Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A consistency based approach to deal with modeling errors and process failures in D.E.S

Carmen Lopez-Varela[*], Audine Subias[†‡] and Michel Combacau[†‡]
[*]Instituto Tecnologico de Culiacan
Departamento de Ingenieria
Industrial Avenida Juan de Dios Batiz sin numero Colonia Guadalupe Culiacan, Sinaloa, Mexico C.P. 80220,
[†]CNRS; LAAS; 7 avenue du colonel Roche F-31077 Toulouse, France
[‡] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France

***Abstract*—In classical consistency based approaches used for diagnosis of discrete event systems (DES), an inconsistency between the behaviors described in the reference model and the behavior observed through the sensors detects the occurrence of a failure in the system. If the model is not considered as error free, this traditional interpretation of inconsistencies must be enhanced to include the model errors as a potential responsible for the inconsistency. This paper has two objectives. First, it proposes a mechanism able to discriminate among inconsistencies the ones due to modeling errors an those caused by failures. Second, it describes a method to restore the consistency between the models and the observations.**

***Index Terms*—Discrete event systems, Failure, Detection, Diagnosis, Modeling error, Petri Nets**

I. Introduction

For DES, the diagnosis is generally regarded as a set of trajectories which are consistent with the observations. Several model based approaches (1),(2),(3),(4) rely on a discrete event model called "diagnoser" linking the faults and the observations. The "diagnoser" is built from the knowledge of the normal and the faulty behaviors. Thus the basic knowledge required is the set of faults and the observations related to each fault. The diagnoser approaches work under the exhaustiveness assumption of the faulty behaviors of the system. It is a strong hypothesis which limits the applicability of the method, because only the faults reported in the "diagnoser" can be detected. Other model based diagnosis approaches use a model of the normal behavior and tackle the diagnosis problem as a consistency problem between the observations and the model of the system (5),(6),(7). Thus, in these approaches a deviation from the behavior described by the model indicates a fault manifestation. But in this case there are others causes to consider in addition to the system malfunctions: an erroneous observation, an erroneous model. The objective of this paper is to propose a consistency based-approach to detect and to correct the inconsistencies originated by modeling errors. Our motivation is to provide an approach able to discriminate these modeling errors from process failures.

II. Models of the system

A consistency based-approach supposes the use of a reference model (a behavioral model of the system) and the observation of the real behavior of the system.

A. Behavioral model: general statement

In DES, whatever the modeling tool considered, a behavioral model relies on event occurrences and can be be represented by a set of paths defining the set of possible evolutions.

An event e is a physical phenomenon that originates a change of states in a DES. A system can be affected either by observable events or unobservable events. An event e is observable (resp. unobservable), if it is (resp. not) possible to verify its occurrence by means of sensors located in the system In this work, we consider that the observations are error-free (no sensor failure), therefore a signal reception from a sensor indicates the occurrence of an observable event, that means that an observable event is associated to each observation from the system.

A path (or trajectory) TR is composed by a set of states $\{M_1, M_2, \cdots M_n\}$ and the events $\{e_1, e_2, \cdots, e_{n-1}\}$ triggering the evolutions between states: $TR = \{M_1, e_1, M_2, e_2, \cdots, e_{n-1}, M_n\}$

Let us note $TR(X)$ the set of behavior paths described in a model X.

B. Models for consistency based approach

In the proposed work we do not consider only one reference model but a set of reference models. This set is given by a model of the Expected behavior noted MOD_E that represents the process behavior to be observed under normal condition of operation (correct behavior), and by a model of Control noted MOD_C that describes the way in which the process must be used to satisfy the set of requests imposed by the user of the system. These two models are designed off-line.

The observed behavior of the system is captured in a model (noted MOD_{OBS}) built on line with the received observations from the process. This model represents the set of all possible paths leading to a state in which the system can be according to the sequence of observations. At the beginning MOD_{OBS} is constituted by only its initial state which is given by the initial states of the reference models. Each time an observation is received from the system, MOD_{OBS} is updated by considering three possibilities of evolution: the system keeps its current state, the system changes of state and evolves to a state that has been previously reached, the system changes of state and evolves to a state that has never been reached. Of course, from the initial state the second possibility is not pertinent. The updating of MOD_{OBS} is then performed through the addition of hypothetical transitions (from and to the current state, from the current state to an existing state) or through the addition of a new state and of a transition from the current state to this new state. Considering these three hypothetical evolutions allows us first to introduce in the model an uncertainty on the state reached after an observation and second, to represent in an exhaustive way all the possible evolutions of the real system. As we will see later, it is after checking the consistency between the observation model and the reference models that MOD_{OBS} is updated by deleting the hypothetical transitions introduced that do not correspond to a behavior of the reference models and by determining the new states.

