Diagnostic des SED basé sur un modèle : trois approches évaluées sur une même étude de cas

Mickaël Danancher, Matthias Roth, Jean-Jacques Lesage, Lothar Litz

Diagnostic des SED basé sur un modèle : trois approches évaluées sur une même étude de cas

Mickaël DANANCHER[1,2], Matthias ROTH[1,2], Jean-Jacques LESAGE[1], Lothar LITZ[2]

[1] Laboratoire Universitaire de Recherche en Production Automatisée - ENS Cachan
61, av. du Président Wilson, 94235 Cachan, France

[2] Institute of Automatic Control - University of Kaiserslautern
P.O. Box 3049, 67653 Kaiserslautern, Allemagne

{mickael.danancher; jean-jacques.lesage}@lurpa.ens-cachan.fr,
{litz; mroth}@eit.uni-kl.de

Résumé— **Dans ce papier, trois méthodes de diagnostic des Systèmes à Événements Discrets (SED) sont évaluées. Les approches considérées sont toutes basées sur des modèles : l'approche par *diagnostiqueur* [1], l'approche par *templates* [2] et l'approche par *résidus* [3]. En appliquant ces trois approches à un même système automatisé, leurs performances et limites en terme de complexité de mise en œuvre, détection et isolation des fautes et génération de fausses alarmes sont analysées.**

Mots-clés— **Systèmes à Événements Discrets, Diagnostic des fautes, Détection des fautes, Isolation des fautes.**

I. INTRODUCTION

La productivité d'une entreprise est un enjeu majeur et a des implications économiques importantes. Pour obtenir une productivité accrue, une grande disponibilité des moyens de production est requise. Celle-ci passe notamment par une diminution des temps de remplacement de composants défaillants. Les méthodes de Détection et d'Isolation des Fautes (abrégées dans cette publication par leur acronyme anglais FDI) permettent de détecter un comportement fautif au plus tôt afin de pouvoir stopper le système de production avant la dégradation complète d'un grand nombre de composants ou même la blessure d'un opérateur. Dans un second temps, elles permettent de réduire les délais de remplacement des composants en fournissant à l'opérateur une localisation (isolation d'une faute) plus ou moins précise du (des) composant(s) défaillant(s) à remplacer avant remise en route.

Dans cet article, trois méthodes de FDI pour les Systèmes à Évènements Discrets (SED) basées sur des modèles sont évaluées. Elles ont été appliquées à un même cas d'étude afin de pouvoir en comparer les possibilités et les limites.

La suite de ce papier est organisée ainsi : dans la seconde partie, un rapide aperçu des différentes méthodes de FDI est dressé et les approches considérées dans le cadre de cette étude sont présentées. Le système sur lequel ces méthodes ont été appliquées est ensuite présenté dans la partie III. Dans une quatrième partie sont décrits les modèles qui ont dû être construits pour appliquer chacune des trois approches. La simulation de différents scénarios de comportements fautifs est ensuite présentée ainsi que les résultats obtenus pour chacune des trois approches et leur comparaison. Enfin, nous concluons dans la dernière partie.

II. FDI DES SED : LES APPROCHES RETENUES

A. Aperçu des différentes approches de FDI

Nous retiendrons la classification des méthodes de FDI proposée par Y. Papadopoulos et J. Mc. Dermid dans [4]. Dans cet article, les auteurs proposent de classer les approches de FDI selon trois catégories. Les *approches basées sur les systèmes experts* utilisent des jeux de règles (de type IF-THEN-ELSE ou plus complexes) qui permettent de déclencher des alarmes lorsque certaines conditions sont vérifiées. Les *approches basées sur les données* utilisent des techniques telles que la fouille de données et la reconnaissance de formes pour détecter des comportements fautifs. Les *approches basées sur des modèles* comparent le comportement réel d'un système au comportement de son modèle pour en déduire l'éventuelle occurrence d'une faute.

Dans cette étude, nous n'avons retenu que des approches de FDI basées sur des modèles. Une description plus précise du principe général en est donnée figure 1. Le système à diagnostiquer est considéré comme un système en boucle fermée constitué d'un contrôleur et d'un processus. Du point de vue du contrôleur, les informations échangées dans le système sont des entrées et des sorties (E/S) logiques. Ces E/S sont exploitées en ligne par un algorithme de FDI qui, à l'aide d'un modèle préalablement construit du système, va permettre la détection éventuelle d'une faute puis l'isolation de cette faute.

