Méthodologie et application du méta-diagnostic de systèmes complexes sur un banc d'intégration avionique

R. Cossé[1,2], D. Berdjag[2], S. Piechowiak[2], D. Duvivier[2] and C. Gaurel[1]

[1] AIRBUS HELICOPTERS, Aéroport Marseille Provence, 13725 Marignane France
{ronan.cosse, christian.gaurel}@airbus.com
[2] LAMIH UMR CNRS 8201, Université de Valenciennes, 59313 Valenciennes France
{denis.berdjag, sylvain.piechowiak, david.duvivier}@univ-valenciennes.fr

Résumé

Le nombre de fonctionnalités des systèmes embarqués aéronautiques ne cesse d'augmenter et leur tests deviennent très difficiles à mettre en place. Des erreurs peuvent apparaître lors des phases de validation des logiciels embarqués. Ces tests sont réalisés sur des sous-systèmes appelés bancs d'intégration qui recréent l'environnement réel d'un aéronef. Cependant ces sous-systèmes peuvent également subir des pannes. Cet article propose une méthode pour réaliser leur diagnostic. Cette méthode est mise en place sur le banc d'intégration d'un hélicoptère via le développement d'un logiciel et d'une méthodologie de diagnostic adaptés. Ceci est illustré avec la description d'un problème réel apparu pendant une phase de test.

1 Introduction

La complexité des systèmes embarqués est importante, surtout dans le monde aéronautique et en particulier celui des hélicoptères. Précédemment, la réalisation des fonctions avioniques des hélicoptères se faisait via l'échange de messages et de données issus de capteurs et d'unités de calculs (architecture fédérée). Aujourd'hui les calculateurs embarquent plusieurs fonctions qui communiquent au travers d'un réseau standardisé (architecture intégrée) [1]. Entre ces calculateurs, le nombre d'événements, de messages et de paramètres à traiter ne cessent de s'accroître de manière importante. De plus, la puissance des calculateurs actuels et la réduction de la taille des composants conduisent à une concentration des fonctions avioniques au sein des systèmes embarqués.

Dans ce contexte, le banc d'intégration avionique est un outil essentiel de vérification des fonctions embarquées de l'hélicoptère. Il est constitué d'équipements avioniques réels et d'outils de simulation et d'espionnage, voir Figure 1. Son objectif est de valider par des tests les spécifications fonctionnelles des logiciels embarqués. Ces tests sont réalisés dans les configurations définies par ces spécifications. La NASA utilise par exemple des bancs d'intégration pour tester de nouvelles technologies de maintenance, de diagnostic et de pronostic des pannes grâce à son système Advanced Diagnostics and Prognostics Testbed (ADAPT) [2].

La complexité des systèmes de tests est supérieure à celle du système embarqué car plusieurs équipements supplémentaires sont nécessaires pour sa validation. Il s'agit d'outils de simulation en temps réel, de reconfiguration automatique et de séquencement des tests. De plus, la reconfiguration du câblage du banc d'intégration est indispensable pour chaque version spécifique d'hélicoptère, en fonction des options et adaptations demandées par les clients. Ceci implique l'ajout de commutateurs permettant une reconfiguration de la connectique correspondant au passage d'une version d'hélicoptère à une autre. Comme il y a une augmentation du nombre d'équipements avioniques sur les hélicoptères, et une diversité importante des demandes de clients d'hélicoptères dans le monde, il y a de plus en plus de reconfigurations de tests et d'essais à réaliser par modèle d'hélicoptère.

Des erreurs de configuration peuvent entraîner l'indisponibilité du banc d'intégration ou même la non-certification d'un hélicoptère. Les erreurs à diagnostiquer sont les erreurs de câblage et de configuration du banc, les erreurs de configuration des logiciels de simulation et d'espionnage, les erreurs du système embarqué. De telles erreurs sont extrêmement difficiles à localiser et à différencier. De plus, l'évolution permanente des hélicoptères imposent une évolution technique et rapide des bancs d'intégration. Il s'agit donc de proposer une méthode qui s'adapte aux bancs existants en constante évolution mais aussi à tous les nouveaux bancs développés, avec un minimum de modifications. La méthode proposée dans cet article consiste à introduire une approche algébrique adaptable et généralisable aux différentes technologies utilisées sur les différents bancs. Dans le cadre du diagnostic du banc d'intégration, il s'agit de simuler une panne dans l'attente d'un comportement anormal cohérent. S'il est incohérent, il s'agit dès lors de diagnostiquer un résultat dont le symptôme peut être une fonctionnalité au comportement normal alors qu'elle devrait être en panne. C'est le méta-diagnostic du banc d'intégration. Nous proposons un algorithme de méta-diagnostic qui sera implémenté sur un banc d'intégration chez Airbus Helicopers.

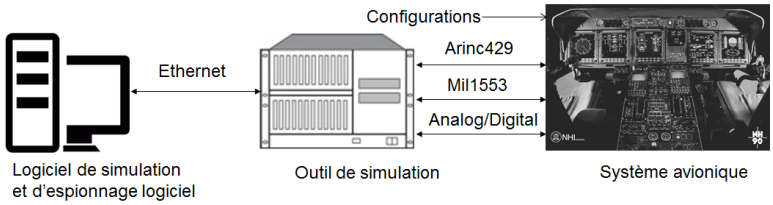

FIGURE 1 – Schéma du banc d'intégration

2 Formulation du problème

2.1 État de l'art

Afin de diagnostiquer les systèmes complexes, l'utilisation des modèles structurels a montré son efficacité. Les raisonnements élaborés jusqu'à aujourd'hui et leur applicabilité sur les systèmes de test sont présentés. Les méthodes structurelles à base de modèles reposent sur la comparaison entre les observations du système et celle prédites par une simulation. Chittaro [3] a défini les modèles comportementaux et structurels qui caractérisent, selon lui, les connaissances fondamentales d'un système. Ressencourt [4] a mis en application ce principe pour le diagnostic de systèmes embarqués dans l'automobile. L'architecture des systèmes embarqués automobiles est similaire à celle de l'aéronautique. Ses caractéristiques reposent sur des contraintes de développement liées à la certification du logiciel et sur l'augmentation du nombre de fonctions.

Cette complexité croissante incite à abstraire le comportement en cas de pannes afin de réduire la phase de raisonnement. Berdjag [5] propose une approche algébrique de réduction de modèles. Le principe de base d'une décomposition en sous-systèmes repose sur l'architecture des communications entre les composants. Deux types de topologies sont utilisés [3] :

- la topologie structurelle basée sur le type de communications entre composants, et l'emplacement des composants au sein du système ;
- la topologie fonctionnelle basée sur le rôle de chaque composant et les flux de données pour la réalisation de chaque fonction embarquée.

