Approche de détection et d'explication d'erreur de commande par filtrage robuste

Pascale Marangé, Serge Debernard, François Gellot, Marie-Pierre Pacaux-Lemoine, Alexandre Philippot, Thierry Poulain, Bernard Riera, Jean-François Pétin

Approche de détection et d'explication d'erreur de commande par filtrage robuste

Marangé Pascale[1], Debernard Serge[2], Gellot François[3], Pacaux-Lemoine Marie-Pierre [2], Philippot Alexandre[3], Poulain Thierry[2], Riera Bernard[3], Pétin Jean-François[1]

1. *Centre de Recherche en Automatique de Nancy, CRAN, CNRS UMR 7039, Université de Lorraine, Faculté des Sciences, BP 70239, F-54506 Vandoeuvre-les-Nancy Cedex, France,*
{pascale.marange, jean-francois.petin}@univ-lorraine.fr

2. *Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines, LAMIH, CNRS UMR 8201, Université de Valenciennes, Campus du Mont Houy, F-59313 Valenciennes Cedex 9, France*
{serge.debernard, marie-pierre.lemoine, thierry.poulain}@univ-valenciennes.fr

3. *Centre de Recherche en STIC, CReSTIC, Université de Reims Champagne-Ardenne, Moulin de la Housse, BP 1039, F-51687 Reims Cedex 2 - France*
{francois.gellot, alexandre.philippot, bernard.riera}@univ-reims.fr

RESUME. *Ce papier présente les travaux réalisés dans le cadre d'une collaboration entre trois laboratoires – le CRAN, le CReSTIC et le LAMIH - autour de la détection d'erreur de commande. L'objectif de cet article est de proposer une approche de filtrage permettant de se prémunir de toute erreur de commande et de proposer, le cas échéant, une aide à la conduite. Ce papier propose une méthode d'obtention du filtre qui tient compte de l'état de la partie opérative et une génération d'explications pour aider l'opérateur dans la correction d'erreur de commande. L'approche proposée est illustrée sur un exemple de tri de caisses.*

ABSTRACT. *This paper presents the work done between three laboratories: CRAN, CReSTIC and LAMIH, about the control error detection. The objective is firstly to propose a filtering approach to protect the system from any control error and secondly, to propose, if necessary, assistance to understand the error. For this, the authors propose on one hand, a method to design the safety filter that takes into account the plant state and on the other, the generation of explanations to support human operator in correcting control error. The proposed approach is illustrated on an example of sorting.*

MOTS-CLES : *Système à Evénements Discrets (SED), Sûreté de fonctionnement, Filtre de commande, Diagnostic SED, Modélisation, Coopération Homme-Machine, Interface Homme-Machine, Aide au diagnostic.*

KEYWORDS: *Discrete Event System (DES), Dependability, Control filter, DES Diagnosis, Human-Machine Cooperation, Human Machine Interface, diagnostic aid.*

DOI:10.3199/JESA.45.1-n © Lavoisier 2012

1. **Introduction**

Pour répondre aux enjeux de sûreté de fonctionnement dans le domaine des systèmes de contrôle-commande, la norme IEC61508 et ses déclinaisons sectorielles recommandent la mise en place de méthodes formelles permettant de garantir un processus sûr de développement. Les réponses scientifiques suggèrent habituellement deux approches (Faure et Lesage, 2001). Dans la première, l'idée est de synthétiser la commande en utilisant la théorie de supervision de Ramadge et Wonham (Ramadge et Wonham, 1989). Cette solution est très intéressante d'un point de vue théorique mais très difficile à mettre en œuvre dans les automates programmables industriels à cause de l'explosion combinatoire de l'espace d'états. La seconde approche, aujourd'hui la plus utilisée, consiste à vérifier formellement le modèle de contrôle-commande par model-checking (Berard *et al*., 1999) ou par calcul symbolique (Roussel et Denis, 2002). Les inconvénients de ces approches se situent à plusieurs niveaux :

- le contrôle-commande est vérifié à partir de modèles de la commande et non celle réellement implantée, l'implémentation d'un modèle passant par une interprétation du programmeur,
- le contrôle-commande est conçu pour une utilisation donnée. Cela signifie que si un changement est effectué dans le programme de l'automate programmable industriel (API) lors de la production par l'équipe de maintenance ou que l'opérateur a un rôle décisionnel dans le pilotage du système, la vérification du contrôle-commande faite avant l'implémentation n'a plus d'intérêt par rapport au contrôle-commande utilisé réellement.

A ces incertitudes sur la commande réellement implantée, s'ajoutent des incertitudes liées au pilotage par un opérateur humain. En effet, le pilotage par l'Homme comporte des incertitudes sur le bon suivi de la procédure et sur ses prises de décisions, face à des situations mal connues ou mal perçues. Dans ce contexte, être capable d'assurer un niveau de sécurité des personnes, des biens et de l'environnement requiert la mise en place de fonctions de sécurité et d'aide au pilotage. Pour résoudre ce problème, le développement d'approches de surveillance (Chaillet-Subias, 1995 ; Combacau, 1991) inhibant certaines requêtes de l'opérateur, définies comme non conformes aussi bien au niveau de la sécurité qu'au niveau fonctionnel est proposé dans cet article. Allant dans ce sens, plusieurs approches de surveillance du comportement de la partie opérative (PO) ou de la partie commande (PC) ont été développées, par exemple par comparaison avec un modèle de référence du comportement normal du système (Chaillet-Subias, 1995 ; Combacau, 1991), ou bien par comparaison avec une émulation du comportement de la PO (Holloway et Krogh, 1990 ; Roussel et Denis, 2002), ou enfin par filtre interposé entre la PO et la commande (Alanche *et al*., 1986 ; El-Khattabi, 1993 ; Lhoste, 1994 ; Marangé *et al*., 2007).

L'utilisation de telles approches permet donc une protection immédiate des équipements, des personnes, des biens et de l'environnement vis-à-vis d'actions

incohérentes – voire dangereuses – de la part d'un opérateur ou de la partie commande, en détectant et en empêchant les mauvaises manipulations qui pourraient soit détériorer le système, soit contredire le séquencement souhaité par le cahier des charges. La priorité est donc donnée au système technique de filtrage proposé dans cet article plutôt qu'à un opérateur, du fait de la nécessité d'une réponse immédiate de ce filtre, alors qu'un opérateur n'est pas forcément assez rapide pour traiter l'incident, voire n'est pas disponible.

En cas d'une action incohérente de la part d'un opérateur, il est alors nécessaire de lui fournir une explication compréhensible lui permettant de comprendre son erreur et d'améliorer son expertise pour utiliser le système. Lorsque l'action incohérente provient de la partie commande, il est alors nécessaire de fournir à l'opérateur qui va assurer la maintenance de cette commande des explications suffisantes lui permettant de diagnostiquer l'erreur de commande et la corriger. Du fait du figeage de la PO par le filtre, la génération d'explications doit lui permettre de rétablir mentalement des indices suffisants pour assurer cette tâche. Enfin, des pannes sur les capteurs ou les actionneurs de la PO peuvent survenir, et il apparait comme évident qu'un système de diagnostic pourrait d'une part être couplé au filtre de sécurité pour que ce dernier puisse fonctionner correctement et, d'autre part, coopérer avec un opérateur de conduite qui pourrait statuer sur des indécisions issues du diagnostiqueur.

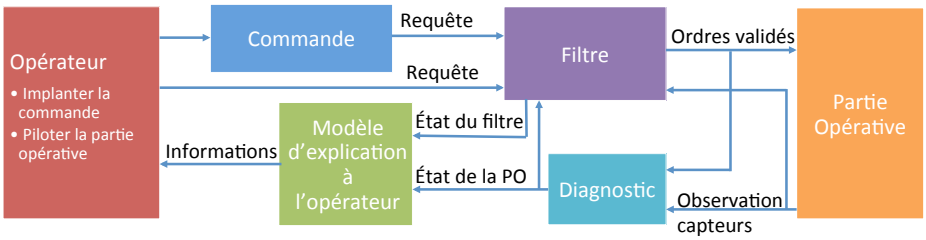

Figure 1 : Projet ADEXEC : introduction d'un filtre de sécurité, d'un module de diagnostic et d'un module d'explication à l'opérateur dans la structure de commande d'une PO

Pour répondre à cette problématique, plusieurs verrous scientifiques ont été identifiés et correspondent à trois modules fonctionnels qui s'intègrent dans la structure classique de commande d'une PO, Figure 1 :

- La mise en place d'une approche formelle pour développer les **filtres** de commande tout en respectant un niveau de SIL donné pour ces fonctions de sécurité. Par exemple à partir du niveau SIL3, la sécurité doit être prouvée formellement.
- L'utilisation d'un **module de diagnostic** pour déterminer l'état réel de la PO, car le résultat du filtrage est fortement dépendant de la fiabilité de celle-ci. En effet, le filtrage va dépendre des informations provenant

de la commande, sur lesquelles le filtre peut agir mais aussi des informations provenant de la PO.

– La mise en place d'un module de **génération des explications** à destination d'un opérateur humain qui devra être compréhensible et suffisamment synthétique vis-à-vis de son niveau de connaissance.

Ces trois points ont été traités en partie dans le projet ADEXEC (Approche de détection et d'explication d'erreur de commande par filtrage robuste) financé par le projet de 3SGS GIS (Groupement d'Intérêt Scientifique - Systèmes de Surveillance, Sûreté de fonctionnement et la sécurité) qui a permis la collaboration de trois laboratoires : le CRAN, le CReSTIC et le LAMIH.