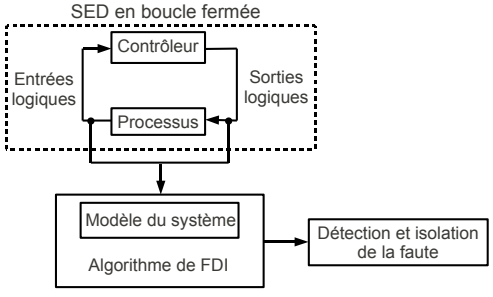


Fig. 1. Principe du FDI des SED basé sur des modèles

Parmi les approches de FDI des SED basées sur des modèles qui ont été proposées depuis une quinzaine d'années dans la littérature, trois ont été sélectionnées pour conduire cette étude comparative : la première approche dite par *diagnostiqueur* est décrite dans [1], la seconde, dite par *templates*, est présentée dans [2] et la dernière, dite par *résidus*, dans [3]. Le choix de ces trois approches a été réalisé en considérant leur complémentarité. En effet :

- l'approche par *diagnostiqueur* est basée sur un modèle incluant les fautes que l'on souhaite détecter alors que les deux autres sont basées sur des modèles du comportement non fautif du système à diagnostiquer,
- les approches par *diagnostiqueur* et par *résidus* sont basées sur des modèles à états alors que l'approche par *templates* est basée sur un modèle événementiel,
- l'approche par *templates* est la seule des trois qui prenne en compte explicitement le temps dans le modèle.

Les trois approches retenues sont brièvement présentées dans la suite. Pour en permettre une meilleure compréhension, une figure présentant le modèle sur lequel est basé le FDI (sous forme d'un exemple) est donnée pour chacune des méthodes.

B. Approche par diagnostiqueur

La méthode par *diagnostiqueur* a été proposée en 1996 par M. Sampath et al. Elle est basée sur une modélisation sous forme d'automates à états finis (AF) de chacun des composants du système (éléments du processus, contrôleur, ...). Ce modèle traduit d'une part tous les comportements normaux du système (comportements non fautifs) et d'autre part le ou les comportements fautifs que l'on souhaite pouvoir détecter. Ensuite, en appliquant les algorithmes proposés dans [1], on obtient le *diagnostiqueur* sur lequel est basé le FDI. Un exemple de *diagnostiqueur* est donné figure 2. Il s'agit également d'un AF défini sur l'alphabet des événements observables, qui permet de détecter des fautes (considérées comme des événements non observables) grâce à trois classes d'états différentes : des états dits "normaux" (ici les états 1N et 8N) dans lesquels il est garanti qu'aucune faute ne s'est produite ; des états dits "Fi-certain" (ici l'état 7F2) dans lesquels il peut être garanti qu'une faute a eu lieu (ici la faute F2) ; des états dits "Fi-incertain" (par exemple : 2N 3F1 4F2 ou encore 5N 6F1) sur lesquels on ne peut pas tirer de conclusion certaine sur l'occurrence d'une faute. Pendant la surveillance du fonctionnement du système en ligne, tant que le *diagnostiqueur* est dans un état normal ou incertain il est supposé qu'aucune faute n'a eu lieu. Dès lors que le *diagnostiqueur* atteint un état Fi-certain, une faute est détectée. Elle est alors forcément isolée car elle a explicitement été traduite dans le modèle.

C. Approche par templates

Dans cette approche, proposée par D. Pandalai et L. Holloway en 2000, le comportement non fautif du système à diagnostiquer est modélisé par une liste de *templates*. Chacun de ces *templates* consiste en une relation causale entre un événement déclencheur et un ou plusieurs événements

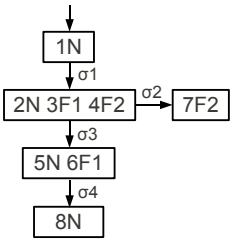


Fig. 2. Un exemple de diagnostiqueur

conséquence. Un exemple de *template* est donné ci-dessous :

$$(e_1, \{(\emptyset, (e_2, [t_{min_2}, t_{max_2}]), w_\emptyset)\})$$

avec e_1 l'événement déclencheur, e_2 l'événement conséquence, $[t_{min_2}, t_{max_2}]$ une fenêtre temporelle. Peuvent également être introduits une condition (ici $\emptyset$) ainsi qu'un tag (ici $w_\emptyset$). La définition complète d'un *template* ainsi que les algorithmes de FDI sont présentés dans [2]. Toutefois, on peut en expliquer brièvement le principe à l'aide de la figure 3. Si l'événement e_1 est observé à la date t_1, alors l'événement e_2 est supposé être observé dans la fenêtre temporelle $[t_1 + t_{min_2}, t_1 + t_{max_2}]$. Si cet événement n'est pas observé dans cet intervalle de temps, alors une faute est détectée. L'examen du *template* impliqué dans la détection de la faute permet ensuite d'isoler la faute. L'événement déclencheur et l'événement attendu sont les deux candidats potentiellement fautifs.

