Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 963756, 13 pages
http://dx.doi.org/10.1155/2015/963756

*Research Article*

# Systematic Process for Building a Fault Diagnoser Based on Petri Nets Applied to a Helicopter

**Miguel A. Trigos,[1,2] Antonio Barrientos,[1] and Jaime del Cerro[1]**

[1]*Centro de Automática y Robótica, Universidad Politécnica de Madrid-Consejo Superior de Investigaciones Científicas, c/José Gutiérrez Abascal 2, 28006 Madrid, Spain*
[2]*Facultad de Ingeniería Mecatrónica, Universidad Santo Tomás, Carrera 18, No. 9-27, 680001 Bucaramanga, Colombia*

Correspondence should be addressed to Miguel A. Trigos; matrigos@industriales.upm.es

Received 19 June 2015; Accepted 7 September 2015

Academic Editor: Bin Jiang



This work presents a systematic process for building a Fault Diagnoser (FD), based on Petri Nets (PNs) which has been applied to a small helicopter. This novel tool is able to detect both intermittent and permanent faults. The work carried out is discussed from theoretical and practical point of view. The procedure begins with a division of the whole system into subsystems, which are the devices that have to be modeled by using PN, considering both the normal and fault operations. Subsequently, the models are integrated into a global Petri Net diagnoser (PND) that is able to monitor a whole helicopter and show critical variables to the operator in order to determine the UAV health, preventing accidents in this manner. A Data Acquisition System (DAQ) has been designed for collecting data during the flights and feeding PN diagnoser with them. Several real flights (nominal or under failure) have been carried out to perform the diagnoser setup and verify its performance. A summary of the validation results obtained during real flight tests is also included. An extensive use of this tool will improve preventive maintenance protocols for UAVs (especially helicopters) and allow establishing recommendations in regulations.

## 1. Introduction

The growth in number and complexity of Unmanned Aerial Systems (UAS) missions within civil frameworks is significant. Nevertheless, there is an enormous gap between large and expensive military systems that are able to board redundant equipment and complete health monitoring systems and small, light, and inexpensive commercial aircrafts. Thus, although there are several initiatives aiming at including the UAVs in the nonsegregated air space by using a unique Aerial Traffic Management (ATM) system, these approaches are not affordable for small commercial units. For this reason, during the last years, different organizations [1] have contributed with outstanding efforts to create a necessary regulatory framework that allows integrating small UAS within everyday activities. According to these efforts, some researchers [2] involved in UAS operations are concerned not only about the frequency and incidence of accidents caused by UAS failures but also about of surveillance and safety of the population. Moreover, few attempts have been made in Europe for identifying their causes, including them into the regulation to define and standardize the foundations and routines for the use of UAS in order to decrease the percentages of failure. In addition, according to the report about data reliability of UAVs in military field of the United States [3], UAVs are highly vulnerable to unforeseen situations of the equipment devices (control station and aircraft) and operating environment where they are tested.

Although multirotor systems (e.g., quadrotors and hexrotors) have recently broken into the market in the category of mini-UAVs due to their simplicity, helicopters are still the most frequent option when hovering maneuvers are required in the mission, mainly due to their higher payload capability with respect to the multirotor systems.

Remotely piloted helicopters are inherently unstable with fast dynamics. They do not have the graceful degradation

properties of fixed-wing aircrafts or airships in case of failures. Even when stability augmentation devices are used, a skilled pilot is required to control them during flight. Therefore, a helicopter (HC) has been used for this research, considering that helicopters are the most complex platform from the mechanical point of view. Hence, a failure in any part of the autonomous helicopter (i.e., sensors, actuators, or control system) can lead to a dangerous situation.

The motivation behind our work is to collect data about the behavior of the helicopter flying in normal and fault conditions, which will provide the foundations for developing strategies of FDI that will allow to prevent the aircraft from accidents. Thus, the main contribution of this work is to report how to develop an FD tool based on PN, capable of detecting permanent and intermittent faults. Moreover, the procedure presented in this work illustrates how the thresholds of each variable have been established, for both normal and fault conditions.

Although initially the tool has been used for FD in UAS, in the near future, it can be applied to any kind of system. FDI techniques have been widely applied in process industry [4, 5]. Some approaches can also be found in the field of UAS to detect faults in sensors and actuators. Thus, Fault Tolerant Control (FTC) techniques or Fault Detection Isolation and Recovery (FDIR) techniques are required if a fault is detected. In these cases, the structure of the controller has to be changed to get the best possible response of the system, since the system cannot be stopped. FD techniques applied to UAS are commonly based on analytical model approaches, signal processing based approaches, and knowledge based approaches [6]

Analytical model-based FD approaches make use of mathematical models. Moreover, helicopters exhibit complex dynamics, which makes it difficult to create high fidelity models able to detect small deviations or abnormal functioning. Nevertheless, fault isolation observers for square and nonsquare linear systems can be found in the literature [7]. They provide the observer with a design that ensures the stability and minimizes the influence of disturbances on the residuals at the same time. Other approaches such as the Unknown Input Observer (UIO) designed by Liu et al. [8] aiming at tracking actuator fault parameters and decoupling the effect of faults and unknown inputs are valuable contributions. Moreover, Qi et al. [9, 10] propose states and parameters combined estimation based on square-root Unscented Kalman Filter (UKF) and Kalman Filter- (KF-) based adaptive UKF with a full nonlinear model of an unmanned helicopter (UH). The KF-based adaptive UKF is composed of two parallel master-slave filters. The KF-based adaptive UKF is simpler and more effective than UKF estimation method.

Signal processing based methods are developed on the basis of thorough analyses of the failure mechanisms to determine signal characteristics that can represent failures. A novel wavelet-based approach to detect an abrupt sensor fault in a UH system is presented in [11], accurately localizing the characteristics of a signal (time and frequency domains) and the occurring instants of abnormal status of a sensor in the output signal, whereas an experimental validation of wavelet-based analytical redundancy technique on a 3-degree-of-freedom UH platform is presented in [12]; the technique successfully detects a fault of small magnitude, consisting of a 10% reduction in the pitch sensor gain. Furthermore, a technique for detecting the faults by measuring the rate of change of data with respect to time is applied in [13].

On the other hand, knowledge based FD approaches require a deep cognition of the entire process, acquired typically during the operation of the system. Thus, the knowledge allows avoiding the reliance on accurate mathematical models. For instance, an adaptive threshold Neural Network scheme for UH sensor failure diagnosis is presented in [14], the adaptive threshold approach eliminates the need for thresholds changing with flight conditions.

Previous works as [15], applied to fault diagnosis of Discrete Events Systems (DES), use Regular Languages, State Graphs, or Finite State Machines (FSMs) for modeling the knowledge. These techniques highlight a main drawback, which is the combinational explosion that makes their elaboration difficult when the number of the components increases. Another approach [16] applies the concepts of marking and justification. This allows performing diagnosis without performing an exhaustive enumeration of the states. An evolution of this work can be found in [17], where interpreted Petri Nets are used to model the system, taking advantage of the power of the PNs by making a better use of their mathematical basis. A generalization of the problem that uses a centralized diagnosis solution for diagnosing a distributed system can be found in [18]. This method known as Diagnoser Approach splits the net into modules that may share places, which are called "bordered places." A communication system connects the different modules and updates the diagnosis information.

Considering fault diagnosis applied to Hybrid Systems, a hybrid integration of discrete and continuous FD techniques can be found in [19], where hybrid automata, timed Petri Nets, fault trees, and signal processing techniques are integrated in a single tool.

Petri Nets are a well-known method for implementing Fault Diagnosers due to their capability to manage concurrence. They allow representing either small or large systems. A large number of references can be found in the literature [20–22].

Among the different types of PNs, interpreted PN is the most adequate to represent the model of a system [23, 24].

This work is a useful guide to close the gap between the theory and a real application, and this is one of the most important contributions of the work, since it shows a complete methodology to build a diagnoser of a complex system based on interpreted PN. Moreover, the work includes real data validation in UAV applications, which has not been previously found in the literature.

The present Petri Net diagnoser is able to detect not only permanent but also intermittent failures avoiding combinational explosion problems. Furthermore, flexible architecture of the PND allows including additional devices to the model. The tool can be used in different areas of engineering (industrial process, Robotics, etc.).

Section 2 summarizes the basis of Petri Nets, necessary for understanding the models in the process of fault

diagnosis. Section 3 describes the methodology for building the model and the diagnoser, starting with the models of the components for each subsystem and ending with a unique PN as diagnoser. Section 4 shows the results when applying the diagnoser to a radio control helicopter considering both normal and fault operations. Finally, the conclusions of the work are presented in Section 5, analyzing different options for future work.