If the models were error-free, the information (behavior paths) contained in the different models of a system should be consistent. The sequence of actions to be executed in order to fulfill the user requirements being described in MOD_C, the corresponding behaviors should be modeled in MOD_E. The next section addresses the problem of inconsistency detection.

III. Inconsistency detection

A. Problem position

An inconsistency between two models U and V for example, with respect to a path TR =$\{M_1$, e_1, M_2, $e_2, \ldots, e_{n-1}$, $M_n\}$ appears when the underlying sequence of events defined by $S_{ev} = (e_1, e_2, \ldots, e_{n-1})$ is recognized by only one model.

The path can be decomposed in a prefix noted TR_1 and a suffix TR_2, such as $TR_1 = \{M_1, e_1, M_2, e_2, \ldots, e_{j-1}, M_j\}$ is a consistent path with two models and that $TR_2 = \{M_j, e_j, M_{j+1}, e_{j+1}, \ldots, e_{n-1}, M_n\}$ is only consistent with one model. This highlights that, TR_1 is a path belonging to both models (U and V), whereas TR_2 only belongs to one of them, (U or V). The decomposition of a path TR in TR_1 and TR_2 having the properties above described, characterizes the intersection between the sets of paths of two models U and V. If $TR(U)$ is the set of paths contained in the model U and that TR_2 is not recognized by V (without loss of generality), then $TR_1 \in TR(U) \cap TR(V)$ and $TR_2 \in TR(U) \backslash (TR(U) \cap TR(V))$.

B. Inconsistency detection mechanism

In our approach, an inconsistency detection is based on the verification of the consistency between the observations resulting from the system operation and the models describing the normal behavior of the system. That means that the paths modeling the observed sequence of events and captured in MOD_{OBS} are compared to the paths described in MOD_E to check their consistency. This consistency verification is realized by synchronization of behaviors. When an event is received, if it is possible to synchronize the observed behavior (MOD_{OBS}) with an evolution of the reference model MOD_E this means that the reference model MOD_E is consistent with the observed behavior. In the other case, the reference model is claimed inconsistent. Then, a similar comparison is done between MOD_{OBS} and MOD_C. Thus, the consistency of (MOD_E and MOD_C) is indirectly checked. This verification step relies on a classical construction of cartesian product of the models. In this product only the transitions labeled with the event received and corresponding to a possible evolution in the two models (MOD_{OBS} and MOD_E or, MOD_{OBS} and MOD_C) are significant as they traduce the synchronized behaviors. The verification is performed each time an observable event is received. According to the previous subsection, the transition non accepted for both models is then given by $TR_2 = (M_{n-1}, e_{n-1}, M_n)$ and the last event e_{n-1} is responsible for the inconsistency. So, as for each observation three hypothetical evolutions are considered in MOD_{OBS} (II-B) the consistency is checked for the three hypothetical corresponding transitions. When the consistency is verified via one of these transitions, the given transition is kept in the observation model while the other hypothetical transitions not confirmed are deleted. In this way, we kept in MOD_OBS only the behaviors consistent with the models and MOD_OBS is ready to integrate the next observation. Moreover the states labeled unknown in the observation model are instantiated by the current state of the model with which the consistency is first verified. If any of the hypothetical evolutions of MOD_OBS can be synchronized with the evolutions of the checked model, an inconsistency is detected.

In fact, at the end of this verification steps an inconsistency can be detected with only one of the reference models (MOD_E or MOD_C) or with both models (MOD_E and MOD_C). Whatever the result of the verification is, as we consider the possibility of modeling errors the two models can be incriminated. Indeed, each inconsistency can always be explained by a lacking path representing a normal behavior in a model or by existing paths representing absurd or abnormal behaviors in the other model. Moreover, each inconsistency can also be explained by a process failure. In the next section, is presented our approach to characterize an inconsistency in order to, first, discriminate a modeling error and a process failure and second, in case or modeling error to not systematically incriminate the two reference models.