Cet article est structuré en sections présentant les différents apports des contributeurs et l'articulation entre eux. Tout d'abord, il propose une approche de sécurisation matérielle et fonctionnelle par la mise en place d'un filtre et la définition d'une approche de vérification assurant la robustesse de ce filtre. La robustesse du filtre est garantie en utilisant des approches de model-checking et en modélisant le système et l'approche de sécurisation par des automates temporisés. Dans un premier temps, le système sera considéré sans défaillance et fera l'objet de la section 1, puis dans un second temps, les défaillances au niveau de la PO seront prises en compte en utilisant les informations provenant du diagnostic. La définition du diagnostic et la coopération du filtre/diagnostic seront respectivement présentées dans les sections 2 et 3. La prise en compte des informations remontées par le système/commande/filtre, pour apporter une explication compréhensible et adaptée au niveau de l'opérateur humain est ensuite abordée dans la section 4. Ces différentes contributions ont été mises en œuvre et évalués sur un exemple de système manufacturier, ce qui fera l'objet de la section 5. Pour terminer, la section 6 conclut et présente quelques perspectives de recherche.

2. Obtention du filtre robuste

La méthode proposée pour sécuriser les ordres envoyés de la PC vers la PO est basée sur l'utilisation de contraintes de sécurité qui agissent comme un filtre logique s'exécutant à la fin du programme de l'automate programmable industriel (API) et qui interdisent les états dangereux (Figure 2).

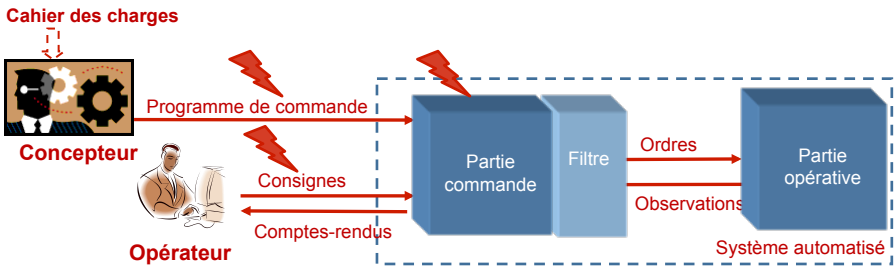

Figure 2 : Principe du filtre de sécurité

Le filtre est un ensemble de contraintes de sécurité (CS) obtenu à partir d'une approche modulaire et itérative de modélisation du procédé et de sa validation. Cela permet de diminuer la complexité, et d'assurer son applicabilité dans le monde de l'industrie. Le filtre est conçu par un expert (Marangé *et al*., 2010 ; Marangé *et al*., 2009) et un ingénieur conçoit la commande de façon classique. Le filtre stoppe le système dans un état sûr dès qu'une contrainte n'est pas respectée.

L'expert mène une analyse dysfonctionnelle (au moyen d'une AMDEC par exemple) pour définir l'ensemble des états dangereux à ne pas atteindre pour les éléments de PO et les produits.

À chaque état dangereux va correspondre un ensemble de contraintes à respecter. L'expert identifie pour chaque état dangereux le sous-ensemble de la PO (les composants et la séquence du flux des produits) à modéliser pour la vérification par model-checking. L'expert définit pour chaque sous-système l'ensemble des contraintes sécuritaires qui constitue le filtre et identifie aussi les observateurs nécessaires à la reconstruction binaire des informations manquantes (non-observabilité d'une pièce entre deux capteurs par exemple).

L'expert procède à la validation de l'ensemble des contraintes pour tous les sous-systèmes en vérifiant que la PO ne peut pas atteindre des états dangereux. L'idée principale est de modéliser par des automates temporisés communicants, d'une part, le comportement des composants et le produit pour chaque interaction avec la commande la plus permissive et, d'autre part, le fonctionnement de l'API. En effet, aucune hypothèse n'est faite sur la commande et les changements possibles des sorties sont étudiés au cours du temps de cycle automate. Ensuite, le model checker UPPAAL (Behrmann *et al.*, 2004) vérifie que l'ensemble des contraintes permet au système d'éviter d'atteindre un état dangereux. Pour cela, différentes propriétés sont utilisées : A[ ] p – " p toujours vrai", E<> p – "p atteignable", A<> p – "passage par p"…. Ce point n'est pas approfondi dans cet article mais de plus amples informations sont données dans (Marangé *et al*. 2009).

Les notations utilisées dans la suite de l'article seront les suivantes :

- *t*: l'instant courant (vue de l'API), t-1 le cycle précédent de l'API,
- $s_k=s_k(t)$: variable logique qui correspond à la valeur de la $k^{ème}$ Sortie booléenne (actionneur) de l'API à l'instant t,
- $S^*k=Sk(t-1)$ variable logique qui correspond à la valeur de la $k^{ème}$ Sortie booléenne (actionneur) de l'API à l'instant t-1,
- ej=ej(t) : variable logique qui correspond à la valeur de la $j^{ème}$ Entrée (capteur) de l'API à l'instant t,
- $E^*j=Ej(t-1)$: variable logique qui correspond à la valeur de la $j^{ème}$ Entrée (capteur) de l'API à l'instant t-1,
- ".", "+", "¯" correspondent respectivement aux opérateurs logiques ET, OU et NON (complémentation),

- Σ et Π sont respectivement la somme logique (OU) et le produit logique (ET) de variables logiques,
- ↑x correspond au front montant d'une variable booléenne x (dans l'API, $\uparrow x = \overline{x^*}.x$),
- ↓x correspond au front descendant d'une variable booléenne x (dans l'API, $\downarrow x = x^*.\bar{x}$),
- S_{API} est l'ensemble des variables $S_{k\ API}$ aux instants t, t-1, t-2 … ,
- E_{API} est l'ensemble des variables E_j aux instants t, t-1, t-2 … ,
- Obs est l'ensemble des observateurs aux instants t, t-1, t-2 … ,
- CS est l'ensemble des contraintes de sécurité.

Les contraintes de sécurité s'expriment dans cette approche sous la forme d'une fonction logique (monôme, produit Π de variables logiques) devant être égale à 0 à chaque cycle de l'API pour que la contrainte soit respectée. A l'état initial, on considère l'état de repos à 0 pour les actionneurs (Sk) comme sûr. A la fin du cycle API, le système est considéré comme sûr de fonctionnement si et seulement si toutes les contraintes de sécurité sont respectées.

Les contraintes doivent être définies pour assurer la commandabilité du système. En d'autres termes, il est possible de réaliser une commande permettant de respecter le cahier des charges. Par exemple, un filtre qui met toutes les sorties à 0 est sûr mais n'assure pas la commandabilité.

Certaines contraintes font intervenir une seule sortie à l'instant t (contraintes dites simples CSs) alors que d'autres en font intervenir plusieurs (contraintes dites combinées CSc).

Les contraintes nécessitent la connaissance des Entrées/Sorties (E/S) à l'instant courant et, éventuellement, aux instants précédents (présence d'un front montant ou descendant par exemple). Le filtre a donc une fonction mémoire. Il peut être aussi nécessaire de définir des observateurs compte tenu du manque d'observabilité du système. Cela est particulièrement vrai lorsqu'il y a des flux de produits. Les observateurs correspondent à une fonction séquentielle des entrées de l'API et permettent idéalement de se ramener à une contrainte logique combinatoire. L'ensemble des contraintes CS est considéré comme nécessaire et suffisant pour garantir la sécurité du système.

Dans notre approche, on suppose que les contraintes de sécurité peuvent toujours être représentées sous la forme d'un monôme et dépendent des entrées (à t, t-1, t-2, …), des sorties (à t, t-1, t-2, …) et d'observateurs (uniquement fonction des entrées à t, t-1, t-2, …). La figure 3 représente l'architecture du filtre.

Les contraintes de sécurité CSs et CSc peuvent être représentées respectivement par les équations (1) et (2) qui sont des fonctions logiques (Riera *et al.*, 2014).

$\forall m \in [1, \mathrm{N}_{\mathrm{CSs}}],\ \exists! k \in [1, \mathrm{N}_S]$ /

$$CSs_m = \prod(s_k, E_{\text{API}}, \text{Obs}, \text{S}^*) = 0 \quad (1)$$

$\forall n \in [1, \text{N}_{\text{CSc}}],\ \exists!\ (k, l, \dots) \in [1, \text{N}_\text{S}]\ avec\ k \neq l \neq \cdots /$

$$CSc_n = \prod(s_k, s_l, \dots, E_{\text{API}}, \text{Obs}, \text{S}^*) = 0 \quad (2)$$

N_s est le nombre de sorties. N_{CSs} et N_{CSc} sont respectivement le nombre de contraintes de sécurité simples et le nombre de contraintes combinées. Il y a seulement deux types de contraintes de sécurité simples (CSs) parce qu'elles s'expriment sous la forme d'un monôme (h_{0m}, h_{1m}) et qu'elles font intervenir une seule sortie à l'instant courant t (équation (3) ou (4)) :

$$\forall m \in [1, \text{N}_{\text{CSs}}],\ \exists!\ k \in [1, \text{N}_\text{S}] /\ \ CSs_m = s_k . h_{0m}(E_{\text{API}}, \text{Obs}, \text{S}^*) \quad (3)$$

$$\text{xor}\ \ CSs_m = \overline{s_k} . h_{1m}(E_{\text{API}}, \text{Obs}, \text{S}^*) \quad (4)$$

Ces contraintes simples expriment le fait que si $h_{0m}(E_{\text{API}}, \text{Obs}, \text{S}^*)$ est égale à 1, s_k doit être nécessairement égale à 0 ($s_k . h_{0m}(E_{\text{API}}, \text{Obs}, \text{S}^*)$ ou égale à 1 ($\overline{s_k} . h_{1m}(E_{\text{API}}, \text{Obs}, \text{S}^*)$ respectivement pour garantir que la contrainte soit égale à 0.