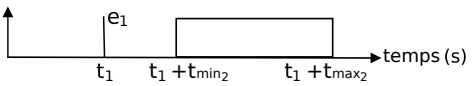


Fig. 3. Illustration d'un template

D. Approche par résidus

Dans cette approche, plus récemment proposée par M. Roth et al. en 2009, un modèle sous forme d'AF est utilisé. Ce modèle doit représenter uniquement le comportement non fautif attendu du système à diagnostiquer. Un exemple d'un tel AF est donné figure 4. La technique par *résidus* mène à la détection d'une faute dès lors que le comportement observé ne peut être reproduit par le modèle [3]. Par exemple (figure 4), si l'événement s3_0 est observé après l'état 3, cela n'est pas reproductible par le modèle, une faute est alors détectée. De plus, cette faute peut ensuite être isolée en calculant les *résidus* qui permettent de caractériser l'écart entre le comportement réel observé du système et celui attendu dans le modèle. Pour ce même exemple, on peut calculer les résidus : Res1 = Comportement observé \ Comportement attendu = {s3_0} et Res4 = Comportement attendu \ Comportement observé = {s1_0, s2_1}. L'union des *résidus* est porteur des candidats potentiellement fautifs (ici {s3_0, s1_0, s2_1}). La définition formelle de ces *résidus* est présentée dans [3].

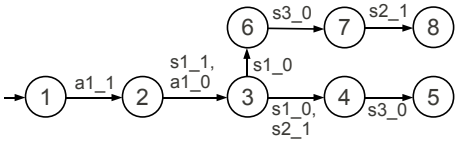


Fig. 4. Un exemple d'AF du comportement sans faute

III. Présentation du cas d'étude

A. Le système étudié

Un système *Pick and Place* (figure 5), proposé comme benchmark par le groupe de travail INCOS [1] a été retenu. Le processus opératif est virtuel, il est émulé sur un ordinateur grâce au logiciel ITS PLC [5], tandis que le contrôleur utilisé est un réel Automate Programmable Industriel (API), câblé à l'émulateur. Le fonctionnement de ce système est le suivant : des pièces sont transportées par le convoyeur A0 jusqu'au poste de prise. Parallèlement, des boites vides pouvant contenir 9 pièces sont transportées par le convoyeur A1 jusqu'au poste de dépose. Les deux vérins double effet horizontaux (A2,A3 et A4,A5) ainsi que le vérin simple effet vertical (A6) permettent de déplacer le préhenseur entre les postes de prise et de dépose. A l'aide de ces vérins et d'un préhenseur magnétique (A7), les pièces sont saisies au poste de prise et déposées dans la boite. Quand la boite est pleine, elle est évacuée. Ce système est par ailleurs muni de 9 détecteurs de position (S0 à S8).

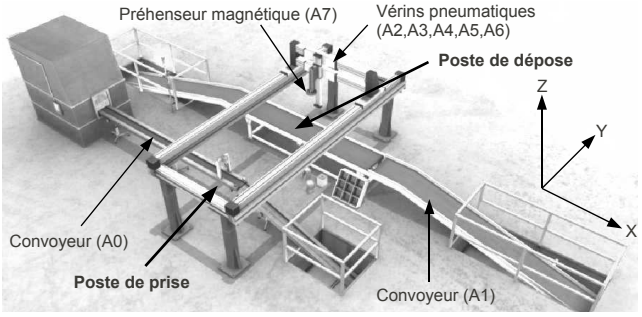


Fig. 5. Le système Pick and Place

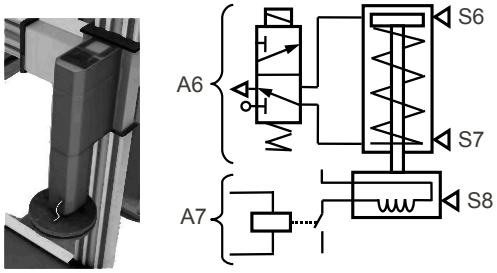
B. Classification des fautes

Afin de simuler différents scénarios de fonctionnement, la classification suivante des fautes à détecter est proposée.

- Les *fautes actionneur* sont des fautes qui impliquent un actionneur (par exemple un vérin ou un moteur), un pré-actionneur (un distributeur ou un contacteur), une connexion (un câble qui relie un pré-actionneur au contrôleur) ou bien la carte de sortie du contrôleur.
- Les *fautes capteur* sont des fautes qui impliquent un capteur (par exemple un capteur de position), une connexion (un câble reliant un capteur au contrôleur) ou bien la carte d'entrée du contrôleur.
- Les *fautes processus* sont des fautes qui impliquent le processus et son environnement (par exemple une boite qui tombe du convoyeur ou bien un opérateur qui ajoute une pièce sur le convoyeur).