## 2. Petri Nets

Petri Nets (PNs) are a graphical and mathematical modeling tool that has been widely applied, especially in the field of automation. They allow describing concurrent, parallel, asynchronous, distributed, and not deterministic systems by using marks to simulate the dynamics and concurrency activities. From mathematical point of view, PNs manage either state equations, algebraic equations, or some other kind of models that can govern the behavior of systems. An excellent reference of the theory that applies to Petri Nets can be found in [25].

A Petri Net (PN) has two types of nodes, namely, places and transitions. A place ($P$) is represented by a circle and a transition ($T$) by a bar. Places and transitions are connected by arcs. The number of places and transitions is finite and not zero. An arc is directed and connects either a place to a transition or a transition to a place. In other words, a PN is a bipartite graph; that is, places and transitions alternate on a path made up of consecutive arcs.

*Definition 1.* An unmarked ordinary PN is a quadruple $Q = \langle P, T, \mathrm{Pre}, \mathrm{Post}\rangle$ such that

$P = \{P_1, P_2, \ldots, P_n\}$ is a finite, nonempty set of places;

$T = \{T_1, T_2, \ldots, T_m\}$ is a finite, nonempty set of transitions;

$P \cap T = \emptyset$; that is, the sets $P$ and $T$ are disjointed;

$\mathrm{Pre} : P \times T \rightarrow \{0, 1\}$ denotes the input incidence application;

$\mathrm{Post} : T \times P \rightarrow \{0, 1\}$ denotes the output incidence application.

$\mathrm{Pre}(P_i, T_j)$ characterizes the weight of the arc $P_i \rightarrow T_j$. Thus, the weight is 1 if the arc exists and 0 if not. Conversely, $\mathrm{Post}(P_i, T_j)$ denotes the weight of the arc $T_j \rightarrow P_i$.

The following notations will be used:

${}^{\circ}T_j = \{P_i \in P \mid \mathrm{Pre}(P_i, T_j) > 0\}$ is set of input places of $T_j$;

$T_j^o = \{P_i \in P \mid \mathrm{Post}(P_i, T_j) > 0\}$ is set of output places of $T_j$;

${}^{\circ}P_i = \{T_j \in T \mid \mathrm{Pre}(P_i, T_j) > 0\}$ is set of input transitions of $P_i$;

$P_j^o = \{P_i \in P \mid \mathrm{Post}(P_i, T_j) > 0\}$ is set of output transitions of $P_i$.

*Definition 2.* A marked PN is a pair $R = \langle Q, m_0\rangle$ in which $Q$ is an unmarked PN and $m_0$ an initial marking. The validation conditions can be expressed in the following way: transition $T_j$ is enabled for a marking $m_k$ if $m_k(P_i) \geq \mathrm{Pre}(P_i, T_j)$ for every $P_i \in {}^{\circ}T_j$.

The reachability set of the PN is the set of all marked reachable from $m_0$, activating only enabled transitions; this set is denoted by $\mathbb{R}\langle Q, m_0\rangle$. A sequence of transitions firing of a PN $(Q, m_0)$ is a sequence of transition $S = T_i, T_j, \ldots, T_k \ldots$, such that $m_0 \xrightarrow{T_i} m_1 \xrightarrow{T_j} m_x \xrightarrow{T_k}$. The set of all firing sequences is called the language $\mathscr{L}(Q, m_0)$:

$$\mathscr{L}(Q, m_0) = \left\{ S = T_i, T_j, \ldots, T_k \ldots \wedge m_0 \xrightarrow{T_i} m_1 \xrightarrow{T_j} m_x \xrightarrow{T_k} \right\}. \quad (1)$$

*2.1. Interpreted Petri Nets.* Among the different types of PN (interpreted and colored), this research work deals with the interpreted Petri Nets (IPNs) mainly due to their features such as synchronization, timed places, and a part for processing data.

The inputs are associated with the transitions and outputs are associated with places. As Figure 1 illustrates, the event $E_j$ and condition $C_j$ are associated with transition $T_j$. Thus, condition $C_j$ is a Boolean function that depends on both the data processing part and the environment. Event $E_j$ is either an external event derived from the environment or the always occurring event $e$. Transition $T_j$ will be activated as follows.

If transition $T_j$ is enabled and if condition $C_j$ is true, when event $E_j$ occurs.

The product $R_j = E_j \cdot C_j$ is called the receptivity of transition $T_j$. Actions denoted in the figure $O_i^*$, $B_i^*$, and $A_i$ are associated with place $P_i$. When a token is deposited in place $P_i$, at instant $t$, the operation $O_i^*$ is carried out and the impulse action $B_i^*$ is sent to the environment. The Boolean output $A_i$ has the Boolean value 1 as long as there is a token in $P_i$.

A control interpreted Petri Net exhibits the following five characteristics.

Necessary characteristics are as follows:

(1) It is synchronized on external events and stable.

(2) It is safe ($m_0 = 1$).

(3) It is deterministic (no conflict).

Frequent characteristics are as follows:

(4) It relies on a data processing part, whose state is defined by a set of variables $V = \{V_1, V_2, \ldots\}$. This state is modified by operations $O_i^*$, which are associated with the places. It determines the value of the predicates $C_j^0$.

(5) It receives Boolean information $C_j^e$ from the ambient. It sends level actions $A_i$ (Booleans) and impulse actions $B_i^*$ (event type), associated with the places, to the environment.

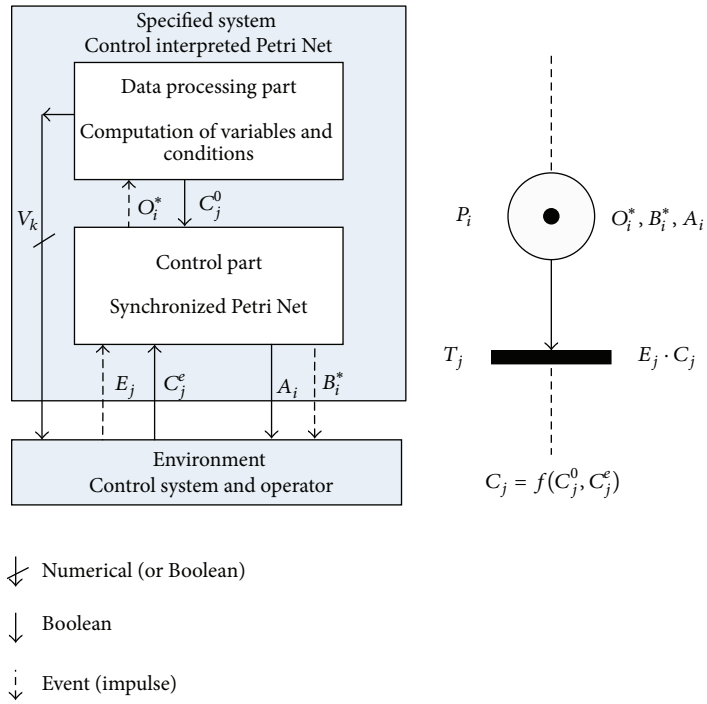


Figure 1: Control interpreted Petri Net.

## 3. Building Process of Model and FD Diagnoser

Let $R = (P, T, \text{Pre}, \text{Post}, m_0)$ be the PN that represents the model of the system to diagnose, where both normal and fault operation are considered. Moreover, transitions $T$ can be classified as unobservable $T_{uo}$ and observable $T_o$, as $T = T_o \cup T_{uo}$. Observable transitions are those transitions that are activated by control events $T_C$ (command supervisor) or the instrumentation deployed in the process $T_S$. Therefore, $T_o = T_C \cup T_S$. On the other hand, not observable transitions are those that are not detected by the system when they happen. Thus, fault transitions $T_f$ are a subset of the unobservable transitions $T_f \subseteq T_{uo}$. $T_f$ can be classified into disjointed sets, corresponding to different types of failure that may occur in the system, consequently $T_f = T_{f1} \cup \cdots \cup T_{fN}$. $N$ denotes the number of components of the system. According to this, when a $F_i$ type fault occurs, a transition of set $T_{fi}$ will be activated. Faults distribution $\Pi_f$ defines the faults set to be evaluated in the system.

This section summarizes the procedure for modeling and diagnoser designing. As previously mentioned, the system is made up of several subsystems working together and sharing different functionalities. Consequently, they are linked by several dependence relationships.

*Step 1* (to divide the system into subsystems). The system $H$ can be split into $M$ subsystems with close relationships among them, depending on their performance. Thus $H = H_1 \cup H_2 \cup \cdots \cup H_M$.