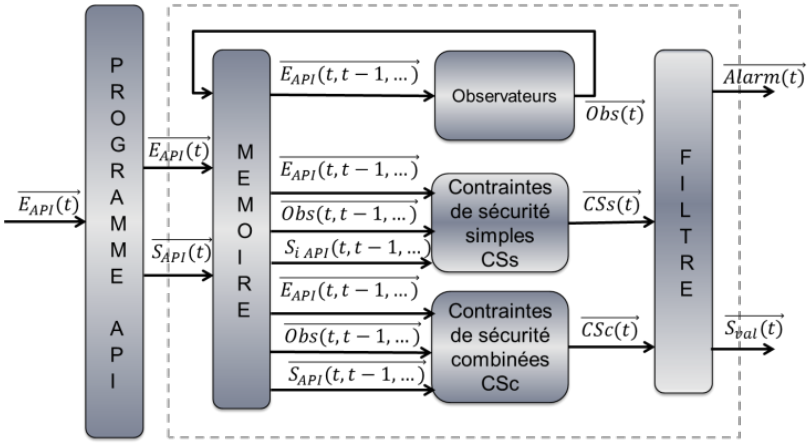

Figure 3 : Architecture du filtre

Pour chaque sortie s_k, il est possible d'écrire la somme des contraintes logiques selon l'équation (5) où f_{s0k} et f_{s1k} sont des polynômes (sous forme $\sum\Pi$) fonctions de E_{API} et de Obs aux instants t, t-1, t-2, … L'équations (5) doit alors être toujours fausse car les contraintes de sécurité doivent être fausses en permanence pour tout cycle API.

$$\sum_{i=1}^{N_{CSs}} CSs_i = \sum_{k=1}^{N_s} \left(s_k . f_{s0k}(E_{\text{API}}, \text{Obs}, \text{S}^*) + \overline{s_k} . f_{s1k}(E_{\text{API}}, \text{Obs}, \text{S}^*)\right) = 0 \quad (5)$$

Il est important de noter que les contraintes de sécurité simples doivent respecter la propriété mathématique suivante :

$$f_{s0k}(E_{\text{API}}, \text{Obs}, \text{S}^*) . f_{s1k}(E_{\text{API}}, \text{Obs}, \text{S}^*) = 0 \quad (6)$$

Dans le cas contraire, cela signifie que 2 CSs sont en contradiction et au moins l'une des deux n'est pas respectée et l'ensemble des contraintes donc n'est pas cohérent. Si la propriété n'est pas respectée logiquement ou à cause d'une situation incohérente du système (par exemple deux capteurs indiquant qu'un vérin est rentré et sorti ne peuvent être activés simultanément), le model-checker ne validera pas l'ensemble des contraintes.

On peut noter que si $f_{s0k}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*)=0$ ou si $f_{s1k}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*)=0$, la propriété est logiquement vérifiée. En complément, si toutes les CSs impliquant s_k sont écrites avec des fronts montants et des fronts descendants de s_k, la propriété est toujours vraie (condition suffisante).

Démonstration :

Si CSs est seulement écrite avec des fronts montants et des fronts descendants, on peut noter que :

$$\mathrm{f_{s0k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) = \overline{s_k^*}.\mathrm{f_{s0k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) \text{ et } \mathrm{f_{s1k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) = s_k^*.\mathrm{f_{s1k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) \quad (7)$$

En conséquence, comme à l'instant n-1, le front $\overline{\mathrm{Sk}^*}.\mathrm{Sk}^* = 0$, la propriété est respectée à l'instant suivant.

$$\mathrm{f_{s0k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*).\mathrm{f_{s1k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) = \overline{s_k^*}.\mathrm{f_{0k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*).s_k^*.\mathrm{f_{1k}}(E_{\mathrm{API}}, \mathrm{Obs}, \mathrm{S}^*) = 0 \quad (8)$$

Lorsqu'une ou plusieurs contraintes de sécurité apparaissent, il nous faut mettre le système en repli dans une situation sûre. Pour ce faire, on calcule pour chaque sortie une valeur qui satisfasse les CSs et les CSc avec une priorité à la mise à zéro des sorties. La méthode de calcul n'est pas présentée dans le cadre de cet article.

Après avoir décrit la méthode permettant d'obtenir un filtre de sécurité robuste, la section suivante présente une approche de diagnostic permettant d'évaluer l'état de la PO, afin d'enrichir le filtre et lever ainsi l'hypothèse d'une PO non défaillante.

3. Obtention du diagnostic

Le domaine de diagnostic est un aspect important dans l'ingénierie des systèmes. Cette importance n'est pas due uniquement à la sûreté de fonctionnement mais aussi au besoin d'atteindre les objectifs de la maintenance afin de garantir un bon rendement et une bonne qualité de fonctionnement du système. L'objectif de cette section est de présenter succinctement une approche décentralisée de diagnostic des SEDs et prendre ainsi en compte l'état réel de la PO dans les informations utilisées dans le filtre.

Afin d'assurer un fonctionnement correct et sûr des systèmes dynamiques, le diagnostic de défauts dans les SEDs a reçu une attention considérable ces dernières années. La notion de diagnostiqueur (ou « diagnoser »), observateur du système, n'est apparue que bien plus tard lors des travaux de (Sampath, 1995). C'est également dans ces travaux que la notion de « diagnosticabilité » est apparue, permettant de garantir le diagnostic d'un défaut dans un délai fini.

Les approches de diagnostic des SEDs que nous pouvons rencontrer dans la littérature peuvent être classifiées selon le type de connaissances disponibles à la construction du module de diagnostic. Il est possible alors de distinguer les méthodes « sans modèle » des méthodes « à base de modèle ».

Les méthodes sans modèle font appel à la disponibilité des données issues des enregistrements réalisés tout au long du fonctionnement du système. Elles sont issues très souvent des systèmes experts (Tzafestas et Watanabe, 1990 ; Alonso-Gonzalez *et al*., 2010). Par conséquent, l'acquisition de connaissances auprès d'experts peut s'avérer difficile et fastidieuse avant de disposer de suffisamment de connaissances afin de réaliser un diagnostic fiable.

Le principe général des méthodes à base de modèle est de détecter une divergence entre le comportement attendu représenté par un modèle du système et le comportement observé représenté par les informations (mesures réelles) fournies par les capteurs (Reiter, 1987 ; Roth *et al*., 2009).

Les approches avec représentation des défauts dans le modèle constituent une grande part des travaux de la littérature. Elles consistent à inclure explicitement dans le modèle, l'ensemble de tous les défauts possibles du système. Dans ces approches, l'objectif est la construction d'un diagnostiqueur qui pourra prendre une décision concernant l'état du système à un instant donné sous forme d'étiquettes (Debouk *et al*., 2000 ; Genc et Lafortune, 2003). Initialement proposées par (Sampath, 1995), ces approches consistent à réaliser un modèle du comportement du système représentant les deux comportements, normal et défaillant. Ce diagnostiqueur est un observateur étiqueté fournissant une information sur le comportement du système. Ces approches sont exclusivement discrètes dans le sens où aucune autre information que celles données par les capteurs et actionneurs Tout ou Rien est présente. Il est cependant parfois nécessaire d'enrichir la connaissance du système, et cela notamment au travers d'informations temporelles ou temporisées. Des approches à base de modèles utilisant des « templates » ou « chroniques » ont alors été développées (Pandalai et Holloway, 2000 ; Milne *et al*., 1994).

La structure du modèle de diagnostic est également un paramètre important. Pour une structure centralisée, le système est représenté par un modèle global associé à un diagnostiqueur unique. Celui-ci collecte des observations afin de prendre une décision finale sur l'état du système (Sampath, 1995). Ces approches se basant sur la construction d'un diagnostiqueur global souffrent de quelques inconvénients et, notamment d'explosion combinatoire suite à la composition des sous-modèles. Cette problématique limite considérablement l'applicabilité de ces approches dans le cas des systèmes complexes à grande taille. Les structures décentralisées permettent de compenser cette explosion combinatoire (Su *et al.*, 2002). Dans cette architecture, le système est divisé en plusieurs parties locales. Les diagnostiqueurs locaux opèrent alors de manière indépendante, sans avoir d'échanges entre eux. La décision finale concernant l'état global du système est obtenue par fusion des décisions ou à l'aide d'un coordinateur (table de décision pour la levée d'ambiguïtés). Enfin, la structure distribuée consiste à voir le système comme un ensemble de composants locaux associés à plusieurs diagnostiqueurs locaux, chacun recevant des observations à

partir d'un composant local spécifique dont il est responsable. Dans cette structure, les résultats locaux de diagnostic sont obtenus grâce à un canal de communication qui permet l'échange d'informations entre les différents diagnostiqueurs locaux (Pencolé *et al*., 2001). Il n'est donc pas nécessaire de construire un modèle global du système.

Une approche de diagnostic décentralisé est proposée autour d'une modélisation par composants pour des systèmes manufacturiers composés de capteurs et d'actionneurs discrets. L'originalité de la proposition consiste à éviter l'étape de modélisation globale de la PO notée G (par exemple, obtenue par composition synchrone des modèles locaux) et éviter ainsi les risques d'explosion combinatoire. Les diagnostiqueurs locaux sont obtenus à partir de la modélisation locale d'une partie du composant, appelée partie de la PO (*PoP*).

Dans les procédés industriels, un système de production est défini comme une chaîne fonctionnelle composée d'une PC qui émet des signaux de commande vers la PO et reçoit les valeurs des capteurs. La PO représente la partie mécanique tandis que la PC est la partie logique qui décrit le comportement désiré. Cet échange d'informations entre la PO et la PC représente les seules informations observables en ligne. Donc, un diagnostiqueur peut être défini comme un observateur du système qui utilise ces informations pour reconstruire les évolutions du comportement grâce à un modèle du comportement normal ainsi que des évolutions conduisant à des défauts. Dans l'état de l'art, nous avons indiqué qu'une approche centralisée est difficile à appliquer pour des systèmes complexes. C'est pour cela que nous nous sommes tournés vers une approche décentralisée malgré la difficulté de détermination du niveau de décomposition modulaire de façon générique.