Cette classification est basée sur la représentation d'un SED en boucle fermée de la figure 1. Il est par ailleurs supposé que le contrôleur a un comportement non fautif (hypothèse nécessaire pour appliquer la méthode par *résidus*).

En raison de la taille du système considéré, il nous a été difficile de construire un modèle pour la méthode par *diagnostiqueur* (même si la totalité des algorithmes nécessaires à son obtention ont été implémentés pour en minimiser la difficulté). En effet, le premier temps de la construction d'un *diagnostiqueur* consiste à réaliser la composition des AF de processus et de l'AF du contrôleur (très complexe). Dans ce cas d'étude, cette composition conduit à un AF de taille beaucoup trop importante pour qu'il soit aisé à un concepteur de le "mapper" avec la *sensormap*, telle que définie dans [1]. D'autre part, l'introduction de multiples fautes à détecter amplifie encore le problème qui vient d'être exposé. Bien qu'il existe des approches permettant d'appliquer la méthode par *diagnostiqueur* à des systèmes de taille assez importante (notamment des approches distribuées [6]), il a été choisi, dans le cadre de ces travaux, de se concentrer sur les méthodes de base (i.e. monolithique pour l'approche par *diagnostiqueur* ou par *résidus*). Pour toutes ces raisons, il a donc été choisi de restreindre l'étude, dans un premier temps, à un sous-système afin de pouvoir lui appliquer les trois approches. Ce sous-système est décrit figure 6. A6 représente le sous-ensemble du mouvement vertical avec distributeur et vérin simple effet tandis que A7 représente le préhenseur magnétique et son contacteur. S6 et S7 sont des détecteurs de position de fin de course du vérin et S8 est le détecteur de prise de pièce (il émet un signal à 1 quand la pièce est prise, à 0 sinon).



(a) (b)

Fig. 6. (a) et (b) : Le sous-système considéré

De plus, la convention suivante est adoptée : l'événement "front montant" d'un capteur Si (respectivement d'un actionneur Aj) est noté Si_1 (resp. Aj_1), les événements "fronts descendants" sont notés Si_0 et Aj_0.

IV. Construction des modèles

A. Construction du diagnostiqueur

La première étape de la modélisation par *diagnostiqueur* est de construire un modèle de chaque composant du processus (A6, A7) et du contrôleur C sous la forme d'un AF. Ces modèles sont donnés figure 7. Ces AF modélisent le comportement non fautif ainsi que des fautes actionneurs (représentées en pointillés) qui ont été introduites. Ces modèles ont été construits à l'aide de la connaissance d'un expert. Une carte des capteurs (*sensormap*) a par ailleurs été définie, également par connaissance d'expert. Contrairement à l'exemple traité dans [1], seuls les capteurs déjà présents dans le sous-système et utilisés pour la commande ont été considérés pour réaliser cette carte des capteurs et le *diagnostiqueur* qui en découle. Aucun capteur supplémentaire n'a été ajouté (par exemple, il serait envisageable, mais de notre point de vue déraisonnable, d'ajouter un capteur de pression pour le vérin, uniquement dans un objectif de FDI).

1. INgénierie de la COmmande et de la Supervision des SED (http ://www.univ-valenciennes.fr/GDR-MACS/groupes_details.php?gt=INCOS)

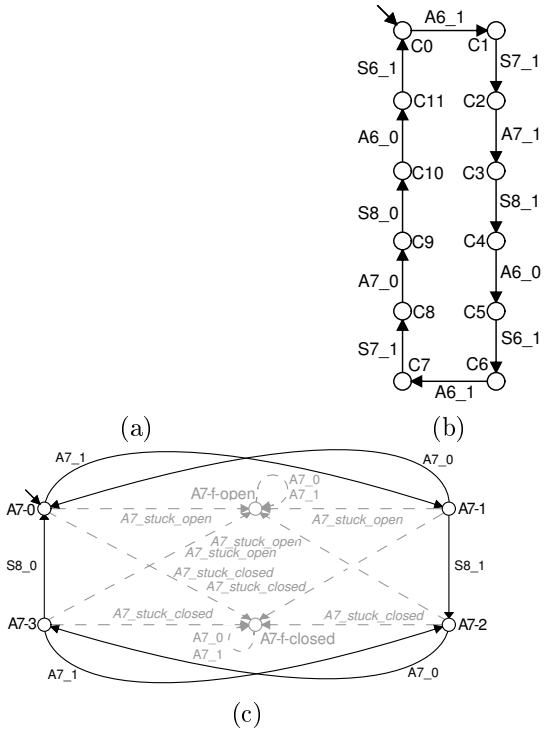


Fig. 7. Modèles du vérin (a), du contrôleur (b) et du préhenseur (c)

Les algorithmes donnés dans [1] ont ensuite été appliqués pour créer le *diagnostiqueur*. Celui-ci possède 48 états et 75 transitions. Seule une partie en est donnée figure 8.

