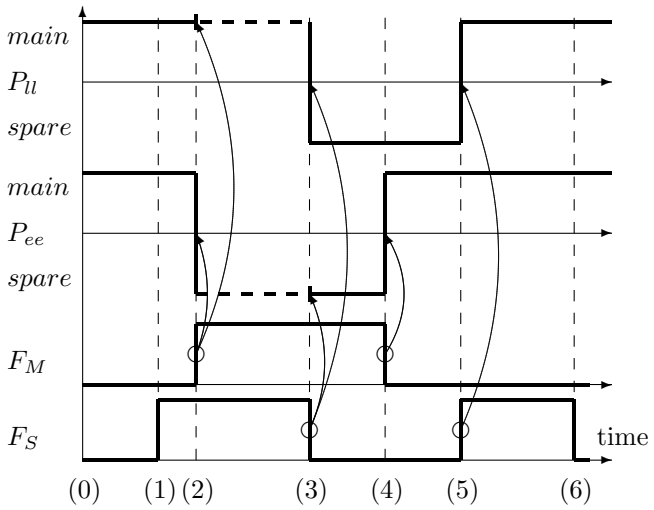

Fig. 6. Example of timing diagrams for illustrating the behavior associated to redundancy policies

different consequences on system safety. More complex redundancy policies can be defined while considering more knowledge on the dysfunctional context (failure modes, mission phases...), but their characterization is out of the scope of this paper.

This study has shown that every redundancy policy can be specified in terms of preemption mechanism. Next section exploits this analogy to extend the definition of the trigger primitive in BDMP.

4. AN EXTENSION OF BDMP FOR MODELING THE REDUNDANCY MANAGEMENT

This section proposes an extension of the BDMP formalism, for integrating the management of standby redundancies according to preemption mechanisms into the safety models.

4.1 Definition of GBDMP

The proposed extension has to solve two modeling issues:

- How to develop the definition of a trigger, to make possible the modeling of any redundancy policies?
- How to make the occurrence of switchings conditional upon the availability of the trigger's controller?

The first point has been discussed in last section: a trigger can be characterized by a preemption mechanism that expresses the redundancy policy. For the second point, we propose to "guard" the triggers by a node of the BDMP. Therefore a switching can be applied by a trigger only if its guard node is not faulty. Moreover, as it was mentioned in subsection 2.3, BDMP gates focus only on the dysfunctional status of their inputs. Then if a trigger fails to activate a non faulty component (because of the failure of its guard), the component is even though considered available. Hence the proposal changes the interpretation of the failure of the gates, to consider that a non demanded component cannot perform its function whatever its dysfunctional status.

Because of the attribution of a guard to the trigger, this extended formalism is called **Guarded** Boolean logic Driven Markov Process (GBDMP). A GBDMP is a 4-tuple $< \mathscr{F}, te, T, (M_i) >$ where (differences with the BDMP formalism are written in bold):

- $\mathscr{F}$ is a multi-top coherent fault-tree, i.e. a 3-tuple $< N, E, k >$ where:
  - $N = G \cup L$, a set of nodes composed of a set of gates and a set of leaves, such as $G \cap L = \varnothing$
  - $E \subset G \times N$, a set of oriented edges such as $< N, E >$ is a directed non-cyclical graph, and $\forall G_i \in G, (\{G_i\} \times N) \cap E \neq \varnothing$.
  - $k : G \longrightarrow \mathbb{N}^*$, a function that determines the kind of the gates. If $k(G_i) = 1$ then G_i is an *OR* gate. If $k(G_i) = Card((\{G_i\} \times N) \cap E)$ then G_i is an *AND* gate.
- $te \in G$ is a particular gate called the top-event.
- T is a set of oriented edges called triggers. A trigger T_i is defined as a 4-tuple $< o_i, g_i, d_i, P_i >$ where:
  - $o_i \in N \backslash \{te\}$ is the origin,
  - $g_i \in N \backslash \{te\}$ **is the guard,**
  - $d_i \in N \backslash \{te\}$ is the destination,
  - P_i **is a policy specified as a preemption mechanism.**
- (M_i) is a set of triggered Markov processes associated to the leaves.