Un système de production est composé d'éléments (actionneurs / capteurs) qui interagissent entre eux ou non. Chaque composant peut être modélisé par son comportement normal en tant que partie de la PO. Ces modèles *PoPs* tiennent compte des spécifications techniques pour obtenir des modèles réalistes (Philippot *et al*., 2010). Par conséquent, un composant i peut être modélisé par un automate $PoP_i = (X_i, \Sigma o_i, \delta e_i, x_{i0}, I_i)$ où X_i est l'espace d'état, Σo_i est l'ensemble des événements observables, δe_i est la fonction de transition, x_{i0} est l'état initial et I_i est l'ensemble des intervalles de temps où les fonctions de transition sont attendues. Une fonction de transition δe_i correspond à une expression logique attendue dans un intervalle de temps I pendant lequel un événement doit se produire.

Un diagnostiqueur local est obtenu après identification de tous les défauts possibles pour chaque état normal de chaque composant. Il s'agit d'une analyse faite par un expert qui permet de définir l'ensemble des défauts associés à une étiquette du diagnostiqueur. Une partition de défauts Π_{Fj} est associée à plusieurs étiquettes indiquant le type de défaillance. Pour modéliser les défauts dans les diagnostiqueurs, les deux hypothèses suivantes sont envisagées :

- un seul défaut par composant peut se produire au même instant,
- la PC est supposée fiable et sûre, par conséquent, la PC ne peut pas être responsable d'un défaut. Cette hypothèse sera levée dans la partie

suivante, grâce à la coopération entre le diagnostic et le filtre.

Pour déterminer tous les candidats possibles responsables d'un comportement défaillant dans une *PoP*, la connaissance peut provenir d'une analyse d'experts et/ou des documents tels que des modes de défaillance et analyse leurs effets (AMDEC).

Les défauts peuvent être modélisés comme observables et/ou non observables (Tableau 1). Le cas des défauts observables semble trivial mais il est en fait souvent source d'erreur ou d'ambiguïté de décision. La problématique étant d'identifier la cause réelle d'une observation, nous continuerons donc à traiter ce cas.

Tableau 1. Partition de défauts

Défauts capteurs observables	Passage inattendu de la valeur capteur de 0 à 1
	Passage inattendu de la valeur capteur de 1 à 0
Défauts capteurs non observables	Capteur bloqué à 0
	Capteur bloqué à 1
Défauts actionneurs non observables	Actionneur bloqué à 0
	Actionneur bloqué à 1

A partir de la modélisation du comportement normal d'une PoP_i et de la partition de défauts associée, un diagnostiqueur local peut être représenté par l'automate $D_i = (X_i \cup XDF_i, \Sigma o_i, \delta_i, x_{i0}, I_i, l_i)$ avec X_i, Σo_i, δe_i, x_{i0} et I_i tels que définis dans les automates PoP_i. XDF_i est l'ensemble des états anormaux, $\delta_i : X_i \times \Sigma_i^* \rightarrow X_i \cup XDF_i$ est la fonction de transition avec les fonctions attendues (δe_i) et inattendues (δu_i) d'un état x et l_i est l'ensemble des fonctions de décision du diagnostiqueur local D_i avec $l_i(x)$ la fonction de décision de l'état x qui peut être une ou plusieurs étiquettes de défaut $\{F_j\}$ ou de type normal $\{N\}$, lorsqu'il n'y a pas de défaut. La prise de décision locale s'effectue de la manière suivante :

- si la fonction de l'étiquette d'un état x est $l_i(x)=\{N\}$, le diagnostiqueur décide avec certitude de la non-présence de défauts,
- si la fonction de l'étiquette est $l_i(x)=\{F_j\}$, le diagnostiqueur indique avec certitude la présence d'un défaut du type F_j,
- si la fonction de l'étiquette est $l_i(x) = \{N, F_j\}$, le diagnostiqueur ne peut décider et le système est dans un cas d'ambiguïté.

Pour définir les fonctions inattendues (δu_i), il est possible de définir toutes les fonctions de transition pour les 2^n possibilités, n étant le nombre d'événements et d'intervalles. Les intervalles sont nécessaires pour améliorer la diagnosticabilité du système. Pour chaque occurrence d'événement, un intervalle de tolérance est également défini (Philippot *et al*., 2010).

L'information recueillie par le diagnostic permet ainsi de connaître l'état de la PO mais également et surtout le composant défectueux en cas de défaut.

4. Coopération filtre / diagnostic

L'utilisation conjointe du filtre de commande et du diagnostic va permettre, d'une part de simplifier le modèle du diagnostiqueur en éliminant les transitions inhibées par la présence du filtre de commande et, d'autre part, d'améliorer la robustesse du filtre qui peut être complété par des informations provenant du diagnostic.

Les modèles de diagnostic sont définis à partir du comportement normal d'une *PoP* et une hypothèse forte suppose que la PC est fiable et sûre. L'utilisation d'une approche de filtre (Marangé *et al.,* 2007) permet de lever cette hypothèse et ainsi donc de simplifier les modèles de diagnostic. La prise en compte du filtre pour simplifier les diagnostiqueurs peut se faire de manière automatique en utilisant l'algorithme de Kumar adapté aux spécifications sous forme de contraintes logiques proposées par (Marangé *et al.*, 2008). Par conséquent, l'utilisation du filtre permet de simplifier la construction des diagnostiqueurs car celui-ci dépend de la granularité des modèles locaux et des performances du filtre de commande.

L'ensemble initial de contraintes peut être enrichi en prenant en compte les informations des diagnostiqueurs. En effet, lorsqu'une défaillance survient sur un capteur ou un actionneur, les contraintes du filtre qui contiennent les variables logiques associées aux dispositifs défectueux deviennent erronées. En effet, une évolution de la commande autorisée normalement peut être interdite et, pire encore, une évolution interdite peut alors devenir autorisée. Par conséquent, les contraintes de filtrage doivent tenir compte du fait qu'il y a une défaillance ou non. Pour chaque partition de défauts, un flag sera mis à vrai lorsque le diagnostiqueur atteindra un état de la défaillance (état sans transition de sortie). Ce flag détermine si la variable considérée peut être utilisée dans la contrainte du filtre (si le flag est faux) ou si une des informations reconstruites doit être utilisée (si le flag est vrai). Dans ce dernier cas, une information temporisée est introduite dans la contrainte à la place de l'information défectueuse.

C'est à partir des informations issues du filtre et du diagnostiqueur qu'une explication peut alors être retournée à l'opérateur s'il est nécessaire qu'il intervienne.

5. Génération d'explication

L'intégration d'un filtre de sécurité, permet d'augmenter la sécurité des produits et de la PO, mais introduit une problématique facteur humain pour les opérateurs. En effet, lorsque le filtre déclenche suite à une requête erronée issue de la PC ou d'un opérateur de production, la PO est figée et il est alors très difficile de comprendre ce qui se passe, et de diagnostiquer correctement l'erreur ou la mauvaise commande. Dans le cadre de ce projet, il était donc nécessaire de :

- concevoir un module d'explications à l'opérateur humain suite au déclenchement du filtre de sécurité,

- valider expérimentalement les niveaux d'explications proposés,
- définir les interactions possibles avec le filtre de sécurité.

Les premiers travaux réalisés ont consisté à cerner de manière exacte le rôle des opérateurs humains. Hormis l'opérateur de conception – celui qui crée la commande qui peut être validée hors ligne – deux « types » d'opérateurs peuvent être distingués :

- un opérateur de production, en lien direct avec la PO et qui peut, en parallèle d'un automatisme, agir sur celui-ci et donc commettre des erreurs,
- un opérateur de maintenance en charge de corriger la PC d'un automate si celle-ci s'avère défaillante.

Les tâches de ces deux types d'opérateurs ne sont pas les mêmes et peuvent conduire à des utilisations différentes du filtre tel qu'il est défini actuellement. Concernant les erreurs issues d'opérateurs de production, on peut prendre le parti pris de donner la priorité au filtre de sécurité, l'impact n'étant au final qu'un arrêt de production. Il reste néanmoins nécessaire de fournir à ces opérateurs suffisamment d'explications pour qu'ils comprennent leurs erreurs et redémarrer la production le plus vite possible. Dans le cas d'une panne d'un élément de la PO, ces opérateurs peuvent aussi enrichir le diagnostic si ce dernier n'est pas capable de déterminer quel élément est en panne. Concernant les opérateurs de maintenance, leur rôle consiste ici à assurer la correction des programmes implémentés sur API, suite au déclenchement du filtre. Ils doivent donc assurer le diagnostic de la commande pour ensuite la corriger. C'est sur ce deuxième aspect que l'étude s'est focalisée pour le moment.

Selon Ribert-Van De Weerdt et Brangier (2000), l'activité de maintenance en détection de pannes de composant comporte trois phases.

- La première phase consiste en la préparation de l'intervention, i.e. l'organisation de l'intervention sur le système selon un pré-diagnostic. Ce dernier s'effectue à partir d'informations diverses, vagues, incomplètes qui donnent différents indices et qui orientent le travail futur et permet d'organiser l'intervention en elle même.
- La deuxième phase est l'intervention proprement dite qui comprend le diagnostic sur lequel nous allons revenir et l'exécution des travaux.
- Enfin la dernière phase se situe après l'intervention et consiste à contrôler l'action réalisée, d'une part, et à rendre compte de cette intervention par compte-rendu par exemple, d'autre part.