IV. Inconsistency characterization

A. Inconsistency origin: error modeling vs failure

Our proposal is to consider a new model used collectively with the reference models. This model called model of Real behavior noted MOD_R is supposed to describe in a correct and complete way all normal behaviors of the system. In several works on monitoring and diagnosis this model is supposed known and implicitly the hypothesis that this model is correct and completed is done. In our approach, this model will be suspected as others models, as we consider the possibility of modeling errors. This questioning of the model of real behavior which is never done in classical consistency based approaches constitutes one originality of the presented work.

We first make the assumption that this model MOD_R is error free. In this case until the observed behavior is consistent with MOD_R an inconsistency between the observed behavior (MOD_{OBS}) and the reference models (MOD_E and MOD_C) is explained by a modeling error in the reference models, and the consistency is restored by the modification of the inconsistent model(s). When the consistency between the observed behavior and MOD_R is broken, as in classical consistency based approaches of diagnosis, we suspect the process and launch a diagnosis on the process. In case of success of this diagnosis step, the process is repaired and/or reconfigured. In case of the process diagnosis fails, we leave our hypothesis on the model of real behavior and we consider that the only explication of the inconsistency origin is a modeling error in this model MOD_R. The consistency is then restored by the models modification.

It clearly appears that the use of this third model MOD_R allows us to identify an inconsistency origin: modeling error or process failure. We present now how in case of modeling error, MOD_R allows to not systematically incriminate the two models (MOD_E and MOD_C) when an inconsistency is detected.

B. Inconsistency and configurations of the models

An inconsistency corresponds to a discrepancy characterized by a path in the model of the observed behavior that has no corresponding path in another model used for reference. Each inconsistency can then be characterized by the way the sets of behavior paths described in the models (MOD_E, MOD_C and MOD_R) intersect, what we call the **configuration of the models**. One discrepancy between the sets of behaviors paths ($TR(E)$, $TR(R)$ and $TR(C)$) can in fact be induced by several configurations of the models. To make this concept of configuration more understandable, we first explain it by considering only two models and we will extend the results to three models after.

If we consider the two models (MOD_E and MOD_C) there are eight possible configurations. To obtain these eight configurations is it necessary to consider the set of paths E contained in the model MOD_E that are not contained in MOD_C ($TR(E)\backslash TR(C)$), the set of paths C contained in the model MOD_C that are not contained in MOD_E ($TR(C)\backslash TR(E)$) and the set of paths EC that are contained in both models. By associating a boolean variable to each of these sets, a configuration is defined by the triplet (E, C, EC). By considering the existence of each of these sets in the configuration, we obtain the 2^3 possible configurations. Each set is considered as an element that appears or not in the configuration. From these 8 configurations only 5 are significant. Indeed, in our approach the two models (MOD_E and MOD_C) are used for reference and then exist. Therefore, the configurations traducing the existence of only one model must be discarded. Moreover the empty configuration has no sense. Figure 1) shows these configurations with the associated elements E, C or EC and the associated positions of $TR(E)$ and $TR(C)$.

Whatever the configuration of the models is, the discrepancy between the models can be explained by a lack in one model

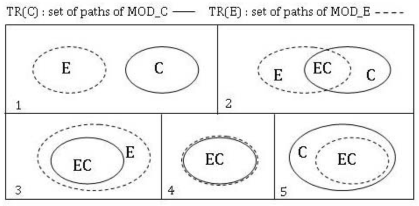

Fig. 1. Possible configurations of MOD_E and MOD_C

but also by additional behavior in the other model. Indeed, let us suppose for instance that an observation is consistent with a behavior path of MOD_E and inconsistent with all the behavior paths of MOD_C. That means that this observation belongs to a path that is in $TR(E)$ but not in $TR(C)$ what is possible with the configurations 1,2 and 3 of Figure 1 where the set E exists.

If we consider now the three models (MOD_E, MOD_C and MOD_R) and by extending the previous reasoning performed on two models, 109 possible configurations of the models can be pointed out. These 109 configurations are obtained by considering that for 3 models 7 subsets exist (E,C,R,EC,ER,CR,ECR). From these 7 subsets 128 (2^7) configurations can in fact be generated, but as the three models must appear in each configuration and as the empty configuration as no meaning in our reasoning, only 109 configurations must be considered.