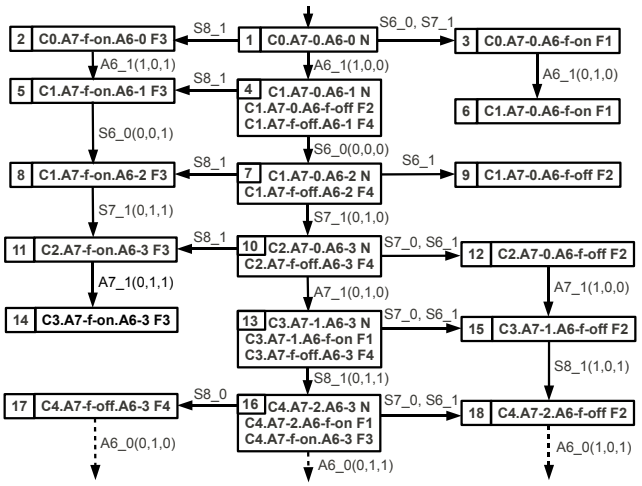


Fig. 8. Un extrait du diagnostiqueur

Il est important de noter que certains choix ont dû être faits, notamment en ce qui concerne les fautes introduites dans le système. Dans le cas présent, seules des fautes actionneur ont été introduites (pour les raisons de complexité décrites au III-B). Par ailleurs, il peut être envisagé d'autres manières de modéliser les comportements fautifs (par exemple : un seul état pour représenter les deux fautes ou bien un état pour chaque faute à partir de chacun des états normaux, i.e. 12 états fautifs au lieu de 2 dans le modèle figure 7 (a)).

B. Construction de templates

La liste des *templates* (table I) a été construite de la manière suivante : l'événement déclencheur est toujours un événement contrôlable (un ordre envoyé par le contrôleur qui est supposé ne pas avoir de comportement fautif) et le ou les événements conséquence sont des événements non contrôlables (le changement de valeur d'un capteur). Ces couples d'événements ont été définis grâce aux relations causales entre événements que l'on peut extraire dans les modèles à états précédemment construits (en ne considérant que les comportements non fautifs). Par exemple dans la figure 7 (a), l'événement contrôlable $A6\_1$ est suivi de deux événements non contrôlables $S6\_0$ puis $S7\_1$. Cette relation se retrouve dans la table I pour les *templates* (1) et (2). Enfin, l'observation de 20 cycles de fonctionnement (supposé non fautif) du système réel commandé a permis de définir la fenêtre temporelle associée à chacun des *templates* ($t_{min} = \bar{t} - 3\sigma$, $t_{max} = \bar{t} + 3\sigma$).

TABLE I

Les templates pour le sous-système

(1)	$(A6\_1, \{$	$(\emptyset, (S6\_0, (0, 0.2]), w_\emptyset)$,
(2)		$(\emptyset, (S7\_1, [1.16, 1.25]), w_\emptyset)\})$
(3)	$(A6\_0, \{$	$(\emptyset, (S7\_0, (0, 0.2]), w_\emptyset)$,
(4)		$(\emptyset, (S6\_1, [1.12, 1.28]), w_\emptyset)\})$
(5)	$(A7\_1, \{$	$(\emptyset, (S8\_1, (0, 0.2]), w_\emptyset)\})$,
(6)	$(A7\_0, \{$	$(\emptyset, (S8\_0, (0, 0.2]), w_\emptyset)\})$,

C. Construction d'un AF représentant le comportement non fautif

Pour modéliser l'ensemble des comportements non fautifs à l'aide d'un AF, il a été choisi d'exploiter également les modèles construits pour le *diagnostiqueur* (figure 7) de manière à ce que les trois modèles sur lesquels repose le FDI expriment des sémantiques cohérentes. Les états et événements représentant des comportements fautifs ont été supprimés des modèles et la composition synchrone sur événements communs des trois AF (C, A7 et A6) a été effectuée. La figure 9 (a) présente le résultat de cette composition.

Il est également possible d'obtenir un tel AF à l'aide d'une méthode d'identification, telle que celle proposée dans [7]. Cette approche a également été appliquée au présent sous-système, un AF comportant 13 états et 14 transitions a été obtenu (voir figure 9 (b)). Ce modèle a également été évalué lors de la simulation de différents scénarios de fautes (cf. section V).