The solicitation of the nodes origin and destination of a trigger is determined by referring to the specified redundancy policy. It is necessary to the activation of a node. Moreover, the activation of any node requires that at least one of its output nodes is active (if their exist). The failure of the leaves are determined by the associated trigger Markov process (see Bouissou and Bon (2003) for the formal definition). The failures of the gates are determined by the state of its inputs: if the number of the input nodes of the gate G_i that are faulty **or not demanded** is greater than $k(G_i)$, then G_i is faulty.

Finally, a switching of solicitation can occurs through a trigger only if its guard is not faulty. Of course the guard can be a never failed leaf (i.e. a leaf whose trigger Markov process does not have any faulty state). Since any policy can be associated to triggers, then in particular it is the case for the policy implicitly associated to BDMP triggers. Hence each BDMP (with BDMP triggers) has an equivalent GBDMP.

In order to prevent inconsistencies, two construction rules are added. The oriented graph $< N, E >$, for which links are added between origin and destination of triggers, must be without cycle. Indeed, the activation and failure of a node cannot depend on themselves. Moreover, each node can be associated to at most one trigger. If this rule is not satisfied, the activation of a node can be associated to different redundancy policies that can be incompatible.

Due to the lack of space, the formal elicitation of the GBDMP semantics will not be described in this paper.

4.2 Modeling with GBDMP

In order to illustrate the benefits of this extension of BDMP formalism, the example introduced in subsection 2.1 is now modeled by a GBDMP.

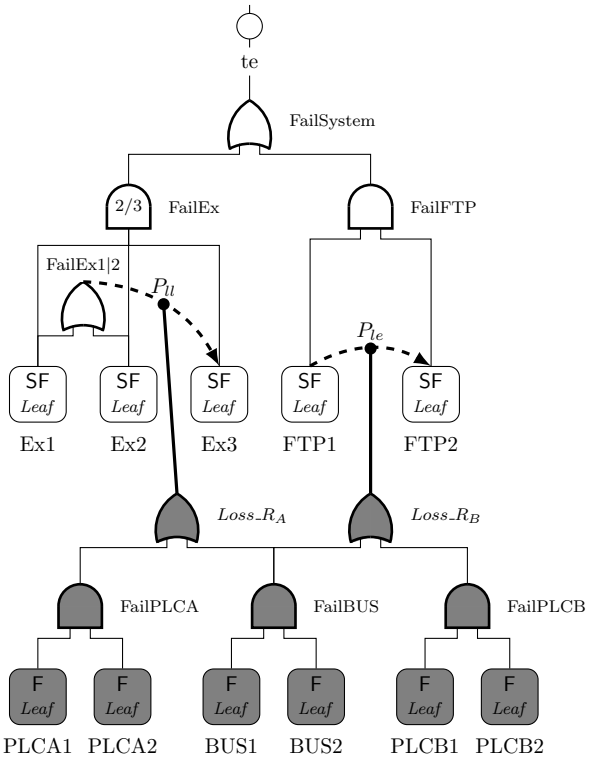

Fig. 7. The GBDMP model of the controlled CFWS

Figure 7 shows the GBDMP model of the controlled CFWS (see Figure 4). Each standby redundancy is modeled by a trigger guarded by the controller that controls it. Thereby the condition for switching between ($Ex1$ or $Ex2$) and $Ex3$ is that R_A is not lost (i.e. the gate $Loss\_R_A$ is not faulty). The condition for switching between $FTP1$ and $FTP2$ is that R_B is not lost (i.e. the gate $Loss\_R_B$ is not faulty). Moreover, a redundancy policy is associated to each trigger, according to the specifications given in the subsection 2.1:

- For the redundancy between the extraction pumps, the replacement must occur when $FailEx1|2$ is faulty and $Ex3$ is not faulty, and the resumption must occur when $Ex3$ is faulty and $FailEx1|2$ is not faulty. This specification correspond to the policy P_{ll} called "latest replacement, latest resumption" in subsection 3.2.
- For the redundancy between the turbo pumps, the replacement must occur when $FTP1$ is faulty and $FTP2$ is not faulty, and the resumption must occur as soon as $FTP1$ is repaired. This specification correspond to the policy P_{le} called "latest replacement, earliest resumption" in subsection 3.2.