Dans le cadre de systèmes industriels, les techniques à base d'arbre de défaillances ont été les premières techniques pour le diagnostic avionique [6] : un arbre de défaillance permet de représenter les

conséquences d'apparition d'un défaut. D'autres ont aussi permis d'intégrer l'incertitude avec des raisonnements qualitatifs en intelligence artificielle pour des systèmes difficiles à diagnostiquer comme ceux étudiés par Mocko [7]. Le développement d'une architecture et de méthodes de diagnostic génériques pour des systèmes industriels a été étudié par Ly [8]. Kuntz [9] a ajouté des relations causales appelées "minimal cuts" au diagnostic de logiciels embarqués avioniques. Des définitions et des méthodes pour la modélisation de systèmes aéronautiques ont été données par Jauberthie [10] pour estimer les paramètres de vol (roulis, tangage, lacet, vitesse) d'un aéronef.

D'un côté, ces représentations sont adaptées aux systèmes aéronautiques, mais ne prennent pas en compte les spécificités des systèmes de test. De l'autre, l'approche à base de modèle du point de vue intelligence artificielle parait être une approche efficace et robuste aux reconfigurations du système si le modèle en tient compte [11].

Malheureusement, ces solutions ne sont pas adaptées pour le méta-diagnostic d'un banc d'intégration. Cette considération est la différence fondamentale entre le diagnostic d'un système avionique sur un aéronef, et celui du système embarqué en phase de développement logiciel. Dans le premier cas, il repose sur une architecture connue et sûre. Dans le deuxième, le banc d'intégration est en constante évolution logicielle et matérielle. La complexité du banc d'intégration comprend celle des logiciels embarqués, des logiciels de simulation et la reconfiguration du cablage de l'hélicoptère. Ces points ne sont à ce jour pas considérés dans les modélisations. Notre approche est basée sur la description du diagnostic proposée par De Kleer [12]. Elle se place dans la continuité du diagnostic pour des systèmes avioniques embarqués proposé par Ly [8], Lefebvre [6], et Kuntz [9]. Le problème est formalisé sous la forme du Meta-diagnostic d'un système complexe [13]. La base du raisonnement consiste à abstraire l'environnement et le banc d'intégration grâce à une organisation en partition et à implémenter un algorithme qui utilise les opérations algébriques afin de converger vers la faute [14].

Notre contribution est soumise aux contraintes industrielles (norme de développement DO-178B, faible disponibilité du banc d'intégration) qui limiteront le choix des modèles. Considérant une topologie structurelle et fonctionnelle, nous décrivons la décomposition du banc sous la forme d'un partitionnement de l'ensemble des composants. Nous présentons ici l'industrialisation de la méthode pour un banc d'intégration et des résultats grâce au développement d'un logiciel de méta-diagnostic complet, doté d'une interface ergonomique pour le recueil des symptômes, d'un moteur décisionnel et connecté à une base de données reprenant l'historique des pannes et des solutions. Nous proposons ici d'introduire de nouvelles méthodes algébriques pour réaliser l'étude de tels systèmes.

2.2 Principe de diagnostic à base de modèle

La localisation des défauts d'un système composé de plusieurs composants dédiés à la réalisation de plusieurs fonctions embarquées repose sur la comparaison entre des observations (mesures) et un ou plusieurs modèles visant à représenter le fonctionnement et l'architecture du banc d'intégration, voir Figure 2. Il est indispensable, dans le cas d'un diagnostic à base de modèle, de disposer d'une représentation. Les méthodes existantes reposent sur un modèle événementiel ou logique plus ou moins facile à élaborer, voir [5] et [9]. Il faut donc trouver le bon compromis entre rapidité de construction du modèle et du raisonnement d'un côté, et précision des résultats pour le problème posé de l'autre. Dans ce contexte, l'algèbre des ensembles et des partitions nous paraît le meilleur choix pour résoudre le méta-diagnostic du banc d'intégration :

- le comportement interne des composants n'est pas nécessairement connu ;
- le modèle peut être construit rapidement grâce aux documents de spécifications et grâce à la connaissance des experts ;
- le résultat du méta-diagnostic pourra être vite calculé.

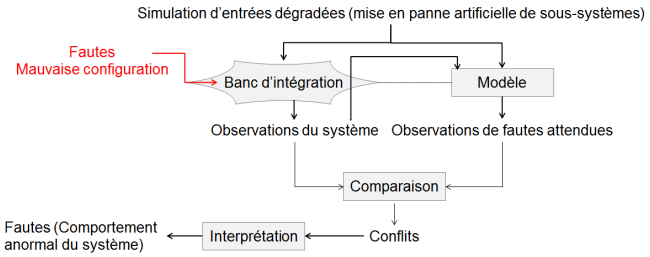

FIGURE 2 – Diagnostic à base de modèle du banc d'intégration

L'utilisation du banc d'intégration tel qu'il est décrit par la Figure 2 repose sur la simulation d'entrées dégradées, i.e mise en panne artificielle de sous-systèmes, et sur l'observation des fautes attendues en sortie du système. Dans le cas d'un défaut, ou d'une mauvaise configuration du banc d'intégration, de nouveaux défauts, non prédits par le modèle, apparaissent et génèrent des conflits. Ce sont ces défauts que nous proposons de diagnostiquer à l'aide de notre représentation.

3 Modélisation

Pour modéliser les systèmes complexes avec les approches structurelles et fonctionnelles décrites précédemment, nous utilisons une représentation en treillis. La clef de notre modélisation est l'utilisation des partitions selon différentes catégories et granularités. Le composant est l'élément de base de notre méta-diagnostic qui est regroupé en sous-ensembles, qui eux-mêmes sont organisés sous forme de partitions, voir Figure 3.