Until the observed behavior is consistent with MOD_R, when an inconsistency is detected with only one of the reference models (MOD_E or MOD_C) only the model inconsistent with MOD_R is pointed out and the inconsistency is explained by a modeling error in this model only. In this way, MOD_R allows us to not systematically incriminate the two models (MOD_E and (MOD_C). The problem is now to identify the configurations of the three models.

C. Identification of the configurations

The configuration of the models can be identified according to the information provided by each observation. We define a recording REC as a characterization of the consistency of an observation (issued from an observed path TR_x) in relation of the three models MOD_E, MOD_R and MOD_C. To give a formal presentation of this concept of recording we define ($CONS_Y$) a Boolean function that represents the consistency (or the inconsistency) with the model Y :

$$\Omega \xrightarrow{CONS_Y} B = \{0, 1\} \tag{1}$$

with $CONS_Y(TR_x) \quad = 1 if(TR_x \in Y)$ The recording locates the observation issued from TR_x in relation of the models of expected behavior (E), of real behavior (R) and of control (C) and is defined by the vector Boolean function by

$$\Omega \xrightarrow{REC_{ERC}} B^3 \tag{2}$$

where $REC_{ERC}(TR_x) = (CONS_E(TR_x), CONS_R(TR_x), CONS_C(TR_x))$

At last, we can have 7 possibilities (the recording (0,0,0) is not significant) of recordings for an observation. For example, the recording $REC_{ERC}(TR_x) = (0,1,1)$ can be interpreted like an observation (issued from TR_x) inconsistent with MOD_E, consistent with MOD_R and consistent with MOD_C. In fact, each observation provides information on the different subsets conforming the configuration of the models. $REC_{ERC}(TR_x) = (0,1,1)$ points out the subsets R,C,RC that means that all the configurations of models in which these subsets exist can be considered in regards to this observation. It can be noticed that the knowledge given by a set of recordings is independent from the order of the recordings.

To identify the configurations of the three models we propose to build off-line a graph in which all the information about the configurations are stored. This graph includes all the possible configurations of the models. On-line, by following the edges of the graph according to the recording obtained each time an observation is received, it is possible to identify the real configurations of the models. The nodes of the graph are labeled by the n-uplets composed from the subsets conforming the configuration of the models and the transitions of the graph are labeled with the recordings. This graph has 7 levels corresponding to the 7 possible subsets (E,C,R,EC,ER,CR,ECR). At a level n, the number of nodes is given by the number of combination of n recordings through the 7 possible as the order of recordings is not pertinent (Level 1, $C_7^1 = 3$ nodes, Level 2, $C_7^2 = 21$ nodes etc...). We obtain a graph with 128 nodes describing 7! sequences of recordings. All the trajectories in this graph converge to a terminal node pointed out the configurations where the seven subsets exist.

Each time a new observation is received, the associated recording provides information to treat a possible inconsistency. Moreover, the associated recording allows to progress in the graph and so to increase the knowledge on the configuration of the models. This is helpful to modify in a suitable way the models in order to restore the consistency. The next section presents some results about the consistency restoring.

V. Restoring the consistency

Our proposal is to restore the consistency by modifying the models such as they represent the observed behavior. The modifications are then realized on the models incriminated. To restore the consistency between an observed behavior and a model it is necessary to include the observed behavior in this model.

A. Mechanisms of inconsistencies correction

The modifications realized on models respect the initial properties of the models if these properties are compatible with the observed behavior. Moreover, the modifications do not affect the part of the model consistent with the passed observed behavior of the process.

Until now we have considered models in a general formalism based on states, events and paths. Nevertheless, the mechanisms of inconsistencies correction depend on a selected formalism. In the remainder, we present the mechanisms developed under interpreted Petri Nets (PN) formalism. In this case, the model properties which must be considered are place and/or transition invariants, vivacity, boundary. Moreover, an inconsistency is detected through the impossibility of firing the transition associated to the current observed event. Thus, the aim of the modifications is to ensure the firing of the incriminated transition. Three modification methods are proposed:

- *Modification of incidence matrix coefficients* The basic idea is to modify the arrow weights in the PN model, i.e. incidence matrix (noted C) coefficients, without modifying the dimension of this matrix (number of columns and rows). Indeed, if an inconsistency has been detected, that means that the transition associated to the observed event cannot be triggered as it is not enabled by the current marking. It is then necessary in order to enable this transition to replace its input places by a set of places with a suitable marking. This induces to modify the input arrows of the transition i.e the input incidence matrix (Pre).
- *Modification of incidence matrix dimension* This modification focusses on the number of rows and columns of the incidence matrix (C). This type of modification consists in integrating or deleting places and/or transitions until the model represents the observed behavior.
- *Modification of the marking* The objective is to only modify the current marking of the PN to enable the transition associated to the observed event and thus to modeled the observed behavior previously stamped as being inconsistent.