4.3 Partial qualitative analysis

Now a comparison of the behavior described by the GBDMP model and the BDMP model (cf. subsection 2.3) is performed through two particular sequences.

First sequence: a sequence to check that the failure of a trigger's guard prevents the occurrence of the corresponding switchings (in particular the resumption):
$0 \overset{f-FTP1}{\longrightarrow} 1 \overset{f-Loss\_R_B}{\longrightarrow} 2 \overset{r-FTP1}{\longrightarrow} 3 \overset{f-FTP2}{\longrightarrow} 4$

For saving space, the event $f - Loss\_R_B$ aggregates all sequences of basic events that lead to lose the function R_B. The expected behavior described by this sequence is reported below:

(0) Initially, the pressurization function is performed by the pump $FTP1$.
(1) After the failure of $FTP1$: the replacement occurs from $FTP1$ to $FTP2$. The function is performed by the pump $FTP2$.
(2) After the loss of R_B: nothing happens.
(3) After the reparation of $FTP1$: the resumption should occur, but it cannot because of the loss of R_B. The function is still performed by the pump $FTP2$.
(4) After the failure of $FTP2$: $FTP1$ is not faulty but the control has failed in activating it. Then the function is not achieved (i.e. the gate $FailFTP$ is faulty).

Table 1 compares the behaviors specified by the BDMP and GBDMP models for this sequence with the expected one. The name of a pump is written in black if the pump is demanded and in gray else, and is struck through if it is faulty. This table confirms the ability of GBDMP to make the occurrence of switchings conditional upon the availability of the trigger's controller.

Table 1. State of FTP pumps during a particular dysfunctional sequence according to the BDMP and GBDMP models, and the reality.

sequence	$0 \overset{f-FTP1}{\longrightarrow} 1$	$\overset{f-Loss\_R_B}{\longrightarrow} 2$	$\overset{r-FTP1}{\longrightarrow} 3$	$\overset{f-FTP2}{\longrightarrow} 4$	
BDMP Fig. 5	FTP1, FTP2	~~FTP1~~, FTP2	~~FTP1~~, FTP2	FTP1, FTP2	FTP1, ~~FTP2~~
GBDMP Fig. 7	FTP1, FTP2	~~FTP1~~, FTP2	~~FTP1~~, FTP2	FTP1, FTP2	FTP1, ~~FTP2~~
expected	FTP1, FTP2	~~FTP1~~, FTP2	~~FTP1~~, FTP2	FTP1, FTP2	FTP1, ~~FTP2~~

Second sequence: a sequence to check that the redundancy policies have been correctly translated into the models:
$0 \overset{f-Ex3}{\longrightarrow} 1 \overset{f-Ex1}{\longrightarrow} 2 \overset{r-Ex3}{\longrightarrow} 3 \overset{r-Ex1}{\longrightarrow} 4 \overset{f-Ex3}{\longrightarrow} 5$
The expected behavior described by this sequence is reported below:

(0) Initially, the extraction function is performed by the pumps $Ex1$ and $Ex2$.
(1) After the failure of $Ex3$: nothing happens.
(2) After the failure of $Ex1$: the replacement does not occur because $Ex3$ is faulty then cannot perform the function anyway. The function is performed only by $Ex2$ (that is not sufficient to achieve it).
(3) After the reparation of $Ex3$: the replacement occurs. The function is performed by $Ex2$ and $Ex3$.
(4) After the reparation of $Ex1$: the resumption does not occur because $Ex3$ is not faulty then can still perform the function. The function is still performed by $Ex2$ and $Ex3$.
(5) After the failure of $Ex3$: the resumption occurs. The function is performed by $Ex1$ and $Ex2$.

Table 2 confirms the ability of GBDMP to capture redundancy policies different than the policy implicitly associated to BDMP trigger.