*Step 2* (to build up the PN model of the components of each subsystem). Let $R_j^i = (P_i, T_i, \text{Pre}, \text{Post}, m_0)$ be the marked PN of the $i$-component belonging to $j$ subsystem, $i = 1, \ldots, N_j$, with $N_j$ being the number of components in the subsystem $j$. $P_i$ is a set of places, $T_i$ is a set of transitions, Pre/Post are the input/output incidence applications, and $m_0$ is the initial marked.

*Step 3* (operation of integration). The following notation will be used to denote the integration operation for the subsystem $j$: $\widetilde{Q^j} = (\widetilde{P^j}, \widetilde{T^j}, \widetilde{\text{Pre}^j}, \widetilde{\text{Post}^j}, \widetilde{m_0^j})$. Therefore, $\widetilde{Q^j}$ refers to the representation of the subsystem behavior $H_j$ through of a stand-alone PN model, which includes $N_j$ different PN models corresponding to its components. Since this model integrates the normal and fault behavior of the system, transitions (observable $T_o$ and unobservable $T_{uo}$) can occur from any place of the model. Let $\widetilde{P^j}$ and $\widetilde{T^j}$ be the sets of places and transitions of $\widetilde{Q^j}$ of the $j$ subsystem, composed by the union of place and transitions of the components, respectively, as follows:

$$\widetilde{P^j} = \bigcup_{i=1}^{N} P_i,$$
$$\widetilde{T^j} = \bigcup_{i=1}^{N} T_i. \tag{2}$$

The following process has to be completed to build the integration model.

*Step 3.1.* Define the set of faults $F$ for each component of the subsystem:

$$F = \{F_1, F_2, \ldots, F_N\}. \tag{3}$$

*Step 3.2* (define the initial place $p_0^j$). It brings the initial places of each component of the subsystem together:

$$p_0^j = p_0^1, p_0^2, \ldots, p_0^N. \tag{4}$$

*Step 3.3* (assign a mark to initial place $p_0^j$). It defines the starting point for the evolution of the subsystem represented.

*Step 3.4* (build the branch corresponding to normal conditions $R_N^j$). It designates the interaction and evolution of the components. Transitions are merely composed by controller commands $T_C$. The set of places of the subsystem is made up of the places of each component.

*Step 3.5* (add the faults places $P_f^j$). A place for each fault from the set $F$ has to be generated.

*Step 3.6* (add the transitions faults $T_f^j$). From each place $p_i^j$ of general PN, a fault transition $t_f^j$ is therefore added:

$$\forall p_i^j : p_i \in P^j \longrightarrow \exists \left(p_i, t_j\right) \in F. \tag{5}$$

*Step 3.7*. Connect the normal places $p_i^j$ to the fault transitions $t_f^j$ and the faults transitions $t_f^j$ to the faults places $p_f^j$ by using arcs.

*Step 4* (refine the general model). It is necessary to consider only the observable part of general model $\widetilde{Q^j}$; therefore, $\widetilde{Q^j} = (\widetilde{P^j}, \widetilde{T^j}, \widetilde{\text{Pre}^j}, \widetilde{\text{Post}^j}, \widetilde{m_0^j})$ must be transformed to refined model $Q^j = (P^j, T^j, \text{Pre}^j, \text{Post}^j, m_0^j)$. It is only made up of observable transitions and places. According to this, fault transitions have to be replaced by readings of the sensor $T_S$. The following process must be followed to refine the model.

*Step 4.1* (identify the sensors of the system). Each subsystem has $Ns_j$ sensors.

*Step 4.2* (build a discrete set of the sensors outputs). $Y$ represents the number of combinations of the sensors readings. Hence, $Y$ are the inputs for the sensors integration table. Consider $|Y| = 2^{Ns}$. See first column (with "$*$") of Table 1.

*Step 4.3* (define the outputs of the integration table). An output (column) in the sensors integration table is added for each branch of normal conditions in PN for each subsystem. Each place represents the current state of sensors readings, as first row (†) in Table 1 depicts.

*Step 4.4* (build the sensors integration table). Consider $h_j = \widetilde{P} \rightarrow \widetilde{Y_j}, l = 1, \ldots, M$, where $\widetilde{Y_j}$ denotes the discrete set of sensor outputs possible of subsystem $j$; it defines

$$Y = \prod_{i=1}^{N} Y_j. \tag{6}$$

The integration sensor table $h_j$ is defined by $h(P) = (h_1(P_j), h_2(P_j), \ldots, h_N(P_j))$. The inputs of the integration table $Y$ and the sensors expected readings of each place of the normal branch must be compared; that is, if the sensor readings and the expected readings are the same, then $h_1(P_j)$ is classified as normal $N$. Moreover, the readings can lead to fault $F$ or to indeterminate $X$ when they have no information to use. Consider $Y_j = Y_{jN} \cup Y_{jF} \cup Y_{jX}$. The results of this classification are marked with "‡" in Table 1.

Table 1: Sensors integration table.

| $S_1{}^*$ | $S_2{}^*$ | $P_1^j(\overline{X}, Y)^\dagger$ | $P_2^j(X, Y)^\dagger$ | $P_3^j(X, \overline{Y})^\dagger$ | $\cdots$ | $P_i^j(X, Y)^\dagger$ |
|---|---|---|---|---|---|---|
| $\overline{L_1}^*$ | $\overline{L_2}^*$ | $F^\ddagger$ | $F^\ddagger$ | $X^\ddagger$ | $\cdots$ | $F^\ddagger$ |
| $\overline{L_1}^*$ | $L_2{}^*$ | $N^\ddagger$ | $X^\ddagger$ | $F^\ddagger$ | $\cdots$ | $X^\ddagger$ |
| $L_1{}^*$ | $\overline{L_2}^*$ | $X^\ddagger$ | $X^\ddagger$ | $N^\ddagger$ | $\cdots$ | $N^\ddagger$ |
| $L_1{}^*$ | $L_2{}^*$ | $F^\ddagger$ | $N^\ddagger$ | $F^\ddagger$ | $\cdots$ | $N^\ddagger$ |

*Step 4.5* (replace the fault transitions and remove the places which are not reachable). The fault transitions of the PN general model $\widetilde{Q^j}$ have to be replaced by the sensor readings. The fault places that cannot be identified because the information available does not allow enabling the market of that place must be removed of $\widetilde{Q^j}$. $\forall p_i : p_i \in P \rightarrow M(P_i) \notin R(Q, m_0)$.

Finally, the refined integration model of each subsystem is made up of normal places $p_i^j = p_i^1 \cup p_i^2 \cup \cdots \cup p_i^N$ and fault places $p_f$ given in $\Pi_f$, $p^j = p_N^j \cup p_F^j$. Transition $T^j$ includes control events $T_C$ and those considered in the integration table $T_S$; consider $T = T_C \cup T_S$. In conclusion, the refined model of each subsystem will be composed only by observable events.

*Step 5* (create the diagnoser). The diagnoser is a PN that is implemented considering the refined model of the system. Let $Q^j = (P^j, T^j, \text{Pre}^j, \text{Post}^j, m_0^j)$ be the PN that represents the refined model of each one of the subsystems $H_j$ of the whole system ($j = 1, \ldots, M$) and $t_f$ the final transition from a sequence $S$, defined as follows:

$$\mathscr{L}\left(T_{fi}\right) = \left\{St_f \in L : t_f \in T\right\}. \tag{7}$$

$\mathscr{L}(T_{fi})$ denotes a set of all sequences of $L$ (languages representing system behavior) that end in a fault transition, belonging to the class $t_{fi}$. Consider $t_f \in T$ and $S \in T^*$. The notation $t \in S$ is used to denote that $t$ is a transition of the sequence $S$, also writing $t_f \in T$ to any $t_f$.

In addition to this, a label has to be assigned to each system fault, producing a set of fault labels defined according to the expression $\Delta F = \{F_1, F_2, \ldots, F_m\}$.

In general, a set of labels $\Delta$ is made up of normal labels $N$ and fault labels $F$; thus $\Delta = \{N\} \cup \{F\}$.

In general, the diagnoser for system $H$ is a PN of the form $Q_d = (P_d, T_d, \text{Pre}, \text{Post}, m_0, p, t, t_{\text{end}})$, such that

(i) $P_d = \{P_1, P_2, \ldots, P_n\}$ is a finite, nonempty set of places;

(ii) $T_d = \{T_1, T_2, \ldots, T_m\}$ is a finite, nonempty set of transitions;

(iii) $P \cap T = \emptyset$; that is, the sets $P$ and $T$ are disjointed;

(iv) Pre: $P \times T \rightarrow \{1, 0\}$ is the input incidence application;

(v) Post: $T \times P \rightarrow \{1, 0\}$ is the output incidence application;

(vi) $m_0$ is the initial marked;

(vii) $p_0$ is the starting place,

(viii) $t_0$ is the starting transition;

(ix) $t_{\text{end}}$ is the ending transition.