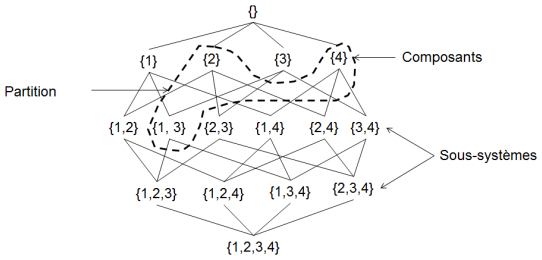

FIGURE 3 – Treillis des sous-systèmes

Pour chacun de ces niveaux, une fonction de diagnostic est introduite pour effectuer le diagnostic de l'élément, du sous-ensemble ou de la partition. Le passage d'un niveau à un autre est possible grâce à la structure imbriquée du modèle. La faute sera visible différemment en fonction de la granularité, et le coût de l'observation sera également différent. Par exemple, une faute sur une partition est observable sous la forme de dysfonctionnement général. Elle se propage à un sous-système puis à un composant. Une discussion sur la méthode à mettre en place pour diagnostiquer les bancs d'intégration est nécessaire non seulement pour démontrer la validité de nos travaux, mais aussi pour décrire les possibilités et les avantages à utiliser ce formalisme. Ces points sont abordés dans cette section.

3.1 Utilisation du treillis des partitions

Dans les paragraphes suivants, nous utilisons les notations suivantes : $S = \{c_i, i \in [1,n]\}$ est l'ensemble des n composants du système ; P_k désigne une partition, σ_j désigne un sous-système, c_i désigne un composant de S. Σ désigne l'ensemble de tous les sous-systèmes. Une partition P_k est un ensemble de n_P sous-systèmes $\sigma_i \in \Sigma$:

$$P_k = \{\sigma_j, j \in [1,n_p] | \forall x \neq y; \sigma_x \cap \sigma_y = \emptyset, \text{ et } \bigcup_{j=1}^{n_p} \sigma_j = S\}.$$

Nous notons $\mathscr{P}$ l'ensemble de toutes les partitions, $\mathscr{P}_s$ l'ensemble de partitions structurelles qui représentent le type de communications, l'emplacement des composants, et $\mathscr{P}_f$ l'ensemble des partitions fonctionnelles qui représentent le rôle de chaque composant par rapport aux fonctions avionique. Ainsi, nous considérons :

- un système de n composant $S = \{c_i | i \in [1,n]\}$,
- une topologie du système qui le décompose en différents sous-systèmes σ_j,
- une fonction de vérification au niveau des composants $checkC$ (voir Définition 1).
- une fonction de vérification au niveau des sous-systèmes $checkS$ (voir Définition 2),
- une fonction de vérification au niveau du système : $checkP$ (voir Définition 3),

Définition 1. *Une fonction de vérification checkC d'un composant c est définie par :* $checkC : COMPS \rightarrow \{0,1,-1\}$ *tel que* $checkC(c) = 1 \Leftrightarrow$ *le composant est sain ;* $checkC(c) = 0 \Leftrightarrow$ *le composant est fautif ;* $checkC(c) = -1 \Leftrightarrow$ *l'état du composant est inconnu.*

Définition 2. *Une fonction de vérification checkS d'un sous-système σ est définie par :* $checkS : \Sigma \rightarrow \{0,1,-1\}$ *tel que* $checkS(\sigma) = 1 \Leftrightarrow \forall c_i \in \sigma, checkC(c_i) = 1$ *;* $checkS(\sigma) = 0 \Leftrightarrow \exists c_i \in \sigma, checkC(c_i) = 0$ *;* $checkS(\sigma) = -1 \Leftrightarrow$ *la vérification de σ est inconnue.*

Définition 3. *Une fonction de vérification checkP d'une partition P est définie par :* $checkP : \mathscr{P}(S) \rightarrow \{0,1,-1\}$ *tel que* $check(P) = 1 \Leftrightarrow \forall \sigma_i \in P, checkS(\sigma_i) = 1$ *;* $check(P) = 0 \Leftrightarrow \exists \sigma_i \in P, checkS(\{\sigma_i\}) = 0$ *;* $check(P) = -1 \Leftrightarrow$ *la vérification de P est inconnue.*

Grâce à ces fonctions, il est alors possible de définir les contraintes du système à satisfaire pour résoudre le problème de méta-diagnostic :

Définition 4. *Les contraintes du système sont définies comme l'ensemble* $CONS = \{P_k \in \mathscr{P}(S)$ *tel que* $checkP(P_k) \neq -1\}$.

L'ensemble $CONS$ permet de connaître les contraintes applicables au banc d'intégration.

3.2 Partitions testables

La connaissance représentée dans le modèle est basée sur la structure du système, c'est-à-dire la connexion et les communications entre les composants. Cette connaissance structurelle va de pair avec la connaissance fonctionnelle, c'est-à-dire le rôle de chaque composant dans la réalisation d'une fonction. Nous utilisons les opérations de combinaison de partitions afin de calculer les partitions testables, voir définition 5. Nous prenons l'exemple simple d'un système composé de quatre éléments $S = \{C_i,\ i \in [1,4]\}$, de trois partitions P_1 = { $\{C_1$,C_2, C_3 } , $\{C_4\}$ }, P_2 = { $\{C_1$,$C_2\}$, $\{C_3$,C_4,$\}\}$, et P_3 ={ { $C_1\}$, $\{C_2$,C_3 }, { C_4 } }.

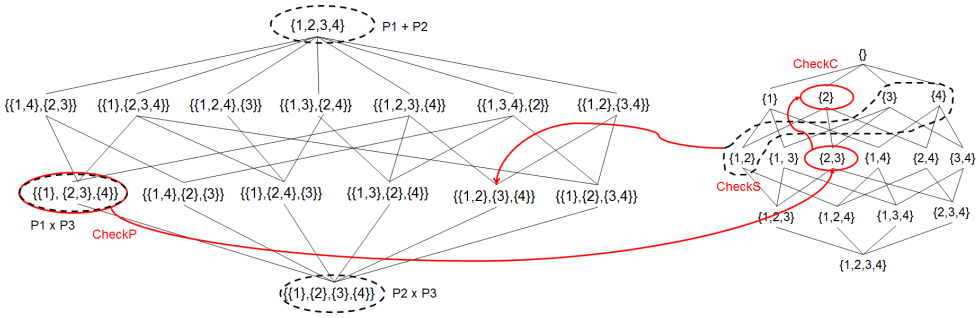

FIGURE 4 – Interconnexions entre la topologie fonctionnelle et structurelle

Définition 5. *Les sous-systèmes σ_j résultants du produit de deux partitions $P = \{\sigma_{j_P}, j_P \in [1, n_p]\}$ et $Q = \{\sigma_{j_Q}, j_Q \in [1, n_q]\}$ sont définis par :*

$$\forall \sigma_j \in P \times Q, \exists \sigma_{j_P} \in P, \exists \sigma_{j_Q} \in Q, \sigma_j = \sigma_{j_P} \cap \sigma_{j_Q}.$$

Le produit de partitions est l'élément qu'il est possible de tester sur un système réel. La vérification des aspects fonctionnels pour une topologie structurelle donnée permet d'abord de lier un défaut à un comportement, et ensuite de réduire les fautes possibles à une fonction et une topologie, puis à un sous-système et un composant, voir Figure 4. La combinaison des partitions permet d'obtenir une partition testable, c'est-à-dire une partition répondant aux contraintes définies initialement. L'élément testable sur le système réel est le produit d'une partition fonctionnel avec une partition structurelle. Si on suppose que P_1 et P_2 représentent deux fonctionnalités et P_3 représente la topologie du système, alors les partitions testables sont $P_1 \times P_3 = \{ \{C_1\}, \{C_2, C_3\}, \{C_4\} \}$ et $P_2 \times P_3 = \{ \{C_1\}, \{C_2\}, \{C_3\}, \{C_4\} \}$.