Selon les auteurs, la deuxième phase de diagnostic se construit en plusieurs étapes (Falzon, 1987; Caverni *et al.*, 1990) : (i) l'acquisition d'indices par exploration de l'environnement et leur catégorisation grossière, (ii) l'affinement de cette catégorisation où des hypothèses sont posées et réduites progressivement, (iii) la localisation du composant défectueux et (iv) enfin une étape de validation. Selon Rasmussen (1986), les processus cognitifs des agents de maintenance sont dirigés soit par le fonctionnement normal du système, soit par le fonctionnement anormal et

les recherches peuvent être fonctionnelles (fonctions réalisées par un composant ou un sous-système) ou topographiques.

L'enjeu majeur de l'aide qui peut être apportée est dû au fait que, lorsque le filtre de sécurité se déclenche, toute évolution de la PO est figée pour assurer la sécurité des installations et/ou des produits. De ce fait, l'opérateur de maintenance ne dispose plus des conséquences d'une commande erronée alors que, sur un système qui ne disposerait pas de filtre, la PO serait dans une situation dégradée telle que certains indices auraient permis d'engager une activité de diagnostic (Figure 4). L'aide apportée doit donc permettre de reconstruire de la manière la plus fidèle possible ces indices, voire d'aller plus loin en proposant des pistes de recherche.

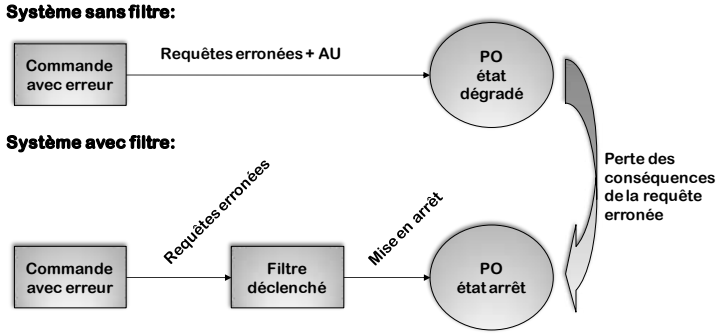

Figure 4 : Impact du filtre sur l'état du procédé

Dans ce cadre, en considérant que la commande est implémentée dans un API sous forme de SFC et/ou de schémas à contacts (Ladder), les erreurs de commande possibles recensées peuvent être :

- un non respect des contraintes de fonctionnement de la PO,
- des erreurs liées aux réceptivités des GRAFCETs,
- des erreurs liées à l'activation d'un actionneur.

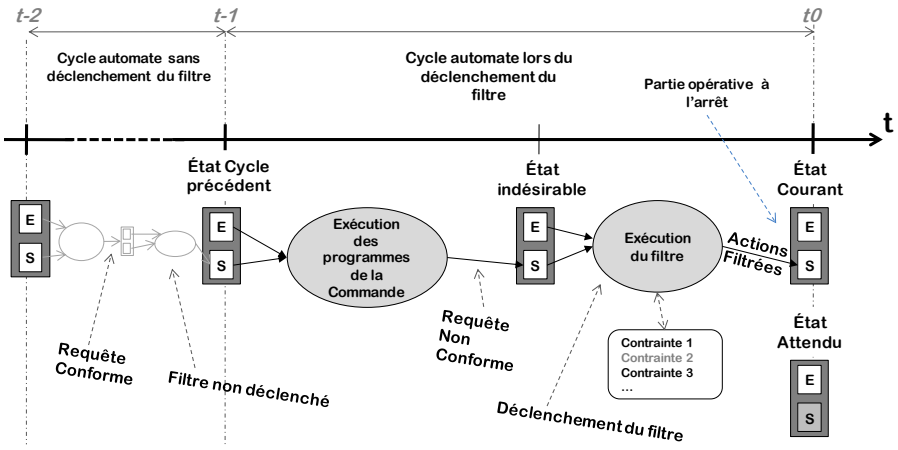

Figure 5 : Informations accessibles dans l'API pour assurer l'aide

Les éléments accessibles pour assurer l'aide sont présentés sur la Figure 5 selon les cycles d'exécution de l'automate. On dispose de l'état des entrées/sorties en début de cycle, des sorties calculées par la commande, de la ou des contraintes de sécurité qui ont été déclenchées. La commande en elle-même ainsi que ses éléments internes tels que temporisations et mémoires sont considérés comme non accessibles, leur analyse étant trop complexe. L'aide au diagnostic de la commande qui peut être faite n'est donc que partielle.

Suite au déclenchement du filtre, la PO n'évolue plus. Le niveau 0, appelé « Etat Courant » et qui n'est pas considéré comme une aide, sert uniquement à faire comprendre à l'opérateur que le filtre de sécurité a déclenché (Figure 6). L'interface présente donc l'état réel de la PO après un arrêt provoqué par le déclenchement d'une des contraintes du filtre de sécurité. La plupart des actionneurs seront à l'arrêt mais dans certaines parties opératives, ils pourront être maintenus actifs pour des raisons de sécurité.

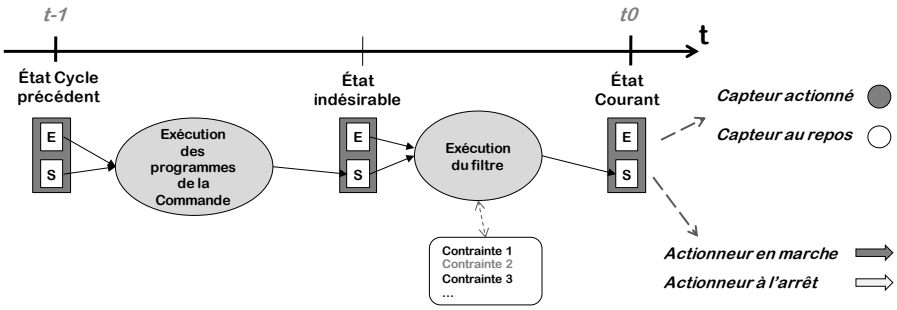

Figure 6 : Eléments pris en compte pour le niveau « Etat Courant »

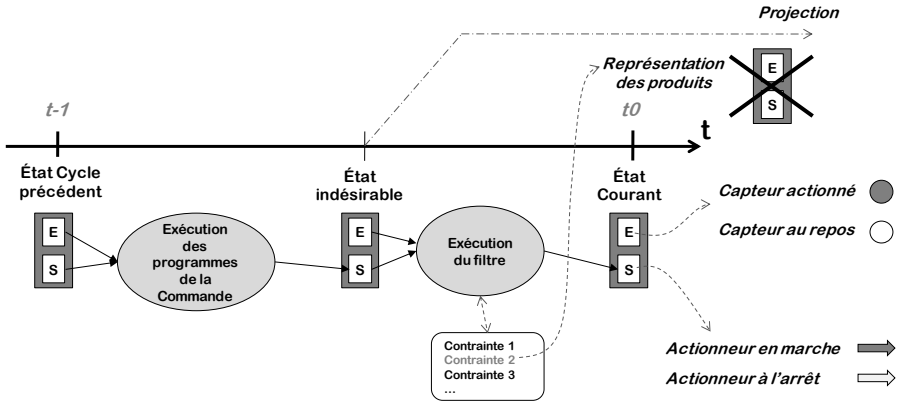

Figure 7 : Eléments pris en compte pour le Niveau « Projection »

Habituellement, le point de départ du diagnostic d'un opérateur d'une machine en panne se réalise à partir de différents indices : éléments de la PO en mauvaise position voire abimés, produits en mauvaise position voire détruits. L'interface présentera pour ce niveau qualifié de « projection », l'état de la PO dans lequel elle aurait été s'il n'y avait pas eu de filtre, c'est-à-dire la représentation des conséquences de la commande erronée. Les informations affichées représenteront donc une projection de l'état futur de la PO à partir de l'état indésirable détecté par une contrainte de sécurité. Elles seront déterminées à partir uniquement de la contrainte déclenchée qui représente une situation interdite, Figure 7. Bien entendu, cette projection ne peut être que partielle mais doit permettre de focaliser l'attention de l'opérateur sur les éléments de la PO en défaut et met en exergue le non respect de ses contraintes de fonctionnement.

Les informations accessibles dans l'automate peuvent ensuite permettre de donner des éléments supplémentaires aux opérateurs pour leur diagnostic. La reconstruction d'indices sera de plus en plus fine par la suite. Dans le niveau d'aide suivant, appelé « état avant arrêt », seront présentés l'état des entrées et des sorties juste avant l'arrêt de la PO provoqué par le filtre, Figure 8. L'interface présentera donc l'état de la PO qui a conduit à l'activation d'une des contraintes du filtre, ce qui permet de focaliser l'attention de l'opérateur sur la situation qui a déclenché le filtre. Pour parfaire ceci, les éléments constitutifs de la contrainte qui a déclenché sont entourés, ainsi que l'action du filtre sur les actionneurs (« OFF », « ON »).