The set of places $P_d$ of the diagnoser is an extension of the set of places of general model. Thus, a place $p$ of $Q_d$ is of the form $(p_i, l_i)$ where places belong to observable places, $p_i \in P_0$, and the label $l_i$ belongs to labels set $\Delta$. The labels for a specific place will be of the form $l_i = \{N\} \vee \{F\}$. Therefore, a place $P_d$ can take a label either of normal or fault operation.

The functions essential for building the diagnoser are as follows.

Label Assignation function LA: $P_o \times \Delta \times T^* \rightarrow \Delta$. Given $p \in P_o$, $l \in \Delta$, and $s \in L(Q, p)$, LA assigns the label $l$ over $s$ (transitions sequences) starting from $p$ and following the dynamics of $Q$, according to

$$\text{LA}(P, L, S) = \begin{cases} \{N\} \, si \quad \forall i \left[T_{fi} \notin S\right] \\ \{F\} \, si \quad \forall i \left[T_{fi} \in S\right]. \end{cases} \tag{8}$$

This mean that PN diagnoser $Q_d$ assigns to each place its respective label, depending on the transition fired, it will be $N$ (normal) if $T_0$ is fired in the sequence $S$ or it will be $F_i$ if it has been fired in a fault transition $T_f$.

It should be pointed out that sink places appear in the integration model $Q^j$. If marking fell into a sink place, the PN would be blocked. In order to avoid this problem, a fault expanding (FE) function has to be created, taking advantage of the concurrence capabilities of the PN.

Consequently, the fault expanding function is defined according to the following expression: EF: $R_N \times F_i \rightarrow R_F$, where $R_N$ is the normal operating branch and $R_F$ is the fault operating branch. For each set of failure $F_i$, a new branch of failures in the PN will be created so as to fulfill the role of overseeing the failures individually. The diagnoser $Q_d$ will have as many branches as the number of faults the system has. $R_G$ specifies the total number of branches of the diagnoser:

$$R_{Qd} = \sum_{i=1}^{M} R_{fi}. \tag{9}$$

The procedure for creation of the diagnoser is as follows.

*Step 5.1.* Define the initial place $p_0$.

*Step 5.2* (add a starting transition $t_0$). Connect $p_0$ to $t_0$ by using an arc.

*Step 5.3* (add an ending transition $t_{\text{end}}$). Connect $t_{\text{end}}$ to $p$ by using an arc.

*Step 5.4* (add the fault branch). For each fault of $\Pi_{fi}$, a fault branch is built according to the fault expanding function EF. Each branch includes a normal branch parallel to a fault branch since each place can fire a fault transition $t_f$ or normal transition $t_N$. When a $t_f$ is fired, the PND goes to a fault place $p_f$ and consequently LA function assigns a label. In order to continue the evolution of the PN, the sensors reading plus control transition must be fired $(T_S + T_C)$.

*Step 5.5* (add recovery transitions $t_R$). For each one of the fault places, its evolution can go in a way of normal conditions (step before) or sensor readings can change from fault state to normal state. Therefore, a recovery transition $t_R$ must be added to the branch and the mark of place must return to normal place priorly:

$$\begin{aligned} &\left(\forall p_{fi} \in P_F\right)\left(\exists t_{Ri} \in T_R : t = t_S\right) \\ &\cdot \left(p_{fi} \times t_{Ri} \longrightarrow p_{N(i-1)}\right). \end{aligned} \tag{10}$$

*Step 5.6* (connect $t_0$ and $t_{\text{end}}$ to the branches). An arc must be connected from $t_0$ to each initial place of the branch and, in the same way, from each initial place of the branch to $t_{\text{end}}$. In this step, weighting arcs are used for avoiding conflicts in the PN; therefore, the weight of the arc must be equal to the number of faults plus one (corresponding to the normal branch of its respective subsystem).

*Step 5.7* (temporal dynamics for diagnosing intermittent faults). For each fault place, a time variable $t$ is added. It is started when the mark arrives to the fault place and ends when the mark goes out because the recovery transitions are fired. This time stored by $t$ must be added to another when the branch falls in fault. $\forall p_{fi} \in P_F$, $\exists t \rightarrow p_{fi} = (M(p_i), t_i)$, where $i = 1, \ldots, Z$, with $Z$ being the numbers of faults of the system. For each fault branch of the diagnoser, a temporal account place $p_C$ is added. When a fault transition is fired, a mark must be added to $p_C$. This means that the numbers of faults are stored into this place $p_C$. And the sum of the times registered by $t$ variable is stored in $p_C$. Then, from each fault transition of the fault branch, an arc must be connected to temporal account place $p_C$. The numbers of marks in the temporal account place is the sum of the marks registered in each fault place:

$$\begin{aligned} &\forall R_F \in R, \\ &\exists p_C : t_{fi} \longrightarrow p_C, \wedge, \\ &t = \sum_{i=1}^{M} t_i, \\ &m\left(p_C\right) = \sum_{i=1}^{Z} m\left(p_{fi}\right), \\ &i = 1, \ldots, Z. \end{aligned} \tag{11}$$

*Step 5.8* (reset of temporal dynamic). When the operator (i.e., the pilot) stops the diagnoser to replace or fix a component, the variables $t$ and temporal account place $p_C$ must be resetting. An arc must be added from $p_C$ to $t_{\text{end}}$ for this purpose. Another arc starting from $t_{\text{end}}$ but without place for arrival should be also added. The weight of this arc is equal to numbers of marks stores in $p_C$. In this manner, $p_C$ is empty (no marks) and $t$ variable is set to zero.

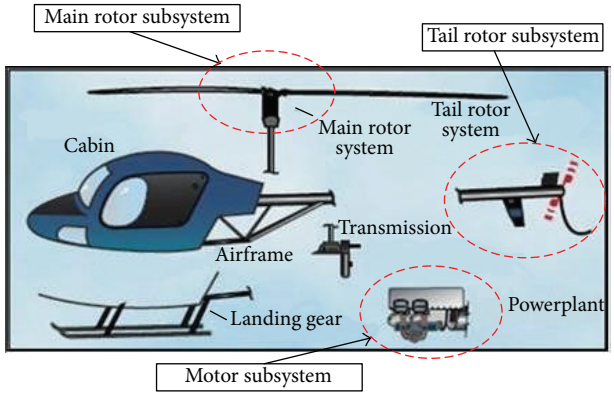


Figure 2: Subsystems of the helicopter.

PN diagnoser evaluates possible changes in each branch. The diagnoser emerges in normal or failure operation according to LA function. The diagnoser evaluates each fault separately and takes into account the failures that are caused by other failures in their transitions; therefore it is able to detect failures simultaneously, regardless of the order in which they occur.

## 4. Application to Helicopter

A radio controlled helicopter model Vario Benzin Trainer was chosen for this study. It relies on six servo-controllers for maneuvering. It has a length of 1.46 meters, a wingspan of 2 meters, and a weight of 7.3 Kg. The system (helicopter) is made up of three main subsystems: engine, main rotor (known as plate), and tail rotor. If any of them fails, the mission unavoidably has to be aborted, because the helicopter could crash with serious consequences rapidly.

The motor system is made up of a small gasoline engine and a servo that controls the throttle. It is responsible for generating the rotation of blades, both main rotor and tail rotor, in a mechanically settled proportional relationship. An electronic controller is typically used to maintain the speed of the rotor constant, independently of the attack angle of the blades. This angle is directly related to the drag that engine has to overcome.

The main rotor system is controlled by four servos. They allow modifying the lift and helicopter attitude (i.e., the roll and pitch angles) by varying the collective and cyclic pitch of the blades along their rotation. Thus, the collective pitch varies the lift and the cyclic one modifies the attitude.

The main component of the tail rotor system (antitorque) is a servo-controller, which receives commands from the pilot in order to modify the pitch angle of the tail rotor blades. The variation of this angle allows modifying the yaw angle of the helicopter (heading), mainly because it varies the antitorque that is required to compensate the one generated by the main rotor.

The full system relies on additional devices, such as power supply, navigation sensors (i.e., inertial systems, tail gyro, GPS, and electronic compass), controllers, communications (antennas and radios), and the ground control station.

The following lines are dedicated to describe the diagnoser implementation for the helicopter, according to the process described in previous sections. It is worth stressing that the no faults have been considered in the controller.