3.3 Méthodologie proposée

La méthodologie employée est constituée de cinq étapes :

1. Identifier les fonctionnalités défaillantes, i.e. les observations de comportements anormaux résultant d'un test d'une fonction avionique ;
2. Identifier les topologies responsables de la perte des fonctionnalités identifiées à l'étape 1 ;
3. Déterminer les partitions vérifiables, celles liées à un défaut, celles dont le comportement est correct ;
4. Identifier les composants ou les sous-systèmes fautifs grâce à l'algorithme de méta-diagnostic ;
5. Proposer les résultats de manière itérative et des observations "optimales" à réaliser pour la nouvelle itération de l'algorithme (étape 1).

Ces étapes sont réalisées grâce à un algorithme itératif qui fournit un méta-diagnostic sous la forme d'une liste de composants "suspects" et d'une proposition d'observation optimale dont la mise-à-jour permettrait d'affiner le méta-diagnostic après une nouvelle itération de l'algorithme.

4 Meta-Diagnostic du banc d'intégration

4.1 Description du Meta-Diagnostic

Il est temps de discuter de la manière dont le treillis et l'opération de multiplication aident à la résolution d'un problème de Meta-Diagnostic du banc d'intégration. Pour nous, le treillis n'est qu'une représentation abstraite du système réel. Il permet de donner un sens à l'élément SD qui caractérise la connaissance du système pour un problème de méta-diagnostic $(SD, COMPS, OBS)$. Une fois la représentation obtenue, les observations liées au problème de méta-diagnostic sont relevées en vue de leur traitement. Une méthode est mise en place pour donner les causes possibles de ces symptômes dans le cadre spécifique des bancs d'intégration. Le résultat est soit un composant C_i, soit un sous-système Σ_i déclaré comme fautif par rapport à une partition fonctionnelle P_f et une partition structurelle P_s du treillis des partitions.
Voici les étapes qui aboutissent à un méta-diagnostic basé sur cette méthode, voir Figure 5 :

1. Observation d'une défaillance et de comportements anormaux par rapport à une procédure de test, une configuration du système et un résultat attendu ("Démarrage du méta-diagnostic").
2. Définition des partitions qui caractérisent le système et la défaillance par rapport au formalisme en treillis $(SD, COMPS)$ ("Initialisation des partitions").
3. Définition des observations associées à la défaillance, c'est-à-dire des couples de partitions $(P_i, P_j), P_i \in \mathscr{P}_f, P_j \in \mathscr{P}_s$
4. Relevé des observations OBS. ("Relevé des observations sur le système").
5. Déduction des fautes possibles ("Calcul des diagnostics : *CheckMultiplicationPartition(d)*").
6. Vérification des méta-diagnostics sur le système réel (composants C_i ou sous-systèmes Σ_i) ("Vérification : *CheckComponents(Fc)* et *CheckSubsystems(Σ_c)*").
7. Si les opérations n'ont pas permis de cibler le composant et la fonctionnalité défaillants, Alors (une itération supplémentaire est nécessaire) :
   (a) Ajout de nouvelles observations ("Modélisation des nouvelles observations")
   (b) Itération supplémentaire de l'algorithme à partir du point 4.
   (c) Sinon affichage des méta-diagnostics.

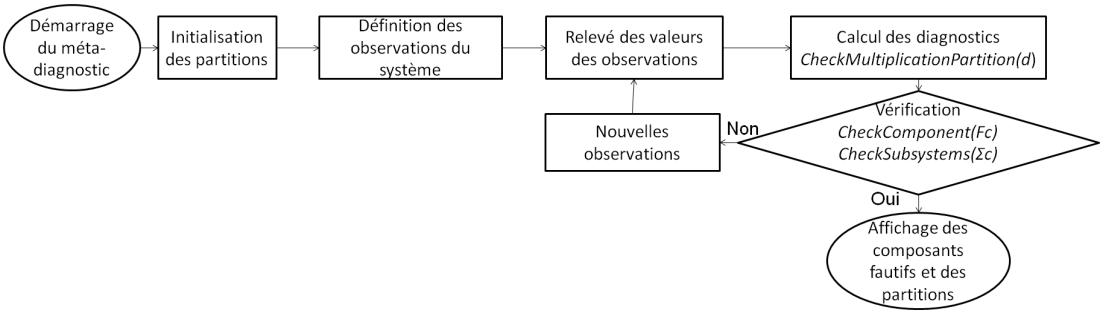

FIGURE 5 – Organigramme de l'algorithme de méta-diagnostic

4.2 Algorithme de méta-diagnostic

L'algorithme propage les informations de défauts relevés par les observations sur deux topologies en treillis et identifie les composants/sous-systèmes responsables. Ensuite, une vérification est requise, et son résultat, s'il est non-concluant, sert de mise à jour, en propageant l'information pour une nouvelle itération, voir Proposition 1. Les données sont composées de composants du système C_i, organisés en sous-systèmes Σ_j, eux-mêmes organisés en partitions P_i.

Proposition 1. $\forall P, Q \in \mathscr{P}^2, checkP(P \times Q) = 0 \Rightarrow checkP(P) = 0 \wedge checkP(Q) = 0.$

L'utilisation du langage objet JAVA permet de représenter simplement la structure de donnée, voir le schéma UML Figure 6.