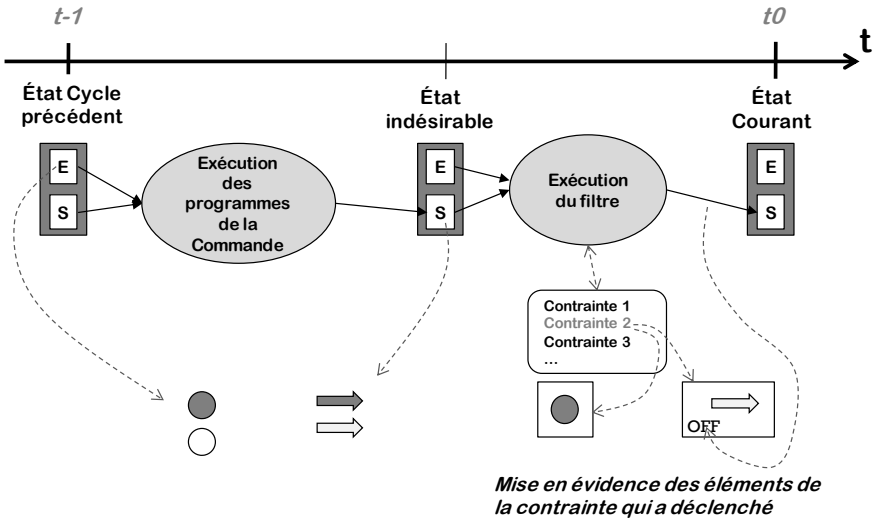

Figure 8 : Eléments pris en compte pour le Niveau « Etat avant Arrêt »

Le niveau d'aide suivant dit « état avancé », va chercher à soutenir la compréhension du mauvais comportement de la commande. L'interface présentera la situation initiale correcte prise en charge par la commande, à partir de laquelle la commande erronée conduit à une situation interdite. L'interface présentera l'état de la PO qui conduit au déclenchement du filtre, complété par les derniers changements d'états des capteurs et/ou actionneurs, ces derniers étant en fait le(s) événement(s) qui a (ont) provoqué une situation non-conforme détectée par le filtre, Figure 9.

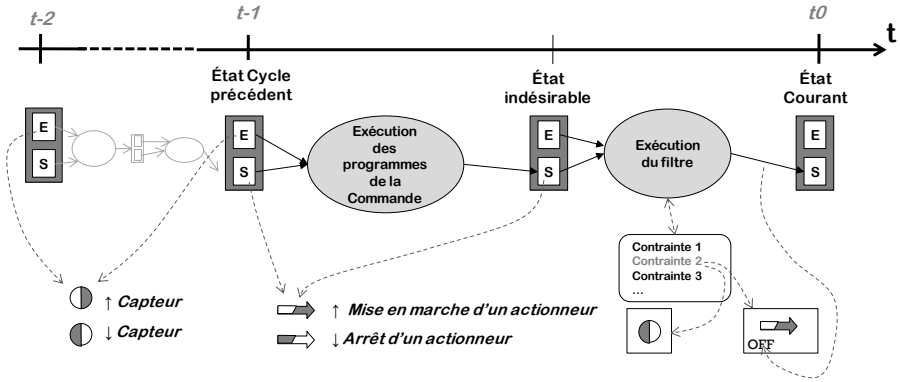

Figure 9 : Eléments pris en compte pour le Niveau « Etat avancé »

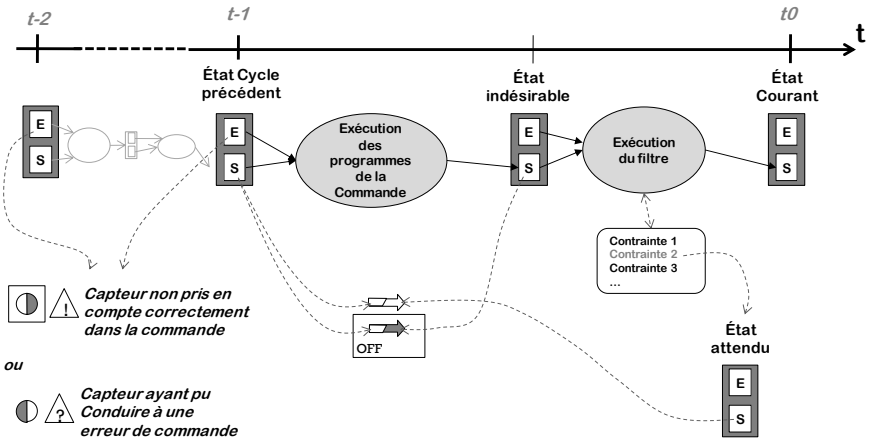

Figure 10 : Eléments pris en compte pour le Niveau « Hypothèse »

Pour corriger la commande erronée, il faut comprendre ou connaître le comportement qu'elle aurait dû avoir. Le dernier niveau appelé « Hypothèse », cherche à soutenir la compréhension du mauvais comportement de la commande en indiquant le comportement qu'elle aurait dû avoir en fonction du contexte. Deux types d'erreurs peuvent ici être déterminés (**Erreur ! Source du renvoi introuvable.)** :

- Pourquoi une information n'a pas été prise en compte pour modifier les sorties ? L'interface se focalise alors sur les capteurs impliqués dans la contrainte qui a déclenché et pour lesquels il y a un changement d'état, mais qui n'ont pas conduit à la bonne évolution d'une ou des sorties.
- Pourquoi une sortie a été modifiée alors qu'elle ne devait pas l'être ? L'interface se focalise alors sur les capteurs non impliqués dans la contrainte pour lesquels il y a un changement d'état ayant pu conduire à un changement d'état d'un actionneur impliqué dans la contrainte.

L'interface présente aussi les actions attendues sur les actionneurs impliqués dans la contrainte, i.e. le comportement que la commande aurait dû avoir.

Le problème majeur de ces différents niveaux et que plus le niveau d'aide augmente, plus l'analyse de tous les cas de figure se complexifie et donc plus le coût de développement des interfaces va augmenter. Pour les premiers niveaux, l'extraction des informations à afficher est quasi automatique alors que pour le dernier, le niveau où des hypothèses sont proposées, il y a explosion combinatoire des cas à étudier. Une série d'expérimentations a donc été menée, l'objectif étant de valider le niveau minimal d'explication permettant à un opérateur de maintenance de comprendre la situation, diagnostiquer l'erreur de commande pour finalement modifier la commande erronée. Ces expérimentations ont été menées sur le cas d'étude qui est présenté maintenant.

6.Cas d'étude : tri de caisses

6.1. Présentation de l'étude de cas

L'ensemble de la méthodologie proposée est maintenant illustrée au moyen d'un système virtuel de la collection ITS PLC proposée par la société portugaise Real Games (Magalhaes *et al*., 2012). Des versions de démonstration ainsi que des descriptions techniques des cinq systèmes industriels virtuels proposés sont téléchargeables gratuitement à l'adresse www.realgames.pt. La collection ITS PLC est un ensemble de logiciels de simulation de PO pour la formation à l'automation (Riera *et al*., 2011). Dans le cadre du travail présenté dans cet article, le système « tri de caisses » a été utilisé, Figure 11.

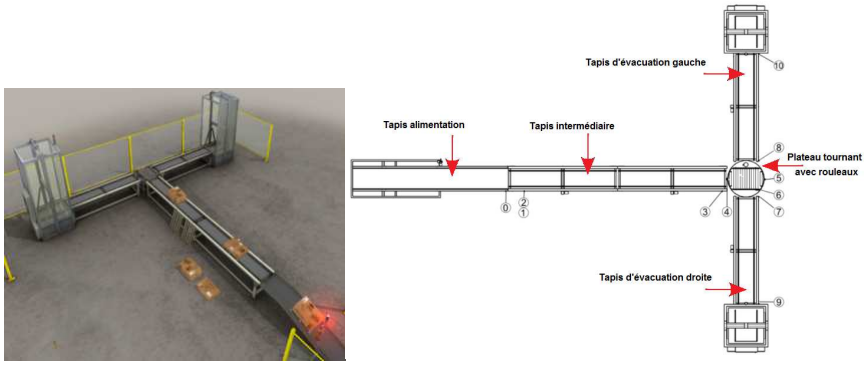

Figure 11 : Système de tri de caisses issu de la collection ITS PLC

L'objectif du système est d'amener des caisses du tapis d'alimentation aux monte-charges en les triant selon leur arrivée ou leur taille par exemple. Le système est instrumenté au moyen de 11 capteurs (tableau 2) permettant de déterminer la taille des caisses (petite ou grande), l'entrée ou la sortie d'une caisse des différents convoyeurs (alimentation, intermédiaire, évacuation) et, enfin, la position du plateau tournant. Les 7 sorties de l'API (tableau 3) permettent de mettre en marche les différents convoyeurs et le plateau tournant.

Tableau 2. Détail des entrées du système de tri de caisses

Capteur	Information	Eléments
C0	Caisse en fin de Convoyeur	Convoyeur d'alimentation
C1	Petite caisse	Convoyeur intermédiaire
C2	Grande caisse	
C3	Caisse en fin de Convoyeur	
C4	Position de chargement	Plateau tournant
C5	Position de déchargement	
C6	Présence de caisse	
C7	Caisse en entrée du convoyeur	Convoyeur d'évacuation à

C9	Caisse en fin de convoyeur	droite
C8	Caisse en entrée du convoyeur	Convoyeur d'évacuation à gauche
C10	Caisse en fin de convoyeur	

Tableau 3. Détail des sorties du système de tri de caisses

Actionneur	Action
A0	Activation du convoyeur d'alimentation
A1	Activation du convoyeur intermédiaire
A2	Activation des rouleaux (chargement et évacuation à gauche)
A3	Activation des rouleaux (évacuation à droite)
A4	Rotation du plateau en position de déchargement (Actionneur monostable à retour automatique)
A5	Activation du convoyeur gauche
A6	Activation du convoyeur droit

Le cahier des charges retenu est le suivant : après un appui sur le bouton « start », les caisses sont acheminées à tour de rôle vers le monte-charge de gauche et le monte-charge de droite en fonction de leur taille. Après un appui sur le bouton « stop », les caisses en cours d'acheminement sont évacuées. Pour ne pas surcharger l'article, on considère que les convoyeurs de sortie sont toujours en marche (A5 = A6 = 1) et la gestion des boutons « start » et « stop » n'est pas présentée. De plus, la mise en place des diagnostiqueurs et la coopération filtre/diagnostic portera sur l'actionneur A4 permettant la rotation du plateau tournant. Celui-ci est un actionneur simple effet, c'est à dire que l'ordre doit être maintenu pour qu'il reste en position souhaité sinon, il revient en position initiale.