*4.1. Classification in Subsystems.* The helicopter is classified into three fundamental subsystems, as Figure 2 highlights, namely, the motor subsystem $H_1$, the main rotor subsystem $H_2$, and tail rotor subsystem $H_3$.

*4.2. Building the PN Model of the Components of Each Subsystem*

*Subsystem Motor $H_1$.* This subsystem has two components to model: controller $H_1^1$ and throttle servo $H_1^2$. The monitored variables are fuel flow ($F$), engine temperature ($T$), engine speed (RPM), and current required by the throttle servo ($I1$). The faults to diagnose are high temperature of the motor: Fault Warming Motor (FWM), lack of gasoline in the fuel tank: Fault Level Flow (FLF), and failure of servo stuck (FSS1). Therefore, the set of faults for motor subsystem is $F1 = \{\text{FWM, FLF, FSS1}\}$.

Figure 3 shows the PN model of the controller and servo. The PN model of the controller represents its cyclic work that is represented by the place $C1$. If the transition AS1 (activation servo 1) is activated, it will move to place $C2$ (new location). When the transition AS2 is activated, the model will return to $C1$, representing a new location. In the same way, in the motor servo PN model, SNA1 represents servo normal operation. When transition AS1 is activated in synchrony with the controller, the model moves to SR1. Finally, when the transition AS2 is activated, the PN returns to SNA1. The rest of the branches represent faults of the subsystem, which are depicted by using dark places and transition, due to being unobservable.

*Main Rotor Subsystem $H_2$.* As previously mentioned, there are two components to model in the plate subsystem, namely, plate servos and controller. The monitored variables are currents required by plate servos ($I2, I3, I4, I5$), roll and pith angles ($Rl, Pt$), and the vibrations in tail ($G$). The faults considered in the diagnoser are failure of servos stuck (FSS2, FSS3, FSS4, and FSS5) and mechanical imbalance of the chassis (FM). Therefore, the set of faults for main plate subsystem is as follows: $F2 = \{\text{FSS2, FSS3, FSS4, FSS5, FM}\}$.

*Tail Rotor Subsystem $H_3$.* Two components have been considered in the model: tail servo and controller. The monitored variables are current required by tail servo ($I6$) and yaw angle ($Yw$). The fault to diagnose is a failure of servo stuck in the tail FSS6. Subsequently, the set of faults is defined as follows: $F3 = \{\text{FSS6}\}$.

*4.3. Operation of Integration*

*Motor Subsystem.* It consists in joining in a single model the two models of the previous step. The initial place $P0 = \{C1; \text{SNA1}\}$ is the union of the initial places of controller ($C1$)

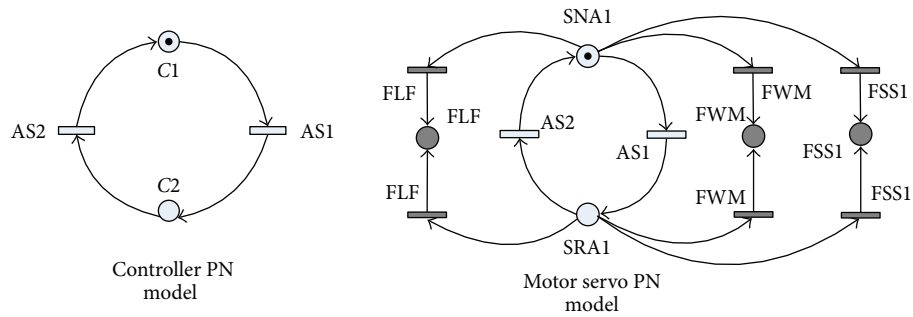


Figure 3: PN model of the components, motor subsystem.

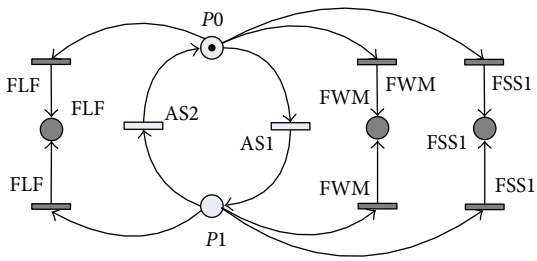


Figure 4: General PN model of motor subsystem.

and servo (SNA1). $P1$ integrates the places of the controller ($C2$) and servo (SRA1). The rest of the components of the PN models are kept. General model can be seen in Figure 4.

*Main Rotor Subsystem.* An initial place $P2 = \{C3; \text{SNA2}\}$ has been defined as the union of the initial places of controller $C3$ and current action of the servos SNA2; SNA3; SNA4; SNA5. Normal branch has been built. Fault places and transitions have been added for each fault of the set $F2$. $P3$ integrates the places of the controller $C4$ and required action of servo SRA2; SRA3; SRA4; SRA5.

*Tail Rotor Subsystem.* As in previous cases, an initial place $P4 = \{C5; \text{SNA6}\}$ has been defined as the union of the initial places of controller $C5$ and current action of the tail servo SNA6. After defining the normal branch, fault places and transitions have been added for each fault of the set $F3$. $P5$ integrates the places of the controller $C6$ and required action of servo SRA6.

### 4.4. Refining the General Model

*4.4.1. Identify the Sensor for Each Subsystem Ns according to the Sensors Required for Monitoring the Desired Variables.* As previously mentioned, motor sensors' set is made up of a fuel flow sensor $F$, a temperature sensor $T$, a speed sensor (encoder) RPM, and current servo of fuel $I1$. The set $Ns$ for the engine subsystem is therefore defined as follows: $Ns1 = \{F, T, \text{RPM}, I1\}$.

Following the same procedure, the main rotor sensor set is $Ns2 = \{I2, I3, I4, I5, Rl, Pt, G\}$, which is made up of four current sensors for the servos of the plate ($I2, I3, I4, I5$), an inertial sensor that estimates the roll and pitch angles ($Rl$, $Pt$), and an accelerometer for measuring vibrations in the frame $G$. Finally, tail rotor sensor comprises two sensors, a current sensor of the tail servo $I6$ and magnetic compass for estimating the heading or yaw angle $Yw$. Subsequently, set is $Ns3 = \{I6, Yw\}$.

*4.4.2. Build a Discrete Set of the Sensors Outputs.* The combinations of sensor readings that have to be used as inputs of the sensors integration table are shown in Table 2.

According to previous definitions, motor subsystem has $|Y_1| = 2^{Ns} = 2^4 = 16$ combinations. According to this formulation, main rotor subsystem has $|Y_2| = 16$ combinations (notice that the four sensors' current of the plate is represented just by a single value: $Is$). Tail rotor subsystem has therefore $|Y_3| = 4$. Each reading is measurement in a discrete way and a threshold has been independent.

*4.4.3. Define the Outputs of the Integration Table.* Two outputs have been added for the motor subsystem, namely, $P0$ and $P1$. Under normal operating conditions, the reading of fuel must be $F$ (i.e., the fuel in the tank is above the minimum threshold of 100 mL), the reading of the temperature must be $\overline{T}$ (i.e., the engine temperature is below the threshold, 90°C), the value for the speed of the rotor must be RPM (i.e., the engine revolutions are over the threshold of 1200 rpm), and the readings from the servo current should be $\overline{I}$ (i.e., the servos demand a value of current from 100 to 600 mA, which denotes normal functioning instead of blockage problems).

*Main Rotor Subsystem.* Two outputs have been included in the table for $P2$ and $P3$. The readings of current sensors $\overline{Is}$ must be in the same range of that of the servos of the plate (different values happened if a more powerful servo would be used to control the tail). The motor subsystems and the readings of the position angles (roll and pitch) must be $\overline{Rl}$, $\overline{Pt}$ (i.e., angle below 45°) and vibrations must be $\overline{G}$ (i.e., vibrations below 2.5 g).

*Tail Rotor Subsystem.* Two outputs have been incorporated to the table for $P4$ and $P5$. The readings of the current servo must be $\overline{I6}$ and the readings of the yaw angle must be $\overline{Yw}$. The outputs of the integration table are represented in Table 3.