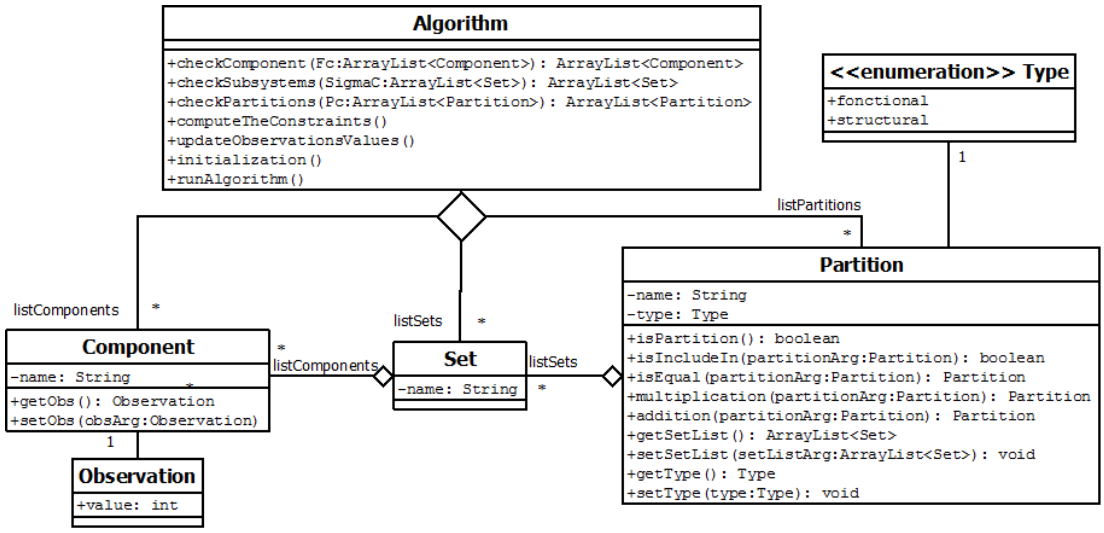

FIGURE 6 – Schéma UML

De plus, il est possible d'ajouter simplement de nouvelles opérations sur les partitions, comme celles présentées dans la section 3.2. Pour ajouter un algorithme de méta-diagnostic, il suffit d'implémenter la classe *Algorithm*. Afin de faciliter la lisibilité de l'algorithme, il a été divisé en trois. Le premier $DIAG(\mathscr{L}(\Sigma, \subseteq))$ est l'algorithme principal qui initialise les variables avec les partitions du système, et les contraintes $Cons = \{P \in \mathscr{P}, checkP(P) \neq -1\}$. Les différentes variables sont Δ pour le méta-diagnostic, F_c/U_c pour les composants fautifs/non-fautifs, P^-/P^+ pour les partitions fautives/non-fautives, Σ^-/Σ^+ pour les sous-systèmes fautifs/non-fautifs, End pour calculer la condition d'arrêt de l'algorithme, et $NCons$ pour calculer le nouvel ensemble des contraintes.

L'algorithme $CheckMultiplicationPartition(d, Cons)$; permet de vérifier les partitions définies comme des contraintes pour le méta-diagnostic. Si une partition $p_{mult} = p_j \times p_k$ est en défaut, i.e. la fonctionnalité du système n'est pas réalisée, alors tous les sous-systèmes issus de p_{mult} peuvent être fautifs. Ils sont vérifiés avec l'algorithme $CheckSubsystems(\Sigma^-)$ qui distingue notamment les sous-systèmes en fonction du protocole de communication utilisé. Les composants supposés fautifs sont vérifiés avec l'algorithme $CheckComponents(F_c)$.

Input : $d = \{p_i, i \in [1, n]\}, Cons = \{cons_i\}$
Output : $\Delta(Diagnosis)$
Global variables : F_c, U_c, P^-, P^+, End
$\Delta, F_c, U_c, P^+, P^-, \Sigma^-, \Sigma^+ \leftarrow \{\}$; $End_C; End_\Sigma \leftarrow false$; $NCons \leftarrow \{\}$;
while $\neg End_\Sigma \wedge \neg End_C$ **do**
  $CheckMultiplicationPartition(d, Cons)$;
  if $\neg End_C$ **then**
    $CheckComponents(F_c)$;
  end
  if $\neg End_\Sigma$ **then**
    $CheckSubsystems(\Sigma^-)$;
  end
  if $\neg End_\Sigma \wedge \neg End_C$ **then**
    foreach $p_i \in NCons$ **do**
      if $check(p_i) \neq -1$ **then**
        $Cons \leftarrow Cons \cup \{p_i\}$
      end
    end
  end
end

Algorithme 1: $DIAG(\mathscr{L}(\Sigma, \subseteq))$

4.3 Complexité de l'algorithme

Nous allons calculer la complexité temporelle de l'algorithme. Elle dépend de plusieurs variables : $nbComp$: le nombre de composant du système, $nbPart$: le nombre de partition, $nbCons$: le nombre de contrainte.
La première étape consiste à charger les éléments du système, la complexité est $O(nbComp+nbPart+nbCons)$. La seconde étape consiste à calculer le produit de partition pour toutes les contraintes.
La multiplication de deux partitions P_1 et P_2 consiste à faire l'intersection tous les ensembles de P_1 avec tous les ensembles de P_2. La complexité de l'intersection de deux ensembles est $O(nbComp^2)$. Cette opération est faite pour tous les ensembles de P_1 en O(nbComp) et tous les ensembles de P_2 en $O(nbComp)$. La complexité du produit de partition est donc $O(nbComp^4)$. La complexité de la seconde étape est $O(nbCons \times nbComp^4)$.
La troisième étape consiste à calculer de nouvelles observations pour résoudre le problème de diagnostic si les contraintes existantes ne suffisent pas. Cette opération est réalisée pour toutes les partitions. La complexité est $O(nbPart^2 \times nbComp^4)$.
Ensuite, les opérations de vérifications sont réalisées par l'ingénieur d'essai. La complexité totale de l'algorithme est $O((nbCons + nbPart^2) \times nbComp^4)$. Elle est donc très importante, d'où l'importance de diminuer le nombre d'itérations en proposant un diagnostic le plus exhaustif possible.
Nous allons aussi évaluer la complexité spatiale de l'algorithme. La taille mémoire utilisé par les variables U_c, F_c est $O(nbComp)$. Celle des variables P^+ et P^- est $O(nbPart^2) = O(2^{2\times nbComp})$. Pour Σ^-, Σ^+, l'espace utilisé est $O(2^{nbComp})$. La complexité spatiale de l'algorithme est $O(2^{2\times nbComp})$.