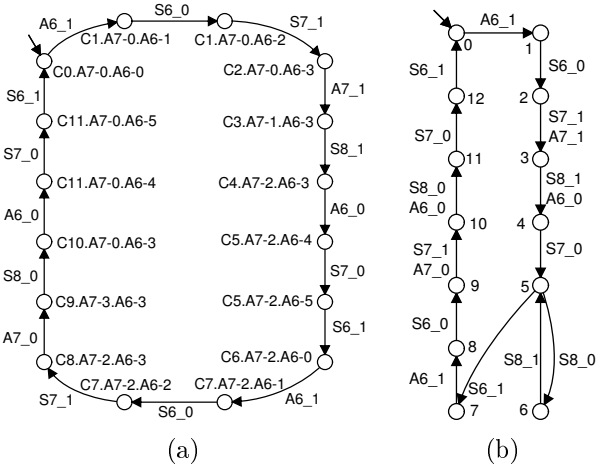


Fig. 9. (a) : Modèle par connaissance d'expert. (b) : Modèle identifié

V. Simulation de différents scénarios de fautes

A. Scenario 1 (faute actionneur)

Description du scénario : la bobine du distributeur du vérin A6 grille alors que le vérin est en train de sortir. Cela entraine le retour du distributeur dans sa position initiale. Même si le contrôleur continue d'envoyer un ordre de sortie au distributeur du vérin (A6 = 1), le vérin retourne à sa position initiale et ne peut plus se déplacer.

Puisque cette faute actionneur a été prévue dans le modèle du *diagnostiqueur* (figure 7 (a)), elle doit donc, selon toute logique, être détectée et isolée par cette approche (théorème de diagnosticabilité démontré dans [1]). La simulation de ce scénario montre bien la détection et l'isolation de cette faute. Le *diagnostiqueur* (voir figure 8) évolue jusqu'à atteindre l'état 9 qui est un état F2-certain, une faute est détectée et F2 signifie qu'il s'agit d'une faute concernant A6 bloqué dans sa position haute.

Concernant l'approche par *templates*, la faute est également détectée car le *template* (2) de la table I n'est pas satisfait. La faute est isolée, les candidats potentiels sont A6 et S7. La localisation est assez précise car l'actionneur A6 fait partie des candidats.

Enfin, concernant l'approche par *résidus*, la faute est détectée (que le modèle ait été construit par connaissance d'expert ou par identification), un comportement qui ne peut être reproduit par le modèle est en effet observé (voir figure 10). La faute est ensuite isolée par calcul des *résidus*. Deux candidats sont obtenus (S6 et S7) pour la modélisation par connaissance d'expert, 3 candidats (S6, S7 et A7) sont obtenus pour la modélisation par identification. Quel que soit le modèle choisi, la localisation est imprécise car l'actionneur A6 ne fait pas partie des candidats.

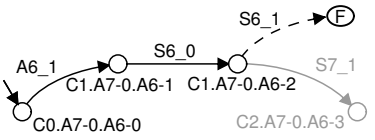


Fig. 10. Évolution de l'AF (obtenu par connaissance d'expert) pour la faute actionneur

B. Scenario 2 (faute capteur)

Description du scénario : le détecteur de prise de pièce S8 est déconnecté au tout début du cycle (le câble de ce capteur est accidentellement arraché ou son connecteur débranché). Le contrôleur ne reçoit plus de signal de ce capteur, la valeur logique qui y est associée reste donc égale à zéro.

Puisque cette faute capteur n'est pas prévue dans le *diagnostiqueur*, elle ne peut théoriquement pas être détectée. Dans le cas précis de ce scénario, sur occurrence de cette faute le *diagnostiqueur* atteint l'état 13 (voir figure 8), un état incertain depuis lequel aucune évolution n'est plus possible puisque le processus est arrêté. Par conséquent la faute n'est pas détectée. Cependant, parmi les nombreux scénarios expérimentés, certaines fautes capteurs ont pu être détectées bien que n'étant pas inclues dans le modèle. Cette détection ne peut toutefois pas être garantie et la localisation qui en découle est incorrecte puisqu'elle implique nécessairement un actionneur.

D'une manière générale, l'approche par *diagnostiqueur* est mal adaptée à la détection de fautes capteur. En effet, les éléments de processus sont modélisés au travers des actions réalisées par les actionneurs (dont les fautes peuvent être prises en compte) et observées par les capteurs. Ces même capteurs sont utilisés, au travers de la *sensormap*, pour construire le *diagnostiqueur*. Considérer des fautes capteur, alors que ceux-ci sont les organes d'observation des fautes révèle donc une difficulté qu'il est difficile de contourner par cette approche. C'est pourquoi le choix a été fait ici de ne pas introduire de fautes capteurs a priori connues dans le *diagnostiqueur*.