6.2. Obtention du filtre

La conception du filtre a permis d'aboutir aux 15 CSs (équations (9) à (23)) et 2 CSc (équations (24) et (25)) représentées sous la forme de monômes logiques. Ces équations ont été vérifiées formellement au moyen du model-checker UPPAAL. Celles-ci garantissent la commandabilité, i.e. il existe au moins une commande permettant d'amener des caisses vers l'ascenseur de gauche et l'ascenseur de droite, et d'assurer la sécurité quelle que soit la commande. Il est à noter que ces contraintes sont les plus permissives possibles (large espace de commande autorisé) mais nécessitent 3 observateurs (2P, P36 et P67).

$$\mathrm{CSs1} = \mathrm{2P}.\mathrm{A0} \quad (9)$$

$$\mathrm{CSs2} = \mathrm{C3}.\overline{\mathrm{C4}}.\mathrm{A1} \quad (10)$$

$$\mathrm{CSs3} = \mathrm{C3}.\mathrm{C4}.\mathrm{C6}.\mathrm{A1} \quad (11)$$

$$\mathrm{CSs4} = \mathrm{C3}.\mathrm{P36}.\mathrm{A1} \quad (12)$$

$$\mathrm{CSs5} = \overline{\mathrm{C5}}.\mathrm{A3} \quad (13)$$

$$\mathrm{CSs6} = \mathrm{C4}.\mathrm{C6}.\mathrm{A2} \quad (14)$$

$$\mathrm{CSs7} = \overline{\mathrm{C4}}.\overline{\mathrm{C5}}.\mathrm{A2} \quad (15)$$

$$\mathrm{CSs8} = \mathrm{C5}.\mathrm{C6}.\overline{\mathrm{A4}} \quad (16)$$

$$\mathrm{CSs9} = \mathrm{C8}.\overline{\mathrm{A4}} \quad (17)$$
$$\mathrm{CSs10} = \mathrm{C7}.\overline{\mathrm{A4}} \quad (18)$$
$$\mathrm{CSs11} = \mathrm{P67}.\overline{\mathrm{A4}} \quad (19)$$
$$\mathrm{CSs12} = \mathrm{C5}.\mathrm{C7}.\mathrm{A2} \quad (20)$$
$$\mathrm{CSs13} = \mathrm{C4}.\overline{\mathrm{C6}}\,.\mathrm{A4} \quad (21)$$
$$\mathrm{CSs14} = \overline{\mathrm{C4}}\,.\mathrm{RE}(\mathrm{A4}) \quad (22)$$
$$\mathrm{CSs15} = \overline{\mathrm{C5}}\,.\mathrm{FE}(\mathrm{A4}) \quad (23)$$
$$\mathrm{CSc1} = \mathrm{C0}.\mathrm{A0}.\overline{\mathrm{A1}} \quad (24)$$
$$\mathrm{CSc2} = \mathrm{C3}.\mathrm{C4}.\mathrm{A1}.\overline{\mathrm{A2}} \quad (25)$$

P36 et P67 sont des observateurs permettant de savoir qu'une caisse est présente respectivement entre les capteurs C3 et C6 exclus, et C6 et C7 exclus, les distances [C3, C6] et [C6, C7] étant supérieures à la longueur d'une caisse, ce qui n'est pas le cas par exemple pour [C6, C8]. P36 est mis à 1 sur un front descendant du capteur C3 et remis à 0 sur un front montant du capteur C6. Il en est de même pour P67 avec les capteurs concernés.

Concernant l'observateur 2P, la distance entre les capteurs C0 et C1 est plus petite que la taille d'une caisse. 2P permet en fait de déterminer que 2 caisses sont présentes entre C0 et C1et non pas une seule à cheval sur les deux capteurs, lorsque C0=C1=1.

6.3. Obtention du diagnostic et Coopération filtre/ diagnostic

L'obtention du diagnostiqueur et la coopération entre le diagnostic et le filtre sont traitées sur le plateau tournant utilisant l'actionneur A4 ainsi que les capteurs C4 et C5. Lorsque le signal de rotation *A4* est envoyé au plateau, cela provoque la désactivation du capteur de C4 dans l'intervalle de temps I2 puis l'activation du capteur C5 dans l'intervalle I4. Ces périodes de temps sont prédéfinies par les spécialistes en fonction du système dynamique et du comportement souhaité. Par apprentissage, il est possible alors d'obtenir 5 intervalles de temps (en ms) comme suit :

I1 =]0, t_{min1}[temps de réaction à l'activation de l'ordre (cycle API)

I2 = [t_{min1}, 130[attente du front descendant de C4

I3 = [130, 2900[rotation en cours

I4 = [2900, 2950[attente du front montant de C5

I5 = [2950, t_{max1}[attente de la désactivation de l'ordre

Les spécialistes établissent les intervalles de temps à partir de la désactivation de l'ordre *A4* pour le retour du plateau en position initiale.

I6 =]0, t_{min2}[temps de réaction à la désactivation de l'ordre (cycle API)

I7 = [t_{min2}, 150[attente du front descendant de C5

I8 = [150, 2880[rotation en cours

I9 = [2880, 2970[attente du front montant de C4

I10 = [2970, t_{max2}[attente de l'activation de l'ordre pour la rotation suivante.

Pour chaque intervalle de temps, une variable booléenne notée *Ii* est mise à 1 lorsque que le temps est compris dans l'intervalle et mise à 0 sinon.

Un diagnostiqueur local est obtenu après identification de toutes les défaillances possibles sur chaque composant de l'installation. Pour l'exemple du plateau, il est possible d'identifier chaque événement défectueux par une étiquette :

- Capteur C4 bloqué à 0 (F1) ou à 1 (F2)
- Capteur C5 bloqué à 0 (F3) ou à 1 (F4)
- Plateau bloqué en C4 (F5) ou en C5 (F6)
- Evolution inattendue de 0 à 1 de C4 (F7) ou C5 (F9)
- Evolution inattendue de 1 à 0 de C4 (F8) ou C5 (F10)
- Un mouvement inattendu ou plateau bloqué sur passage de C4 à C5 (F11) ou de C5 à C4 (F12)

Trois partitions de défaut sont définies et appartiennent à :

- Capteur C4: ΠC4 = {F1, F2, F7, F8}
- Capteur C5 : ΠC5 = {F3, F4, F9, F10}
- Plateau: ΠP = {F5, F6, F11, F12}

La Figure 12 donne le modèle de diagnostic du plateau. Le filtre utilisé pour la construction du diagnostic tient notamment compte des contraintes (22 et 23).

Pour chaque partition de défaut, un flag est positionné à vrai lorsque le diagnostiqueur atteint un état de défaillance avec certitude : *def_P* pour la partition du plateau, *def_C4* pour la partition du capteur C4, *def_C5* pour la partition du capteur C5. Ce flag détermine si la variable considérée peut encore être utilisée dans la contrainte de filtre ou non. Dans ce cas, les contraintes du filtre doivent être enrichies par une information complémentaire. Pour cela, les intervalles de temps issus de l'apprentissage sont utilisés. Les informations C4 et C5 peuvent être alors remplacées par leur estimée lorsque l'intervalle est actif :

- Pour C4 : $\widehat{C4} = I1 + I2 + I9 + I10$
- Pour C5 : $\widehat{C5} = I4 + I5 + I6 + I7$

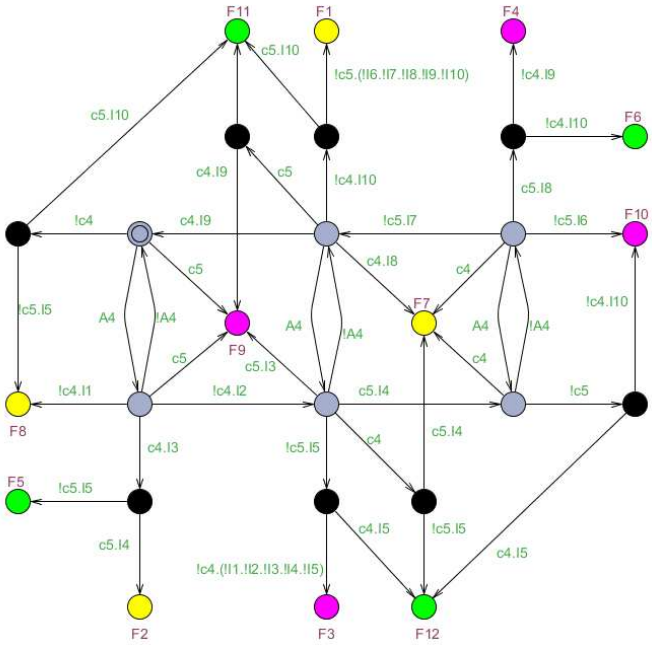

Figure 12 : Modèle de diagnostic du plateau

Avec ces informations supplémentaires issues du diagnostic, les contraintes de sécurité sont enrichies. Par exemple, les contraintes CSs5, CSs6 et CSs13 deviennent :

$$\mathrm{CSs5} = (\overline{\mathrm{def\_C5}}.\overline{\mathrm{C5}} + \mathrm{def\_C5}.\overline{\widehat{C5}}).\mathrm{A3} \quad (29)$$

$$\mathrm{CSs6} = (\overline{\mathrm{def\_C4}}.\mathrm{C4} + \mathrm{def\_C4}.\widehat{C4}).\mathrm{C6}.\mathrm{A2} \quad (30)$$

$$\mathrm{CSs13} = (\overline{\mathrm{def\_C4}}.\mathrm{C4} + \mathrm{def\_C4}.\widehat{C4}).\overline{\mathrm{C6}}.\overline{\mathrm{def\_P}}.\mathrm{A4} \quad (31)$$

On remarque qu'il n'y a pas d'équivalence lorsqu'un défaut apparaît sur l'actionneur. En effet, lorsque celui-ci est défectueux, il convient de stopper l'ensemble du système pour garantir la sécurité. A contrario, l'équivalence d'une information capteur permet dans certains cas de continuer la production en mode dégradé sans pour autant perdre l'aspect sécurité. Si cela n'est pas possible, il est envisageable de remonter des informations à l'opérateur pour que celui-ci prenne la décision de continuer ou non la production malgré la présence d'une défaillance.