Input : $d = \{p_i\}, Cons = \{cons_i\}$
Outputs : $F_c, U_c, P^-, \Sigma^-, \Sigma^+$
foreach $(p_j, p_k) \in P^2 : p_i \neq p_j$ **do**
  $p_{mult} \leftarrow p_j \times p_k$
  if $p_{mult} \in Cons$ **then**
    if $checkP(p_{mult}) = 0$ **then**
      $P^- \leftarrow P^- \cup \{p_i\}$
      foreach $\sigma_i \in p_i$ **do**
        $\sigma_i \leftarrow \sigma_i - \{c_k / c_k \in U_c\}$
        if $\sigma_i = \{c_i\}$ **then**
          $F_c \leftarrow F_c \cup \sigma_i$
        end
        else if $\sigma_i \notin \Sigma^+$ **then**
          $\Sigma^- \leftarrow \Sigma^- \cup \{\sigma_i\}$
        end
      end
    end
    if $checkP(p_{mult}) = 1$ **then**
      $P^+ \leftarrow P^+ \cup \{p_i\}$
      foreach $\sigma_i \in p_i$ **do**
        if $\sigma_i = \{c_i\}$ **then**
          $U_c \leftarrow U_c \cup \sigma_i$
        end
        else
          $\Sigma^- \leftarrow \Sigma^- - \{\sigma_i\}$
          $\Sigma^+ \leftarrow \Sigma^+ \cup \{\sigma_i\}$
        end
      end
    end
  end
  else
    $NCons \leftarrow NCons \cup \{p_{mult}\}$
  end
end

Algorithme 2: $CheckMultiplicationPartition(d, Cons)$

Inputs : $\Sigma^-(faulty\ subsystems)$
Outputs : $\Delta\ F_c, U_c, End_\Sigma$
Initialization : $\sigma_+, \sigma_- \leftarrow I$;
foreach $\Sigma_i \in \Sigma^-$ **do**
  if $checkCom(\Sigma_i) = 0$ **then**
    $\Delta \leftarrow \Delta \cup \{\Sigma_i\}$
    $End_\Sigma \leftarrow true$
  end
  else
    $\Sigma^- \leftarrow \Sigma^- \setminus \{\Sigma_i\}$
    $\Sigma^+ \leftarrow \Sigma^+ \cup \{\Sigma_i\}$
  end
end

Algorithme 3: $CheckSubsystems(\Sigma^-)$

Inputs : $F_c(faulty\ components)$
Outputs : $\Delta\ F_c, U_c, End_C$
Initialization : $F_c, U_c \leftarrow I$;
foreach $c_i \in F_c$ **do**
  if $checkC(c_i) = 0$ **then**
    $\Delta \leftarrow \Delta \cup \{c_i\}$
    $End_C \leftarrow true$
  end
  else
    $F_c \leftarrow F_c \setminus \{c_i\}$
    $U_c \leftarrow U_c \cup \{c_i\}$
  end
end

Algorithme 4: $CheckComponents(F_c)$

5 Application à l'ATB

Nous présentons le banc d'intégration utilisé chez AIRBUS HELICOPTERS ainsi que le logiciel que nous avons développé pour mettre en oeuvre le méta-diagnostic décrit dans les sections précédentes. Le système avionique étudié est celui de l'hélicoptère NH90 utilisé dans le cadre de missions diverses par plus de vingt pays différents, ce qui correspond à vingt configurations de l'hélicoptère et de son système avionique. Il est composé de deux sous-systèmes principaux : le système "CORE" qui regroupe les fonctionnalités de navigation, de surveillance des équipements, de radiocommunication, et le système "MISSION" qui regroupe les fonctionnalités spécifiques à chaque hélicoptère : secours en mer, vol en terrain hostile, système de contre mesure, d'armement. Un calculateur est le contrôleur du bus d'un sous-système. Il s'agit du Core Management Computer (CMC) pour le système "CORE" et du Mission

Tactical Computer (MTC) pour le système "MISSION". Chaque calculateur est connecté à un ou deux sous-systèmes via des communications sur un bus de données série multiplexé (MIL-STD-1553) et via des communications points à points (ARINC429) et séries (RS-485). Les autres calculateurs du système avionique sont : le Plant Management Computer (PMC) dédié à la surveillance des équipements avioniques de l'hélicoptère, le Multifunction Display (MFD) dédié à l'affichage d'informations pour le vol, la navigation, la gestion et la surveillance des moteurs sur un écran, le Display and Keyboard Unit (DKU) dédié à l'interface de l'équipage avec l'avionique, l'Inertial Reference System (IRS) : la centrale inertielle de l'hélicoptère, le Radio Altimeter (RA) : le radioaltimètre de l'hélicoptère.
Le banc d'intégration sur lequel nous travaillons permet la vérification des fonctions "CORE" du NH90. Il est constitué de deux CMC : c_1 et c_2 ; deux PMC : c_3 et c_4 ; cinq MFD : c_5, c_6, c_7, c_8, c_9 ; deux DKU : c_{10}, c_{11} ; deux IRS : c_{12}, c_{13} ; un RA : c_{14}. Nous notons $COMPS_{ATB} = \{c_i, i \in [1, 14]\}$. Le système avionique testé sur le banc d'intégration $COMPS_{SUT}$ est un sous-système de $COMPS_{ATB}$. Il est représenté à la Figure 7. $COMPS = COMPS_{SUT} = \{c_1, c_2, c_3, c_4, c_5, c_{10}, c_{12}, c_{14}\}$.

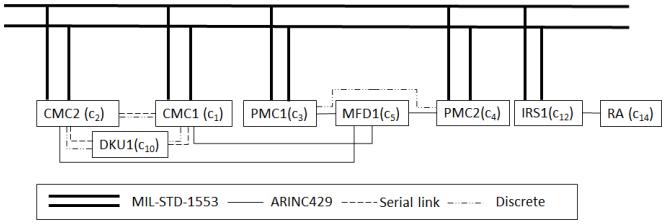

FIGURE 7 – Architecture du sous-système avionique

5.1 Modèle du banc d'intégration et configuration du méta-diagnostic

Pour notre cas d'étude, nous considérons le test de deux fonctions avioniques sur le banc d'intégration. La première consiste à vérifier le fonctionnement du pilote automatique réalisé principalement via les calculateurs RA, $IRS1$, et $MFD1$. Elle est décrite avec la partition p_{NAV}= $\{\sigma_{NAV}$,$\sigma_{\neg NAV}\}$ tel que : σ_{NAV} = $\{$ RA,$IRS1$,$MFD1\}$ et $\sigma_{\neg NAV}$ = $\{CMC1$,$CMC2$,$DKU1$,$PMC1$,$PMC2\}$. La seconde consiste à vérifier l'état des équipements avioniques de l'hélicoptère. Cette fonction est réalisée via l'échange de messages entre les calculateurs avioniques avec le PMC. Elle est décrite avec la partition p_{PERF} = $\{\sigma_{PERF}$,$\sigma_{\neg PERF}\}$ tel que : σ_{PERF} = $\{PMC1$,$PMC2$,RA,$IRS1$,$MFD1\}$ et $\sigma_{\neg PERF}$ = $\{CMC1$,$CMC2$,$DKU1\}$. Le banc d'intégration est aussi décrit grâce aux décompositions structurelles correspondant aux différents protocoles de communication utilisés.