Table 2. State of Ex pumps during a particular dysfunctional sequence according to the BDMP and GBDMP models, and the reality.

sequence	0	$\xrightarrow{f-Ex3}$ 1	$\xrightarrow{f-Ex1}$ 2	$\xrightarrow{r-Ex3}$ 3	$\xrightarrow{r-Ex1}$ 4	$\xrightarrow{f-Ex3}$ 5
BDMP Fig. 5	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, Ex3	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~
GBDMP Fig 7	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, Ex3	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~
expected	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, ~~Ex3~~	~~Ex1~~, Ex2, Ex3	Ex1, Ex2, Ex3	Ex1, Ex2, ~~Ex3~~

Finally, this brief analysis shows that the GBDMP model complies with the expected behavior contrary to the BDMP model. More generally, this case study illustrates that GBDMP overcomes the BDMP limitations for addressing the issue introduced in subsection 2.1.

5. CONCLUSION

In order to improve the representativeness of BDMP for performing relevant MBSA, this paper questions the ability of this formalism to model any standby redundancy policy, and its possible loss. It appears the BDMP formalism is not adapted for addressing this issue. The implicit assumptions made by the BDMP definition, that explain this limitation, have been explicitly formulated. Hence the Guarded BDMP has been defined as an extension of BDMP formalism to enrich its modeling capabilities. This extension is based on a refinement of BDMP triggers and on a new interpretation of the gates failure. A guard -defined as a sub-tree of the GBDMP- has been added to the triggers. Moreover the switching policy followed by a trigger can be freely specified. The specification is based on an analogy between standby redundancies -defined for repairable critical systems- and preemption mechanisms -defined for concurrent systems-. Finally a case study coming from nuclear power industry has been considered for illustrating the benefits of GBDMP compared to BDMP to address the above-mentioned issue.

A natural perspective of this work is the development of methods to assess qualitative and quantitative safety attributes, based on a GBDMP model. But to apply these formal analysis, the semantics has to be formally described. This task will be addressed in the authors next publication. In parallel to this theoretical work, the authors are developing a prototype software tool to support the approach. This tool aims to implement the GBDMP formalism, a model library, a consistency checker, a graphical discrete event simulator, and two classical formal analysis: the minimal cut sequences extraction, and an availability assessment based on the generation of a Markov chain.

ACKNOWLEDGEMENTS

This work is funded by the French Investment of Future Program: Generic Components of Embedded Software as part of the CONNEXION project.

REFERENCES

Andre, C. (1996). Representation and analysis of reactive behaviors: A synchronous approach. In *Symposium on discrete events and manufacturing systems*, 21 pages. Lille (Paris).

Baruah, S.K., Rosier, L.E., and Howell, R.R. (1990). Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4), pp. 301–324.

Berry, G. (1993). Preemption in concurrent systems. In *13th Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 72–93. Bombay (India).

Bouissou, M. and Bon, J.L. (2003). A new formalism that combines advantages of fault trees and markov models: Boolean logic Driven Markov Processes. *Reliability Engineering and Systems Safety*, 82(2), pp. 149–163.

Carer, P., Bellvis, J., Bouissou, M., Domergue, J., and Pestourie, J. (2002). A new method for reliability assessment of electrical power supplies with standby redundancies. In *7th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS02)*. Napoli (Italy). 6 pages.

Chaux, P., Roussel, J.M., Lesage, J.J., Deleuze, G., and Bouissou, M. (2012). Systematic extraction of minimal cut sequences from a BDMP model. In *21th European Safety & Reliability Conference (ESREL'12)*. Helsinki (Finland). Session 16B, 8 pages.

Hall, T., Schwartz, M., and Hamer, S. (1996). Gps-based traffic control preemption system. US Patent 5,539,398.

Kriaa, S., Bouissou, M., and Pietre-Cambacedes, L. (2012). Modeling the stuxnet attack with BDMP: Towards more formal risk assessments. In *7th International Conference on Risk and Security of Internet and Systems (CRiSIS)*. Cork (Ireland). 8 pages.

Pietre-Cambacedes, L. and Bouissou, M. (2010). Attack and defense modeling with BDMP. In *Computer Network Security*, volume 6258 of *Lecture Notes in Computer Science*, pp. 86–101. Springer.

Piriou, P.Y., Faure, J.M., and Lesage, J.J. (2014). Control-in-the-loop model based safety analysis. In *24th European Safety & Reliability Conference (ESREL'14)*, pp. 655–662. Wroclaw (Poland).