Concernant l'approche par *résidus*, cette faute n'est pas détectée non plus. Dans le cas d'un modèle construit par connaissance d'expert (figure 9 (a)), l'état C3.A7-1.A6-3 est atteint. Dans le cas d'un modèle identifié (figure 9 (b)), l'état 3 est atteint. Aucun nouvel événement observé ne fait ensuite évoluer l'AF. Comme aucun nouvel événement (reproductible ou non par l'AF) n'est observé, la faute n'est pas détectée.

L'approche par *templates* permet toutefois de détecter et d'isoler cette faute. Aucun nouvel événement n'étant observé, la prise en compte du temps entraine la violation du *template* (5) de la table I. Deux candidats sont donnés : A7 et S8. La localisation est une fois encore assez précise puisque le capteur S8 fait partie des candidats.

C. Scenario 3 (faute processus)

Description du scénario : une pièce se décroche inopinément pendant son transport du poste de prise au poste de dépose (décrochage dû par exemple à l'accélération du manipulateur).

Une telle faute n'était pas intégrée dans le *diagnostiqueur*. Elle est cependant détectée, mais isolée comme étant une faute du préhenseur qui se bloque en position "préhenseur inactif", ce qui n'est ici pas le cas. Ce scénario est typique de fautes non diagnosticables, mais dans la pratique détectables. La localisation qui en découle est cependant toujours incorrecte.

Concernant l'approche par *templates*, la faute est détectée (*template* (2) de la table I) et 2 candidats sont obtenus (A6 et S7). Ces candidats se révèlent pourtant être très imprécis car ils n'ont un rapport que lointain avec la faute.

Concernant l'approche par *résidus*, la faute est détectée et correctement isolée. Deux candidats sont obtenus (A6 et S8). Le candidat S8 a un rapport direct avec la faute.

D. Scenario 4 (génération de fausses alarmes)

Description du scenario : le système fonctionne sans qu'aucune faute n'apparaisse mais un manque de stabilité de la prise de pièce par le préhenseur magnétique engendre une oscillation de l'état du détecteur de présence pièce S8 (balancement de la pièce sans que celle-ci ne tombe). Ce scénario, non introduit intentionnellement mais observé dans la pratique, permet de mettre en évidence les éventuelles fausses alarmes qui peuvent être générées par les algorithmes de FDI. En effet, comme ce comportement reste acceptable tant que la pièce ne se décroche pas, le considérer comme une faute est excessif.

Avec l'approche par *diagnostiqueur*, une faute est détectée lors de la remontée du vérin (après la prise de la pièce).

Le *diagnostiqueur* se trouve alors dans un état F4-certain, ce qui signifie que le préhenseur est bloqué en position "préhenseur inactif".

Le même problème se pose avec l'approche par *résidus* avec un modèle élaboré par un expert, un comportement non reproductible est observé et une faute est détectée. Par contre, un FDI par méthode des *résidus* à partir d'un modèle obtenu par identification révèle des performances intéressantes car ce comportement spécifique (le mouvement de la pièce) a été observé durant l'identification (voir figure 9 (b), entre les états 5 et 6), il est donc inclu dans le modèle. Par conséquent, puisqu'aucune faute n'est détectée, il n'y a pas d'émission de fausse alarme.

Pour ce scénario aucune fausse alarme n'a eu lieu en utilisant l'approche par *templates* car le comportement observé (l'oscillation de l'état du détecteur S8) ne constitue ni l'évènement déclencheur de l'un des *templates* ni l'événement attendu de l'un des *templates* déclenchés. Dans un cas général, avec cette approche par *templates*, des fausses alarmes ou bien des fautes manquées peuvent cependant survenir à cause du vieillissement du système. Par exemple, les vérins se déplacent plus lentement et les capteurs fin de course délivrent donc l'information plus tard que lors de l'étape de construction du modèle.

E. Évaluation de chaque approche et comparaison

Pour évaluer les performances de chacune des approches relativement à cette étude de cas, 4 critères sont considérés :

- Détectabilité (incluant l'éventuelle diagnosticabilité),
- Précision de l'isolation (certaine ou ambigüe),
- Existence de fausses alarmes,
- Complexité de la mise en œuvre de l'approche.