La communication série :
$$\begin{array}{lcl} \sigma_{Serial1} & = & \{CMC1, CMC2, DKU1\} \\ \sigma_{Serial2} & = & \{PMC1, PMC2\} \\ \sigma_{\neg Serial} & = & \{MFD1, IRS1, RA\} \\ p_{Serial} & = & \{\sigma_{Serial1}; \sigma_{Serial2}; \sigma_{\neg Serial}\} \end{array}$$

la communication ARINC :
$$\begin{array}{lcl} \sigma_{ARINC} & = & \{CMC1,CMC2,PMC1,PMC2,MFD1,IRS1,RA\} \\ \sigma_{\neg ARINC} & = & \{DKU1\} \\ p_{ARINC} & = & \{\sigma_{ARINC}\,;\, \sigma_{\neg ARINC}\} \end{array}$$

et la communication MIL-STD-1553 :
$$\begin{array}{lcl} \sigma_{MIL} & = & \{CMC1, CMC2, PMC1, PMC2, IRS1\} \\ \sigma_{\neg MIL} & = & \{MFD1, DKU1, RA\} \\ p_{MIL} & = & \{\sigma_{MIL}\,;\, \sigma_{\neg MIL}\}. \end{array}$$

5.2 Développement du logiciel de méta-diagnostic

L'architecture générale de la suite logiciel de méta-diagnostic développée est décomposée en deux outils distincts : un outil de mise à jour du modèle et un outil de méta-diagnostic du banc d'intégration. Elle est présentée à la Figure 8.

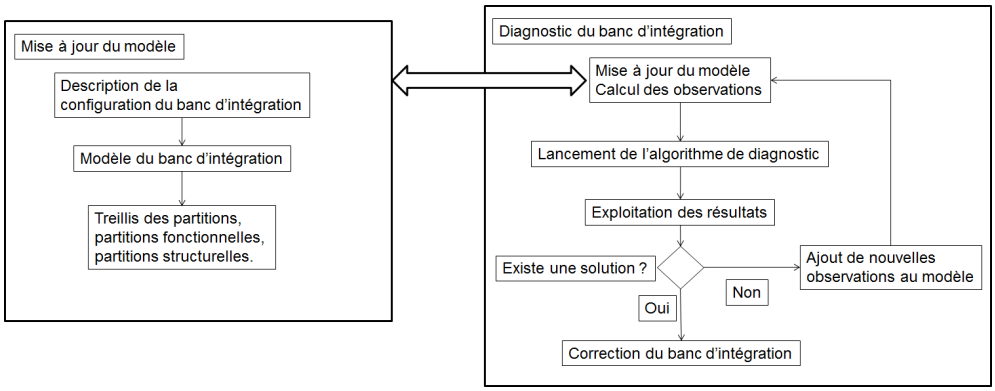

FIGURE 8 – Architecture de la suite logiciel

Cela permet de réaliser un méta-diagnostic complet. Premièrement, l'outil de capitalisation des défauts a été développé pour enrichir une base de données à l'aide d'un historique des pannes. Le second outil permet de lancer l'algorithme de méta-diagnostic avec la modélisation définie par le premier outil.

Les premières fenêtres, en haut de la Figure 9 permettent de définir les composants du banc d'intégration et les fonctionnalités utilisées. Les fenêtres en bas de la Figure 9 donnent les résultats du méta-diagnostic. La fenêtre de gauche donne les composants fautifs et les partitions associées à ce méta-diagnostic. La fenêtre de droite donne les sous-systèmes fautifs et les partitions associées à ce méta-diagnostic.

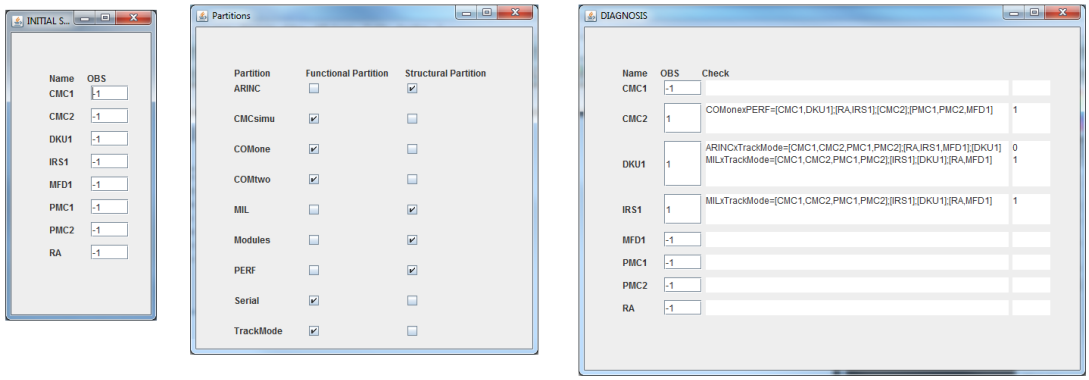

FIGURE 9 – Logiciel de méta-diagnostic

Les deux outils sont complémentaires. Le premier cherche dans un historique. S'il n'est pas suffisant, le second calcule les méta-diagnostics possibles via l'algorithme de méta-diagnostic.

5.3 Itérations et résultats de l'algorithme

La méthodologie consiste à vérifier plusieurs fonctionnalités du système avionique. Les itérations sont définies à l'aide des fonctions disponibles grâce aux équipements présents sur le système avionique. Par exemple, l'utilisation d'une centrale inertielle apporte les informations d'altitude, de vitesse et de position, en supposant qu'une simulation envoie les signaux GPS, ou qu'un GPS est utilisé. Cependant, les signaux ne sont vérifiables que sur un support de données.