Table 2: Outputs' sensor set.

| Combinations | Motor readings | | | | | Plate readings | | | Tail rotor readings | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $F$ | $T$ | RPM | $I1$ | $Is$ | $Rl$ | $Pt$ | $G$ | $I6$ | $Yw$ |
| 1 | $\overline{F}$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $\overline{Is}$ | $\overline{Rl}$ | $\overline{Pt}$ | $\overline{G}$ | $\overline{I6}$ | $\overline{Yw}$ |
| 2 | $\overline{F}$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $I1$ | $\overline{Is}$ | $\overline{Rl}$ | $\overline{Pt}$ | $G$ | $\overline{I6}$ | $Yw$ |
| 3 | $\overline{F}$ | $\overline{T}$ | RPM | $\overline{I1}$ | $\overline{Is}$ | $\overline{Rl}$ | $Pt$ | $\overline{G}$ | $I6$ | $\overline{Yw}$ |
| 4 | $\overline{F}$ | $\overline{T}$ | RPM | $I1$ | $\overline{Is}$ | $\overline{Rl}$ | $Pt$ | $G$ | $I6$ | $Yw$ |
| 5 | $\overline{F}$ | $T$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $\overline{Is}$ | $Rl$ | $\overline{Pt}$ | $\overline{G}$ | | |
| 6 | $\overline{F}$ | $T$ | $\overline{\text{RPM}}$ | $I1$ | $\overline{Is}$ | $Rl$ | $\overline{Pt}$ | $G$ | | |
| 7 | $\overline{F}$ | $T$ | RPM | $\overline{I1}$ | $\overline{Is}$ | $Rl$ | $Pt$ | $\overline{G}$ | | |
| 8 | $\overline{F}$ | $T$ | RPM | $I1$ | $\overline{Is}$ | $Rl$ | $Pt$ | $G$ | | |
| 9 | $F$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $Is$ | $\overline{Rl}$ | $\overline{Pt}$ | $\overline{G}$ | | |
| 10 | $F$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $I1$ | $Is$ | $\overline{Rl}$ | $\overline{Pt}$ | $G$ | | |
| 11 | $F$ | $\overline{T}$ | RPM | $\overline{I1}$ | $Is$ | $\overline{Rl}$ | $Pt$ | $\overline{G}$ | | |
| 12 | $F$ | $\overline{T}$ | RPM | $I1$ | $Is$ | $\overline{Rl}$ | $Pt$ | $G$ | | |
| 13 | $F$ | $T$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $Is$ | $Rl$ | $\overline{Pt}$ | $\overline{G}$ | | |
| 14 | $F$ | $T$ | $\overline{\text{RPM}}$ | $I1$ | $Is$ | $Rl$ | $\overline{Pt}$ | $G$ | | |
| 15 | $F$ | $T$ | RPM | $\overline{I1}$ | $Is$ | $Rl$ | $Pt$ | $\overline{G}$ | | |
| 16 | $F$ | $T$ | RPM | $I1$ | $Is$ | $Rl$ | $Pt$ | $G$ | | |

Table 3: Outputs' set of the integration table.

| Readings motor | | | | Motor places | |
|---|---|---|---|---|---|
| $F$ | $T$ | RPM | $I1$ | $P0(F, \overline{T}, \text{RPM}, \overline{I})$ | $P1(F, \overline{T}, \text{RPM}, \overline{I})$ |
| Plate readings | | | | Plate places | |
| $Is$ | $Rl$ | $Pt$ | $G$ | $P2(\overline{Is}, \overline{Rl}, \overline{Pt}, \overline{G})$ | $P3(\overline{Is}, \overline{Rl}, \overline{Pt}, \overline{G})$ |
| Tail rotor readings | | | | Tail rotor places | |
| | | $I6$ | $Yw$ | $P4(\overline{I6}, \overline{Yw})$ | $P5(\overline{I6}, \overline{Yw})$ |

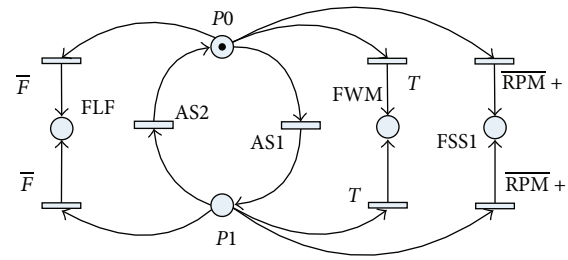


Figure 5: Refined PN model of the motor.

*4.4.4. Build the Sensors Integration Table.* The sensor readings $Y$ have to be cross-checked with their corresponding normal operations of the place for each subsystem. This allows determining whether sensors readings indicate a fault $F$, a normal behavior $N$ or which combinations of sensors will provide with undetermined information to the diagnosis system $X$.

The integration table of motor subsystem is showed in Table 4. This step requires a deep knowledge of the system.

*4.4.5. Replace the Faults Transitions and Remove the Places which are Not Reachable.* This step consists in replacing transitions unobservable (fault transitions) by sensors readings identified in the integration table. Motor subsystems: transition FLF is replaced by $\overline{F}$, the transition FWM is replaced by $T$, and the transition FSS1 is replaced by $(\overline{\text{RPM}} + I1)$. Main rotor subsystem: any fault of the stuck servo FSS is replaced by the readings of the $Is + Rl + Pt$ and the mechanic fault FM is replaced by $G$. The refined PN of the motor is shown in Figure 5. Tail rotor subsystem: fault of the servo stuck FSS6 is replaced by the readings of $I6 + Yw$.

*4.5. Building of the Diagnoser.* Finally, the whole algorithm is integrated in one single PN, the FD helicopter. The PN that represents the diagnoser is mainly composed of three branches, corresponding to each subsystem: motor, main rotor, and tail rotor branches, as Figure 6 depicts. An initial place $p_0$ for starting and starting transition $t_0$ have been defined. The starting transition $t_0$ has to be activated manually by the operator in the ground control station so as to start the diagnoser. This moves a token for each of the branches of the helicopter subsystems. Likewise, the PN diagnoser has an ending transition $t_{\text{end}}$, which allows the pilot to finish the diagnoser. The refined PN models of each subsystem are joined to the diagnoser with their places and transitions, respectively; then, based on the functions of fault expansion EF, the fault braches are added for each subsystem. Recovery transitions $t_R$ for each one of the fault places of each subsystem are added, connecting with the previous normal place. Consequently, each one of the transitions and places has to be connected by arcs, considering that their weights are equal to the numbers of faults plus one (normal branch). Finally, the temporal dynamic must be implemented adding the temporal account places $p_C$, which is used for storing the time and number of fault for each variable of the system.

The diagnoser makes an online assessment of whole system and serves as the supervisor, indicating whether a branch falls into failure by means of the label assigned

Table 4: Integration table of the motor subsystem.

| Sensors readings | | | | Motor places | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $F$ | $T$ | RPM | $I1$ | $P0(F, \overline{T}, \text{RPM}, \overline{I})$ | | | $P1(F, \overline{T}, \text{RPM}, \overline{I})$ | | | | |
| $\overline{F}$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | FLF | $N$ | FSS1 | $N$ | FLF | $N$ | FSS1 | $N$ |
| $\overline{F}$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $I1$ | FLF | $N$ | FSS1 | FSS1 | FLF | $N$ | FSS1 | FSS1 |
| $\overline{F}$ | $\overline{T}$ | RPM | $\overline{I1}$ | FLF | $N$ | $N$ | $N$ | FLF | $N$ | $N$ | $N$ |
| $\overline{F}$ | $\overline{T}$ | RPM | $I1$ | FLF | $N$ | $N$ | FSS1 | FLF | $N$ | $N$ | FSS1 |
| $\overline{F}$ | $T$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | FLF | FWM | FSS1 | $N$ | FLF | FWM | FSS1 | $N$ |
| $\overline{F}$ | $T$ | $\overline{\text{RPM}}$ | $I1$ | FLF | FWM | FSS1 | FSS1 | FLF | FWM | FSS1 | FSS1 |
| $\overline{F}$ | $T$ | RPM | $\overline{I1}$ | FLF | FWM | $N$ | $N$ | FLF | FWM | $N$ | $N$ |
| $\overline{F}$ | $T$ | RPM | $I1$ | FLF | FWM | $N$ | FSS1 | FLF | FWM | $N$ | FSS1 |
| $F$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $N$ | $N$ | FSS1 | $N$ | $N$ | $N$ | FSS1 | $N$ |
| $F$ | $\overline{T}$ | $\overline{\text{RPM}}$ | $I1$ | $N$ | $N$ | FSS1 | FSS1 | $N$ | $N$ | FSS1 | FSS1 |
| $F$ | $\overline{T}$ | RPM | $\overline{I1}$ | $N$ | $N$ | $N$ | $N$ | $N$ | $N$ | $N$ | $N$ |
| $F$ | $\overline{T}$ | RPM | $I1$ | $N$ | $N$ | $N$ | FSS1 | $N$ | $N$ | $N$ | FSS1 |
| $F$ | $T$ | $\overline{\text{RPM}}$ | $\overline{I1}$ | $N$ | FWM | FSS1 | $N$ | $N$ | FWM | FSS1 | $N$ |
| $F$ | $T$ | $\overline{\text{RPM}}$ | $I1$ | $N$ | FWM | FSS1 | FSS1 | $N$ | FWM | FSS1 | FSS1 |
| $F$ | $T$ | RPM | $\overline{I1}$ | $N$ | FWM | $N$ | $N$ | $N$ | FWM | $N$ | $N$ |
| $F$ | $T$ | RPM | $I1$ | $N$ | FWM | $N$ | FSS1 | $N$ | FWM | $N$ | FSS1 |