Whatever the modification method is, it is possible to obtain several modified incidence matrices restoring the consistency and called **Consistency Restoring Matrix** (C_{RC}). The C_{RC} matrix expresses the introduction of new behaviors in the model. Thus, the modified PN model may have new properties. In this article, only the first modification method is detailed in the next section.

B. Modification of incidence matrix coefficients

As said previously, an inconsistency corresponds to the impossibility to fire a transition t_{inc} in a PN model. By changing the input places of t_{inc} by a set of marked places (p_j) such that for the current marking M: $M(p_j) \geq Pre(p_j, t_{inc})$ it is possible to enable the transition. This modification must not change the initial properties of the PN model which are the place invariants and the transition invariants.

A place invariant is a linear function of marking where the constant value depends on the initial marking of the PN. The set of places implicated in this relation is given by $Ps_r = \{p \in P \backslash V_{Pr}(p) \neq 0\}$ with P the set of places of the PN, and V_{Pr} a P-flow i.e. an integer $|P|$-component weight column vector solution of $(V_{Pr})^T \bullet C = 0$. If V_{Pr} is a natural solution, V_{Pr} is a P-semiflow and the place invariant is said positive.

A transition invariant is given by a sequence of firing noted σ_r, that does not modify the marking of the PN. The set of transitions implied in the invariant, is defined by $Ts_r = \{t \in T \backslash V_{Tr}(t) \neq 0\}$ with T the set of transitions of the PN and V_{Tr} a T-flow i.e. an integer $|T|$-component weight column vector solution of $C \bullet V_{Tr} = 0$. If V_{Tr} is a natural solution, V_{Tr} is a T-semiflow and can be interpreted as the characteristic vector of the sequence σ_r.

The different steps of the modification method are listed below:

Step 1: For the current marking (M_k) of the PN model to modify (MOD_E and/or MOD_C), determine PM_k the set of marked places. This set is supposed to not be empty.

Step 2: Determine the set of possible input places for the transition t_{inc}, at the origin of the inconsistency detection. This set noted M_{poss} is the set of possible modifications: $M_{poss} = \{Ep_i | i = 1..2^q\}$, with Ep_i a subset of PM_k and q the number of places contained in PM_k. This step ensures that the only invariants that will be modified are the ones including the transition t_{inc} blamed during the inconsistency detection step. Indeed, the aim of the modification method is to conserve only the invariants corresponding to consistent behaviors and to modify the others.

Step 3: Build the set of possible input place vectors Pre_{poss} of the transition t_{inc}:

$Pre_{poss} = \{Pre_i(., t_{inc}) \backslash Pre_i(., t_{inc}) = [x_1, .., x_n]^T\}$

with n the number of places of the PN, $Pre_i(., t_{inc})$ the column of the input incidence matrix Pre associated to t_{inc} and to the subset of places Ep_i. $\forall p \in P, Pre_i(p, t_{inc}) = x_j = 1 \quad \text{if} \quad p \in Ep_i, \quad x_j = 0 \;\text{ if } \; p \notin Ep_i$.

Step 4: Construct the set of incidence vectors C_{poss} associated to the transition t_{inc}.
$C_{poss} = \{C_i(., t_{inc}) \mid C_i(., t_{inc}) = Post(., t_{inc}) - Pre_i(., t_{inc})\}$. $Post(., t_{inc})$ is the column of the output incidence matrix $Post$ associated to t_{inc}.

Step 5: Determine EI_r the set of elements of C_{poss} that keeps the positive place invariant of the original PN, characterized by V_{Pr}. $EI_r = \left\{C_i(., t_{inc}) | V_{Pr}^T.C_i(., t_{inc}) = 0\right\}$

Step 6:Determine E_{sol} the set of elements of C_{poss} keeping the positive place invariants of the initial PN. $E_{sol} = \{C_i(., t_{inc}) | C_i(., t_{inc}) = (EI_1 \cap EI_2 \cap EI_2 \ldots \cap EI_m)\}$ with m the number of positive place invariants of the initial model. The solution $E_{sol} = \emptyset$, indicates that there is no modification that conserves all the positive place invariants. A solution with more than one element in the set E_{sol}, means that it exists more than one modification.