La détectabilité est définie ici comme la capacité à détecter une faute, alors que la diagnosticabilité est une garantie de la détecter [1]. D'après les simulations effectuées, les fautes actionneur qui ont été introduites dans le modèle du *diagnostiqueur* sont détectables. Elles sont par ailleurs diagnosticables, c'est là leur atout majeur. Les autres catégories de fautes sont non diagnosticables car non introduites dans les modèles. L'expérience a montré qu'elles pouvaient cependant être accessoirement détectées mais l'isolation qui y est associée est incorrecte. Pour les approches par *diagnostiqueur* et par *résidus*, un autre phénomène a été observé, lié aux états après lesquels aucun nouvel évènement observé ne permet de faire évoluer le modèle. Les fautes menant à une telle non-occurrence d'événement ne sont pas détectables avec ces approches. Ce phénomène n'est pas observé pour l'approche par *templates* car la prise en compte du temps permet toujours de faire évoluer le modèle.

Concernant la précision de l'isolation des fautes, la méthode par *diagnostiqueur* permet de localiser une faute sans ambigüité si elle est détectée. Ce n'est pas le cas des autres approches qui ne permettent d'obtenir qu'un ensemble de candidats potentiellement liés à la faute.

L'existence de fausses alarmes pose également un problème car elles vont entraîner des arrêts intempestifs du système de production. Dans les expériences réalisées et avec les trois approches envisagées, elles sont le fait d'une modélisation incomplète du système par connaissance d'expert (ce qui nous semble être inéluctable). La méthode de construction de modèle par identification proposée dans [7] révèle, dans ce cadre, un intérêt particulier puisqu'elle permet effectivement de réduire le nombre de fausses alarmes dans le cas étudié.

Pour évaluer la complexité de mise en œuvre de chaque approche, il est difficile de proposer un critère objectif. Les algorithmes de FDI n'étant pas très difficiles à implémenter, c'est l'étape de modélisation qui influe le plus sur la complexité de la mise en oeuvre. L'expérience a montré que l'approche par *diagnostiqueur* a été la plus complexe dans sa phase de modélisation (notamment la détermination de l'AF du contrôleur et de la *sensormap*). Outre les nécessaires choix concernant la construction des modèles décrivant le comportement non fautif et les comportements fautifs à introduire dans le modèle, cette approche a très vite montré des limites pour un système de taille un peu plus importante. En terme de difficulté de construction d'un modèle, l'approche par *templates* et surtout l'approche par *résidus* avec identification d'un modèle se sont révélées être les plus performantes.

VI. Conclusion

De nombreuses autres expériences de FDI ont été conduites sur ce cas d'étude. Les approches par *résidus* et par *templates* ont même été appliquées avec succès au système complet décrit dans la section III.A. Nous n'avons présenté dans ce papier que quelques uns des scénarios de détection de fautes que nous avons expérimentés en les sélectionnant parmi les plus représentatifs.

Les trois approches possèdent chacune forces et faiblesses. Une solution, permettant d'exploiter au mieux les forces de chacune sans en subir trop les faiblesses, serait d'utiliser ces approches de manière combinée. L'approche par *résidus* pourrait en effet être appliquée en identifiant un modèle du système complet, les quelques fautes critiques qui exigent la diagnosticabilité pourraient être prises en compte dans des *diagnostiqueurs* locaux et enfin les contraintes temporelles majeures pourraient être modélisées à l'aide de *templates*.

Références

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis : Failure diagnosis using discrete-event models. *IEEE Trans. on control systems technology*, 4 (2):105–124, 1996.

[2] D. Pandalai et L. Holloway : Templates languages for fault monitoring of timed discrete event processes. *IEEE Trans. on automatic control*, 45:868–882, 2000.

[3] M. Roth, J.-J. Lesage et L. Litz : A residual inspired approach for fault localization in DES. *In Proc. of the 2nd IFAC Workshop on Dependable Control of Discrete Event Systems, (DCDS'09)*, pages 347–352, 2009.

[4] Y. Papadopoulos et J. McDermid : Automated safety monitoring : A review and classification of methods. *Int. Journal of Condition Monitoring and Diagnostic Engineering Management*, 4(4):14–32, 2001.

[5] B. Riera, P. Marangé, F. Gellot, O. Nocent, A. Magalhaes et B. Vigário : Complementary usage of real and virtual training manufacturing systems for safe PLC training. *In 8th IFAC Symposium on Advances in Control Education, (ACE09)*, 2009.

[6] R. Debouk, S. Lafortune et D. Teneketzis : Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems : Theory and applications*, 10(1–2):33–86, 2000.

[7] M. Roth, J.-J. Lesage et L. Litz : An FDI method for manufacturing systems based on an identified model. *In Proc. of the 13th IFAC Symposium on Information Control Problem in Manufacturing, (INCOM'09)*, pages 1389–1394, 2009.