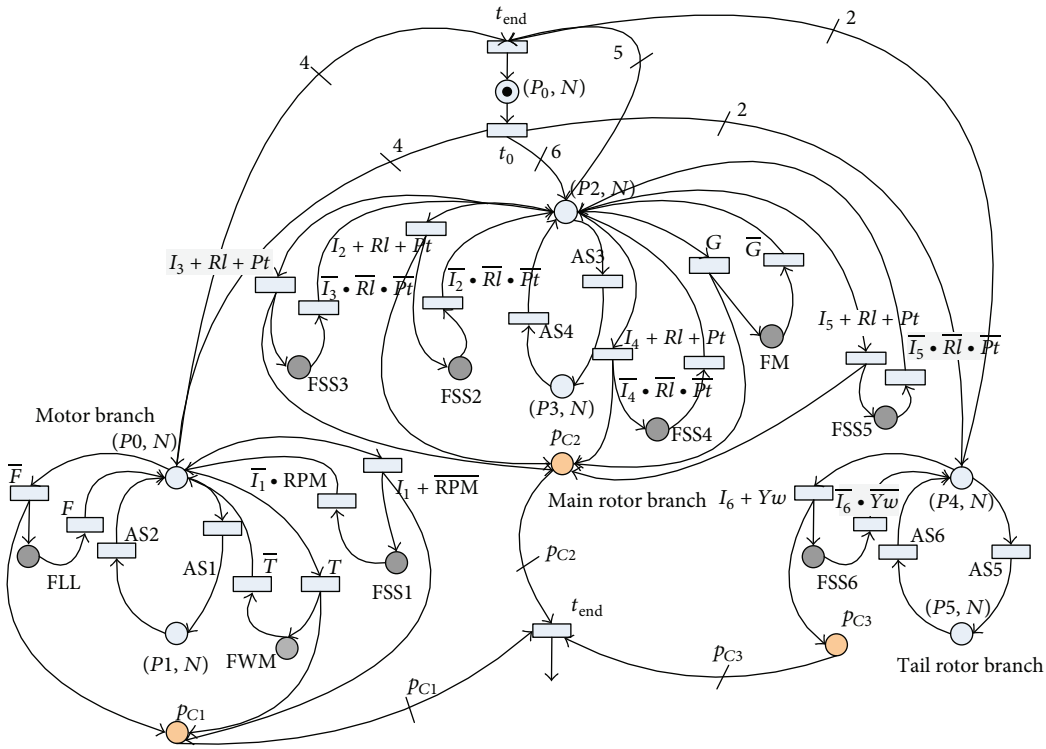


Figure 6: Petri Net diagnoser.

by function LA. If this happens, the other branches continue assessing the system, although, due to vulnerability of the helicopter, it should be landed for repairing immediately.

Sometimes, small faults happen during a flight for short periods of time. If no additional action is provided, the recovery transitions will hide these faults to the operator. The diagnoser relies on a display to show the operator this kind of information.

## 5. Analysis of Results

A series of field missions has been carried out in order to evaluate the PN diagnoser performance. Several lab tests were

Table 5: Thresholds for monitored signals of the UAV.

| Variable | Symbol | V. Normal (1) | V. Fault (0) |
|---|---|---|---|
| Temperature | $T$ | $T$ (ambient) to 90°C | >90°C |
| Voltage | $V$ | 6.5–7 V | <6 V |
| Fuel flow | $F$ | 100–500 mL | <50 mL |
| RPMs | RPM | 1200–1500 rpm | <1200 rpm |
| Vibrations | $G$ | 0.5–2.5 g | >2.5 g |
| Servos current | $Is$ | 100–600 mA | >600 mA |
| Roll/pitch angles | ∅ | $Rl/Pt < 45°$ | >45 degrees |

previously carried out in order to set up the thresholds for the variables that determine the normal operation of the helicopter. A summary of the variables threshold in each of the variables is shown in Table 5.

Diverse types of failures were intentionally introduced in the system during the field tests. Namely, different unbalances were introduced both in main and in tail rotor, aiming at increasing the vibrations of the helicopter frame. Moreover, several problems in the engine carburetor were also induced (e.g., the fuel flow was restricted to the engine).

A complete Data Acquisition System (DAQ) has been designed and built so as to implement the diagnoser. It is made up of two main components: the instrumentation onboard the helicopter and the components running into the ground control station (GCS). Since DAQ has been built for providing data and housing to the PND algorithm, it is responsible for sensing the field variables of the helicopter and sending the information to the GCS. This information is used to feed the fault diagnosis algorithm; which monitors the normal or fault operation of the aircraft by using the diagnoser Petri Net. The PND reports the frequency with which any device on the aircraft falls into alarm or fault.

A LabVIEW module that contains a graphical interface is responsible for acquiring, storing, and displaying all data from the UAV. The DAQ system works in parallel and independently of the control system; therefore it is fully autonomous and does not share any resources with the aircraft. This allows using the system with any other UAV. Moreover, since there is a redundancy in the information (i.e., position and attitude), a simple comparison method could be useful to detect malfunctioning of the navigation systems.

It is worth noting that the tests required to evaluate the performance of the diagnoser are extremely risky since they require forcing real failures during the flights. Furthermore, some difficulties are commonly found when working with real-time systems that have to be shipped on a vehicle with an extremely limited payload (e.g., data processing and storing limitations and consumption of computing elements). Additional problems arise when the information should be transmitted to ground station (noise, range of communications, and limited bandwidth). Therefore, it is essential to be as pragmatic as possible; that is, the maximum sampling time should be used and only the essential discrete variables monitored.

Another important limitation is the reaction time that the pilot requires to decide if a mission has to be aborted when a warning/fault is detected. The UAS operator could observe during this period of time how a monitored variable falls into alarm (warning) and, in some cases, into fault for short periods of time, and recover the normal condition.

A summary report corresponding to the results obtained during real flights (normal, vibrations in main and tail rotor, and wrong mix in fuel) is shown in Table 6. It reveals the number of times that each variable exceeded the threshold during the mission. For instance, in the normal conditions flight, the vibrations exceeded 93 times the value of the 2.5 g.

The behavior of the flight in normal conditions is showed for temperature, vibrations, and RPM variables in Figure 7. It can be concluded that those variables deserve special mention over the others. Each flight stage is referred to by numbers; for example, take off is denoted by stage 2. Although RPM and vibration thresholds exceeded intermittently their failure thresholds during the flight, no warning was activated. The RPM value highlights changes during takeoff and landing stages. Vibrations during idle stage (motor started but UAV on the ground) exceeded the thresholds of fault; nevertheless they recover their normal range during the flight.

The helicopter safety strongly depends on its suitable mechanical performance, and the expertise of the pilot supported by this diagnoser is the best indicator to decide whether a mission should be aborted or not.

## 6. Conclusions

This research addresses the fault diagnosis applied to an UAS by using Petri Nets. A methodological process for creating the model and the diagnoser has been presented. The process has been split into subsystems. This allows assessing the failures in an independent way.

As result highlights, the created Petri Net diagnoser is able to detect not only permanent but also intermittent failures, providing the pilot with a set of health monitoring alarms.

The main advantages of the proposed method are the ability to be implemented in complex processes, the flexibility to make changes (i.e., including additional devices to the diagnoser), the reduction of the combinational explosion, and a systematic construction.

An application on a real system (a radio controlled helicopter) has been presented as demonstrator. Since the helicopter is a quite vulnerable platform, it requires a robust fault diagnosis method. Although real experiments pose a high risk, a valuable amount of flights have been carried out in order to assess the performance of the algorithm and methodology. Moreover, the use of a commonly used tool such as LabVIEW allows providing the diagnoser with additional features that turn out to be of great importance for preventive maintenance.

This work opens the door to some fields of development in the area of the FD to be considered in further work, such as to perform spectral analysis of vibration aiming at detecting more types of mechanical damages. Moreover it could be applied to generate mathematical models of normal and fault operation of the aircraft.