Step 7: Replace the old column vector of the incidence matrix C associated to the transition t_{inc} by one element of E_{sol} i.e. $C(., t_{inc}) = C_i(., t_{inc}) \quad \forall C_i(., t_{inc}) \in E_{sol}$ The incidence matrix obtained after this step is the Consistency Restoring Matrix C_{RC}. The number of matrices determined depends on the number of elements in E_{sol}.

Step 8: Construct the new PN model defined by each matrix C_{RC}.

Let us see now, the application of this modification method on a pedagogic example.

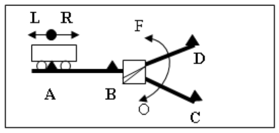

Fig. 2. Pedagogic example

VI. Application of the inconsistency correction method

Let us consider the system presented in (Fig. 2) with the interpreted Petri net model of the expected behavior PN_E and MOD_E (Fig. 3) the subjacent model of the expected behavior given by the marking graph. The initial marking of PN_E is $M_0 = [10000]^T$. Let us suppose that the control applied to the process generates the event b in the process. Let us suppose also that after the verification step (not detailed here), the observation of b is considered consistent with the models used i.e. MOD_E, MOD_R and MOD_C. The recording associated is $REC_{ERC}(TR_b) = (1, 1, 1)$ and there is no inconsistency detection. The evolution of the model (MOD_E)by the firing of the transition labeled b leads to the new marking $M_k = [01000]^T$. Now, let us suppose a second control producing the event a with the associated recording $REC_{ERC}(TR_a) = (0, 1, 1)$. This recording points out a modeling error in MOD_E, because the observation is consistent with MOD_R indicating thus the normality of the observed process behavior. As the observation is consistent with MOD_C, this last model is not incriminated and thus it will not be modified. Once, an inconsistency is detected to restore the consistency the model of expected behavior PN_E is modified according the method previously described.

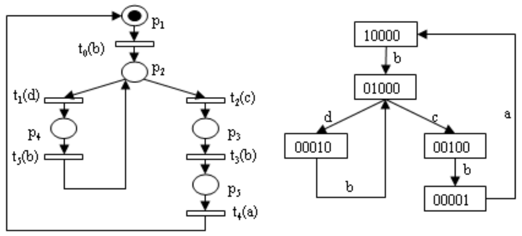

Fig. 3. PN_E and MOD_E

Step 1: The inconsistent transition is $t_{inc} = t_4$ in PN_E; the current marking is $M_k = [01000]^T$; the set of marked places is $PM_k = \{p2\}$.

Step 2: The set of possible input places of t_4 is $M_{poss} = \{\emptyset, \{p2\}\} = \{Ep_1, Ep_2\}$. The empty set is not a solution for our problem, thus the only solution is $p2$.

Step 3: The set of possible input place vectors of t_4 is $Pre_{poss} = Pre_2(., t_4) = [01000]^T$.

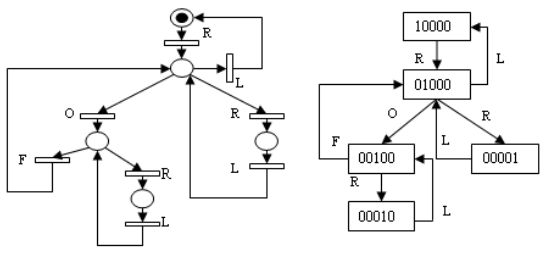

Fig. 4. PN_C and MOD_C

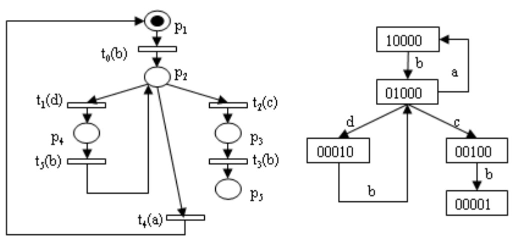

Fig. 5. PN_E et MOD_E after modification

Step 4: The set C_{poss} of incidence vectors is $C_{poss} = \{C_2(.,t_4) = Post(.,t_4) - Pre_2(.,t_4) = [10000]^T - [01000]^T = [1(-1)000]^T\}$.