Supposons d'abord que l'on vérifie la fonction de navigation sur le bus numérique : $p_{NAV} \times p_{MIL} = 0$, sachant que $p_{NAV} \times p_{MIL} =$ { { RA,$MFD1$ },{ $CMC1$,$CMC2$,$PMC1$,$PMC2$ },{IRS1},{DKU1}}. La vérification des sous-systèmes s'impose sur { RA,$MFD1$ } et { $CMC1$, $CMC2$, $PMC1$, $PMC2$ } par rapport à la fonction de navigation pour les communications MIL-STD-1553. Il s'agit de vérifier les commutateurs dédiés à ce protocole par rapport à la fonction de navigation. La vérification des composants $IRS1$ et $DKU1$ doit aussi être effectuée. Il s'agit de vérifier le téléchargement de leur logiciel embarqué, leur configuration et leur espionnage/simulation par rapport à p_{NAV} et p_{MIL}. Ensuite, si le méta-diagnostic n'a pas abouti, une nouvelle itération est nécessaire.

Les résultats du méta-diagnostic sont donnés par l'intermédiaire du logiciel. Cela consiste à afficher les composants ou sous-systèmes défaillants par rapport à une fonctionnalité et une topologie du système. Sur la Figure 9, les composants fautifs sont affichés sur la fenêtre "DIAGNOSIS", les sous-systèmes sont affichés sur la fenêtre "Communication Diagnosis". La vérification des calculateurs doit résoudre le problème, le méta-diagnostic est $\Sigma = \{DKU1, IRS1\}$. Supposons maintenant que les deux composants soient hors de cause et que $p_{PERF} \times p_{ARINC} = 0$, c'est-à-dire que nous observons en plus un dysfonctionnement de la fonction de surveillance sur le protocole de communication ARINC. Les nouveaux sous-systèmes incriminés sont { $CMC1$,$CMC2$ } et { $PMC1$,$PMC2$,RA,$MFD1$ }. Or, le deux CMC sont localisés dans la même baie avionique. Il s'agit donc de vérifier la configuration de ce bloc par rapport au protocole ARINC et à la fonction PERF. La correction des "pin codings" de la baie résout le problème posé au départ. Le diagnostic est une mauvaise configuration du "pin coding" de la baie contenant les CMC1,2.

6 Conclusion

Le travail présenté dans cet article a mis en évidence les avantages d'une nouvelle méthodologie pour le méta-diagnostic de systèmes complexes : le meta-diagnostic du banc d'intégration avionique. Un haut niveau d'abstraction est nécessaire pour prendre en compte la complexité de tels systèmes. Dans le cadre du diagnostic à base de modèle, une nouvelle représentation sous forme de partitions, de sous-systèmes et de composants a été définie et validée avec un algorithme de méta-diagnostic. L'application de ce formalisme a été développée pour le banc d'intégration d'un hélicoptère. Ceci a permis de réaliser un logiciel de méta-diagnostic d'un système réel. Les résultats sont exploités et permettent une réparation plus rapide du banc d'intégration. L'architecture du logiciel permet le développement de nouveaux algorithmes. Il est aussi possible d'utiliser facilement la solution sur d'autres systèmes fonctionnels. L'utilisation d'autres opérateurs sur le treillis des partitions et une comparaison avec d'autres méthodes ensemblistes sont étudiés. Par exemple, il est possible de propager des fautes d'un composant à une partition en passant par un sous-ensemble grâce à l'addition.

Références

[1] Christopher Watkins and Randy Walter. Transitioning from federated avionics architectures to Integrated Modular Avionics. In *Digital Avionics Systems Conference, 2007. DASC '07. IEEE/AIAA*

26th, pages 1–10, Octobre 2007.

[2] Scott Poll, Ann Patterson-hine, Joe Camisa, David Garcia, David Hall, Charles Lee, Ole J. Mengshoel, David Nishikawa, John Ossenfort, Adam Sweet, Serge Yentus, Indranil Roychoudhury, Matthew Daigle, Gautam Biswas, and Xenofon Koutsoukos. Advanced diagnostics and prognostics testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis*, pages 178–185, Nashville, USA, Mai 2007.

[3] Luca Chittaro, Giovanni Guida, Carlo Tasso, and Elio Toppano. Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems : a case study in diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Man and Cybernetics*, 23(6) :1718–1751, 1993.

[4] Hervé Ressencourt. Hierarchical modelling and diagnosis for embedded systems. *17th International Workshop on Principles of Diagnosis DX'06*, pages 235–242, Juin 2006.

[5] Denis Berdjag, Vincent Cocquempot, Cyrille Christophe, Alexey Shumsky, and Alexey Zhirabok. Algebraic approach for model decomposition : Application for fault detection and isolation in discrete-event systems. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 21(1) :109–125, Mars 2011.

[6] Arnaud Lefebvre, Zineb Simeu-Abazi, Jean-Pierre Derain, and Mathieu Glade. Diagnostic of the avionic equipment based on dynamic fault tree. In *Proceedings of the IFAC-CEA conference*, Octobre 2007.

[7] Gregory M. Mocko and Robert Paasch. Incorporating Uncertainty in Diagnostic Analysis of Mechanical Systems. volume 127, pages 315–325, Mars 2005.

[8] Canh Ly, Kwok Tom, Carl S. Byington, Romano Patrick, and George J. Vachtsevanos. Fault diagnosis and failure prognosis for engineering systems : A global perspective. In *Proceedings of the Fifth Annual IEEE International Conference on Automation Science and Engineering*, CASE'09, pages 108–115, Piscataway, NJ, USA, 2009. IEEE Press.

[9] Fabien Kuntz, Stéphanie Gaudan, Christian Sannino, Éric Laurent, Alain Griffault, and Gérald Point. Model-based diagnosis for avionics systems using minimal cuts. In *DX 2011 22nd International Workshop on Principles of Diagnosis*, pages 138–145, Murnau, Germany., Octobre 2011.

[10] Carine Jauberthie and Elodie Chanthery. Optimal input design for a nonlinear dynamical uncertain aerospace system. Toulouse, France, Avril 2013.

[11] Randall Davis and Walter C. Hamscher. Model-Based Reasoning : Troubleshooting. pages 297 – 346, Juillet 1988. San Francisco, CA, USA.

[12] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3) :197–222, 1992.

[13] Ronan Cossé, Denis Berdjag, David Duvivier, Sylvain Piechowiak, and Christian Gaurel. Meta-Diagnosis for a Special Class of Cyber-Physical Systems : the Avionics Test Benches (Accepted). In *Proceedings of The 28th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems*, IEA/AIE 2015, Seoul, Corea, 2015.

[14] Ronan Cossé, Denis Berdjag, Sylvain Piechowiak, David Duvivier, and Christian Gaurel. Methodology and Application of Meta-Diagnosis on Avionics Test Benches (Accepted). Proceedings of 26th International Workshop on Principles of Diagnosis (DX-15), Paris, FR, 2015.