Table 6: Summary of the helicopter behavior in missions.

| Variable | | #Normal F | #F Vib Plato | #F Vib Cola | #F Mix |
|---|---|---|---|---|---|
| Temperature | $T$ | 0 | 0 | 0 | 0 |
| Voltage | $V$ | 0 | 0 | 0 | 0 |
| Fuel flow | $F$ | 0 | 0 | 0 | 0 |
| RPMs | RPM | 6 | 10 | 3 | 25 |
| Vibrations | $G$ | 93 | 28 | 26 | 53 |
| Servos current | $Is$ | 0 | 1 | 0 | 0 |
| Roll/pitch/yaw angles | $\emptyset$ | 0 | 0 | 0 | 0 |

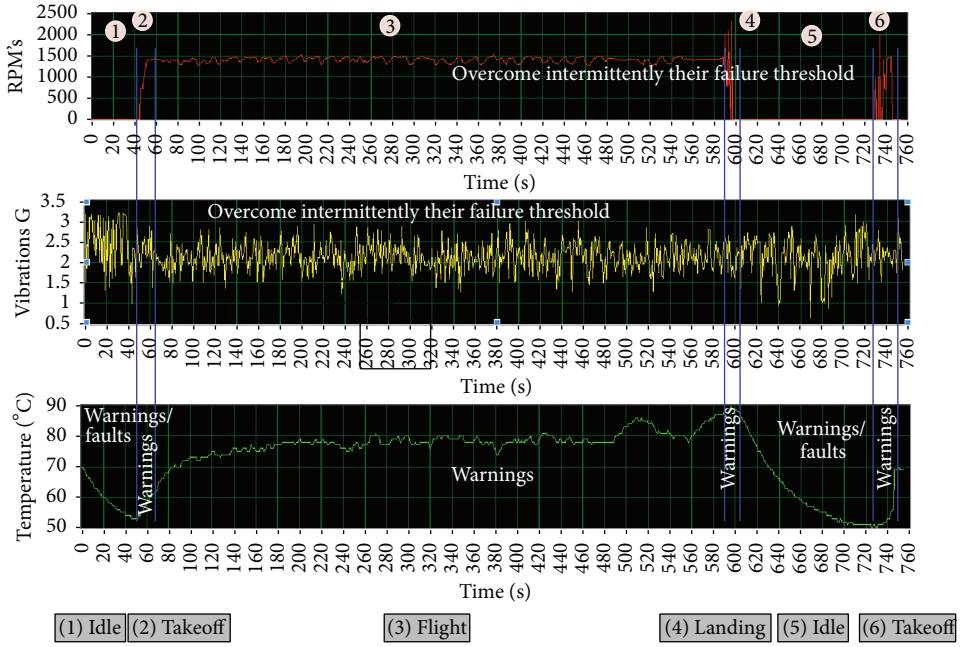


Figure 7: UAV variables flying in normal conditions.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work has been supported by the project RoboCity2030-III-CM (Robótica Aplicada a la Mejora de la Calidad de Vida de los Ciudadanos; Fase III; S2013/MIT-2748), funded by the I+D program at Comunidad de Madrid and cofunded by Fondos Estructurales of European Union and by the project Protección Robotizada de Infraestructuras Críticas, DPI2014-56985-R, by Ministerio de Economía y Competitividad of Spain.

## References

[1] National RPAS Regulations, January 2015, https://www.eurocontrol.int/articles/national-rpas-regulations.

[2] M. A. Garzon Oviedo, A. Barrientos, J. Del Cerro et al., "Tracking and following pedestrian trajectories, an approach for autonomous surveillance of critical infrastructures," *Industrial Robot*, vol. 42, no. 5, pp. 429–440, 2015.

[3] K. Hayhurst, J. Maddalon, and P. Miner, "Unmanned aircraft hazards and their implications for regulation," in *Proceedings of the 25th Digital Avionics Systems Conference*, NASA Langley Research Center, Hampton, Eastsound, Wash, USA, October 2006.

[4] R.-E. Precup, P. Angelov, B. S. J. Costa, and M. Sayed-Mouchaweh, "An overview on fault diagnosis and nature-inspired optimal control of industrial process applications," *Computers in Industry*, 2015.

[5] Z. Gao, S. X. Ding, and C. Cecati, "Real-time fault diagnosis and fault-tolerant control," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3752–3756, 2015.

[6] X. Qi, D. Theilliol, J. Qi et al., "Fault diagnosis and fault tolerant control methods for manned and unmanned helicopters: a literature review," in *Proceedings of the 2nd International Conference on Control and Fault-Tolerant Systems (SysTol '13)*, pp. 132–139, Nice, France, October 2013.

[7] A. Wahrburg and J. Adamy, "Robust fault isolation observers for non-square systems—a parametric approach," in *Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pp. 1275–1280, National

Autonomous University of Mexico, Mexico City, Mexico, August 2012.

[8] L. Liu, Y. Shen, and E. H. Dowell, "Integrated adaptive fault-tolerant h infinity output feedback control with adaptive fault identificatio," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 3, pp. 881–889, 2012.

[9] J. T. Qi, D. L. Song, C. Wu, J. Han, and T. Wang, "KF-based adaptive UKF algorithm and its application for rotorcraft UAV actuator failure estimation," *International Journal of Advanced Robotic Systems*, 2012.

[10] C. Wu, J. Qi, and J. Han, "AESMF based sensor fault diagnosis for RUAVs," in *Proceedings of the 24th Chinese Control and Decision Conference (CCDC '12)*, pp. 3384–3389, Taiyuan, China, May 2012.

[11] J.-T. Qi and J.-D. Han, "Application of wavelets transform to fault detection in rotorcraft UAV sensor failure," *Journal of Bionic Engineering*, vol. 4, no. 4, pp. 265–270, 2007.

[12] R. Waschburger, H. M. Paiva, J. J. Ribeiro E Silva, and R. K. H. Galvão, "Fault detection in a laboratory helicopter employing a wavelet-based analytical redundancy approach," in *Proceedings of the Conference on Control and Fault-Tolerant Systems (SysTol '10)*, pp. 70–75, IEEE, Nice, France, October 2010.

[13] V. K. Kaliappan, H. Young, A. Budiyono, and D. Min, "Fault tolerant controller design for component faults of a small scale unmanned aerial vehicle," in *Proceedings of the 8th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2011*, pp. 79–84, Incheon, Republic of Korea, November 2011.

[14] J. Qi, X. Zhao, and Z. Jiang, "An adaptive threshold neural-network scheme for rotorcraft UAV sensor failure diagnosis," in *Advances in Neural Networks*, vol. 4493 of *Lecture Notes in Computer Science*, pp. 589–596, Springer, Berlin, Germany, 2007.

[15] O. González-Miranda and M. Cerrada-Lozada, "Diagnosis of controlled discrete-event systems: an approach based on chronicles and modular analysis by using automata models," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 11, no. 2, pp. 191–201, 2014.

[16] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[17] A. Ramírez, E. Ruiz, I. Rivera, and E. López, "Online fault diagnosis of discrete event systems. A Petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 31–39, 2007.

[18] S. Gene and S. Lafortune, "Distributed diagnosis of place-bordered petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 2, pp. 206–219, 2007.

[19] F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich, and P. Cheung, "Monitoring and fault diagnosis of hybrid systems," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 35, no. 6, pp. 1225–1240, 2005.

[20] P. M. Nasr and A. Y. Varjani, "Petri net model of insider attacks in SCADA system," in *Proceedings of the 11th International ISC Conference on Information Security and Cryptology (ISCISC '14)*, pp. 55–60, IEEE, Tehran, Iran, September 2014.

[21] O. Grunt and R. Briš, "SPN as a tool for risk modeling of fires in process industries," *Journal of Loss Prevention in the Process Industries*, vol. 34, pp. 72–81, 2015.

[22] L. Pettigrew, V. Blomenhofer, S. Hubert, F. Groß, and A. Delgado, "Optimisation of water usage in a brewery clean-in-place system using reference nets," *Journal of Cleaner Production*, vol. 87, pp. 583–593, 2015.

[23] I. Koch, "Petri nets in systems biology," *Software & Systems Modeling*, vol. 14, no. 2, pp. 703–710, 2015.

[24] D. M. Muñoz, A. Correcher, E. García, and F. Morant, "Stochastic DES fault diagnosis with coloured interpreted petri nets," *Mathematical Problems in Engineering*, vol. 2015, Article ID 303107, 13 pages, 2015.

[25] R. David and H. Alla, *Discrete, Continuous and Hybrid Petri Nets*, Springer, Berlin, Germany, 2005.