Steps 5 and 6: The Petri net of the expected behavior PN_E has only one positive place invariant: $V_{P1} = [11111]^T$ with $V_{P1}^T \bullet C_2(.,t_4) = 0$. That means that the modification conserves the positive place invariant of the initial model (PN_E). The set of solutions keeping the place invariant is given by $E_{sol} = EI_1 = \{C_2(.,t_4)\}$

Steps 7 and 8: The Consistency Restoring Matrix C_{RC} including the modifications is given by:

$$C_{RC} = [C(.,t_0)C(.,t_1)C(.,t_2)C(.,t_3)C_2(.,t_4)C(.,t_5)]$$

The initial PN has two T-semiflows that correspond to the firing sequences $\sigma_1 = t_0t_2t_3t_4$ and $\sigma_2 = t_1t_5$ with respectively $V_{T1} = [101110]^T$ and $V_{T2} = [010001]^T$. The sequence σ_2 (which does not contain the incriminated transition) is conserved in the modified PN (see Fig. 5) ($C_{RC} \bullet V_{T1} = [101110]^T \neq 0$ et $C_{RC} \bullet V_{T2} = [010001]^T = 0$). But the modification creates a new transition invariant given by the firing sequence $\sigma_3 = t_0t_4$.

The new PN_E defined by C_{RC} and its reachable marking graph representing MOD_E are displayed in (Fig. 4).The observed behavior is now represented in MOD_E. With the introduction of this behavior in the model a new firing sequence has been created ($\sigma = t_0t_2t_3$). This sequence is isolated because it leads to a blockage situation. It is possible that the behavior associated to σ represents a modeling error. Note that besides the determination of the inconsistency origin i.e. modeling error, the recordings $< (1,1,1),(0,1,1) >$ are used to progress online in the identification tree and then to identify the possible configurations for the three models. Of course, by using only two observations the identification will leads to a set of configurations for the three models. Nevertheless, this identification is pertinent in order to adapt the models modifications. This part of the work is not detailed in this article.

VII. Conclusion

We have presented an approach for detection and identification of inconsistencies in D.E.S with the particularity first to consider both failures and modeling errors, second to use a consistency based approach. Currently, most of researches in the field of diagnosis of DES consider only failures and moreover are based on faults models. The problem of the discrimination between modeling errors and process failure is addressed through the concept of configurations of the models and some results about consistency restoring have been exposed. It can be outlined that currently the proposed approach modifies the models in case of modeling errors but also in case of process failure when the diagnosis process fails. This is this because in this approach the objective is to restore the consistency. Nevertheless, one extension of this work is to consider an approach less systematic for the model modification.

References

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. C. Teneketzis, *Diagnosability of discrete-event systems* IEEE Transactions on Automatic Control, vol. 40, No. 9, pp1555–1575, 1995.

[2] R. Debouk, S. Lafortune and D. Teneketzis, *Coordinates decentralized protocols for failure diagnosis of discrete event systems*, Proceedings of the 37th IEEE Conference on Decision and Control, vol. 10-11, Tampa, Florida, USA, pp 3763–3768, 1998.

[3] S. Genc and S. Lafortune, *Distributed diagnosis of discrete event systems using Petri nets*, In Application and Theory of Petri Nets, Series Lecture Notes in Computer Science, vol. 2679, pp 316–336, 2003.

[4] P. Baroni, G. Lamperti, Pogliano and M. Zanella, *Diagnosis of a class of distributed discrete event systems*, IEEE Transaction on Systems, Man, and Cybernetics, Part A, vol. 30, New York, USA, 731–752, 2000.

[5] J. Ashley and L. Holloway, *Diagnosis of conditions systems using diagnosis causal network*, IEEE International Conference on Systems, Man and Cybernetics, Tucson, USA, pp 17–22, 2001.

[6] S. Soldani, M. Combacau, J. Thomas and A. Subias, *Intermittent fault detection through message exchanges, a coherence based approach*, 17th International Workshop on Principles of diagnosis DX - 06, Burgos, Spain, pp 251–256, 2006.

[7] I. Tabakow, *Using place invariants and test point placement to isolate faults in discrete event systems*, Journal of Universal Computer Science, Springer,vol.13, No. 2, pp 224–243, 2007.