Les interfaces permettant d'expliquer la situation suite au déclenchement du filtre sont à présent détaillées.

6.4. Génération d'explication

Dans le cadre du système de tri de caisses, les interfaces correspondant au module d'explication ont été développées puis implémentées sur contrôleur WAGO à l'aide du logiciel CodeSys, ces derniers permettant l'implémentation aisée de petites interfaces de visualisation. Ces interfaces sont connectées de manière transparente au filtre de sécurité et aux programmes utilisateurs. Tous les niveaux définis initialement ont été implémentés afin de valider leur représentation avec des utilisateurs. Différentes expérimentations ont été menées afin de déterminer le niveau minimal permettant d'aider efficacement un agent de maintenance en charge de corriger la commande suite à un déclenchement de filtre.

6.4.1. Présentation des interfaces

Les figures présentées ici ont été refaites pour une meilleure visibilité en petit format. La situation représentée pour toutes ces figures correspond au déclenchement de la contrainte CSs10, i.e. à un relâchement de la commande du plateau tournant A4 (qui a pour effet de faire revenir celui-ci à sa position initiale) alors que la caisse qu'il contient n'a pas encore été évacuée complètement (C7 actif).

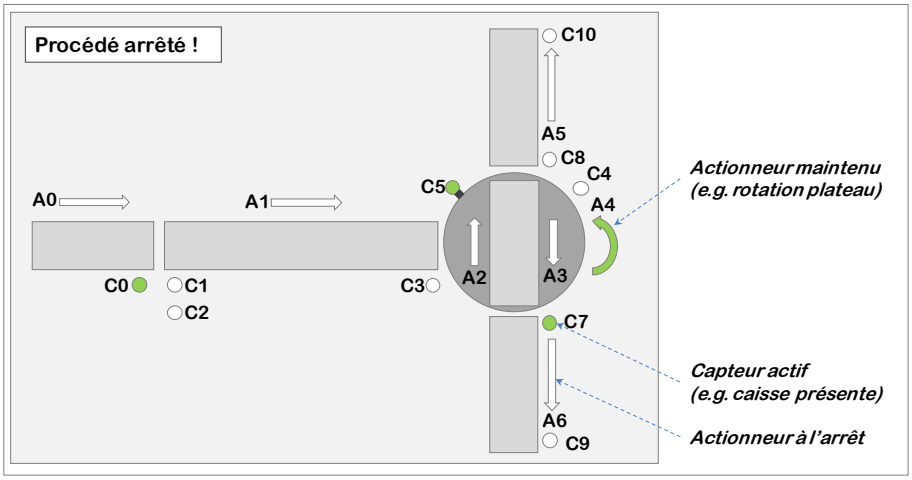

Figure 13 : Interface correspondant à la vue « Etat courant »

La PO s'y prêtant bien ici, toutes les interfaces ont été représentées par une vue de dessus. L'interface « Etat Courant » (Figure 13) s'affiche lors du déclenchement du filtre. Elle présente donc l'état réel des capteurs et actionneurs qui sont en général mis à l'arrêt (sauf ici où l'actionneur A4 est maintenu pour ne pas détruire la caisse que contient le plateau tournant). Lorsqu'un observateur du filtre est positionné, une caisse est affichée.

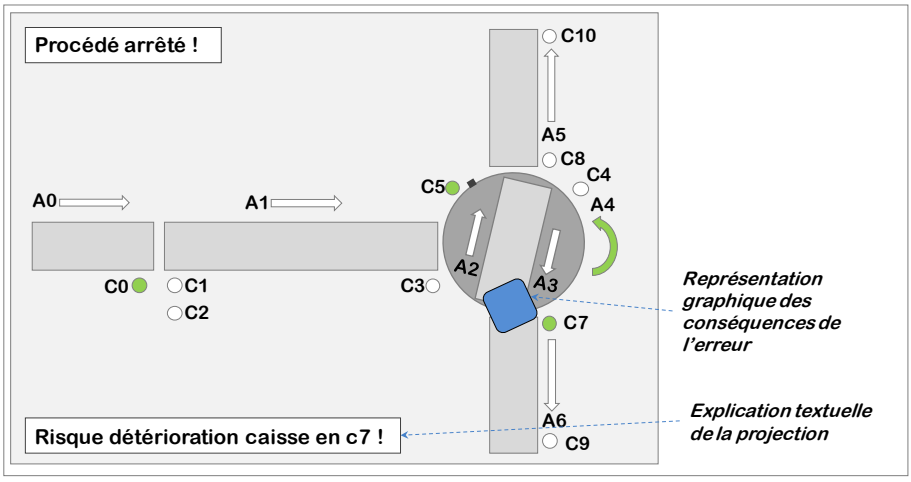

Figure 14 : Interface correspondant au niveau « Projection »

Le niveau « projection » correspondant au premier niveau d'aide, représente la PO dans l'état où elle aurait dû être si le filtre n'avait pas été intégré, (Figure 14). Elle donne donc des indices sur le résultat virtuel de la commande erronée puisque la PO a été figée avant.

Le niveau suivant, « état avant arrêt », représente l'état des capteurs et actionneurs juste avant le déclenchement du filtre, Figure 15. Il montre en fait l'état interdit qu'il ne fallait pas atteindre. Ici, le plateau tournant (A4) n'est plus maintenu (sur cette PO, le plateau va se repositionner dans le prolongement des tapis A0 et A1) et donc la caisse qu'il contient (C7 activé) risque d'être détruite. Sur cette vue, même si l'opérateur ne connaît pas les équations des contraintes, les différents éléments qui la composent sont entourés en rouge car c'est leur combinaison qui la déclenche.

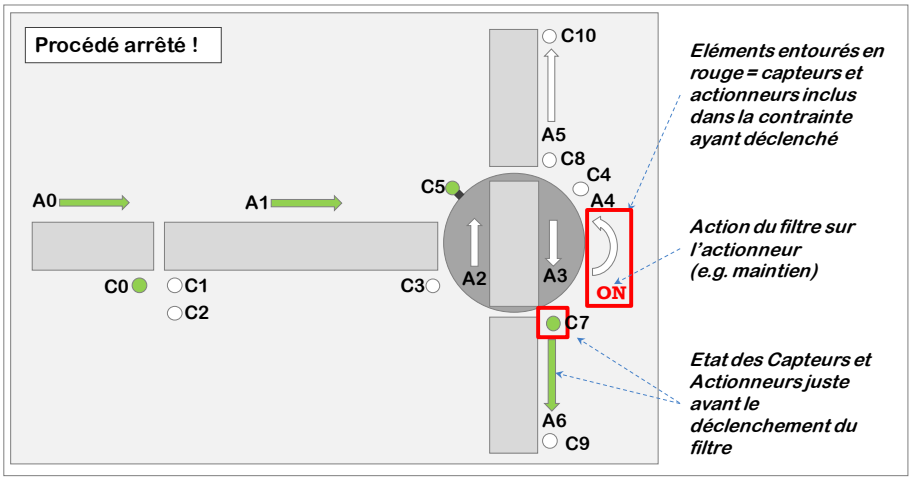

Figure 15 : Interface correspondant au niveau « Etat avant arrêt »

Sur le niveau suivant (Figure 16) est représenté l'état avant arrêt et l'état précédent. On présente donc des changements d'état. Dans l'exemple, on voit que le capteur C7 est passé de 0 à 1, montrant qu'une caisse arrive en début de tapis d'évacuation droit, l'actionneur A3 était activé et reste activé (d'où le fait que la caisse se présente en début de tapis), l'actionneur A6 passe de l'état d'arrêt à l'état d'activation, ce qui paraît logique puisque la caisse arrive dessus, et enfin, l'actionneur A4 qui était à présent activé et maintenait le plateau tournant en position de déchargement, est arrêté, d'où le déclenchement de la contrainte.

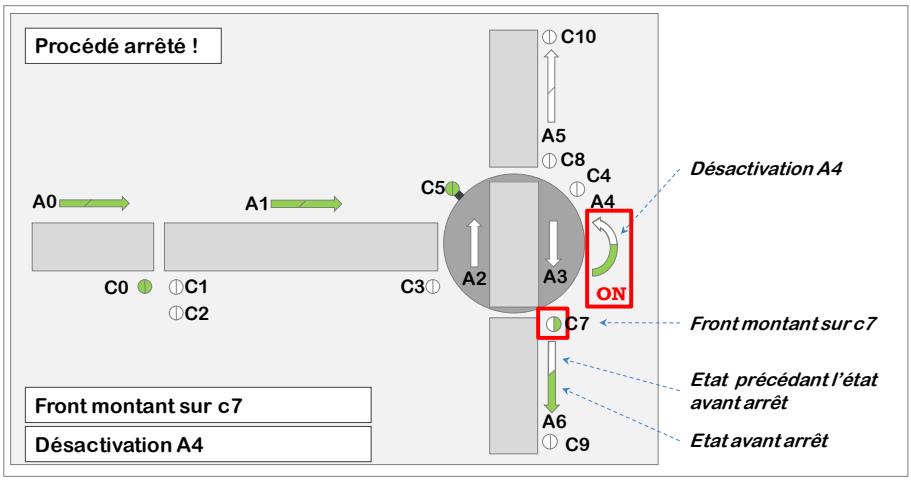

Figure 16 : Interface correspondant au niveau « Etat avancé »

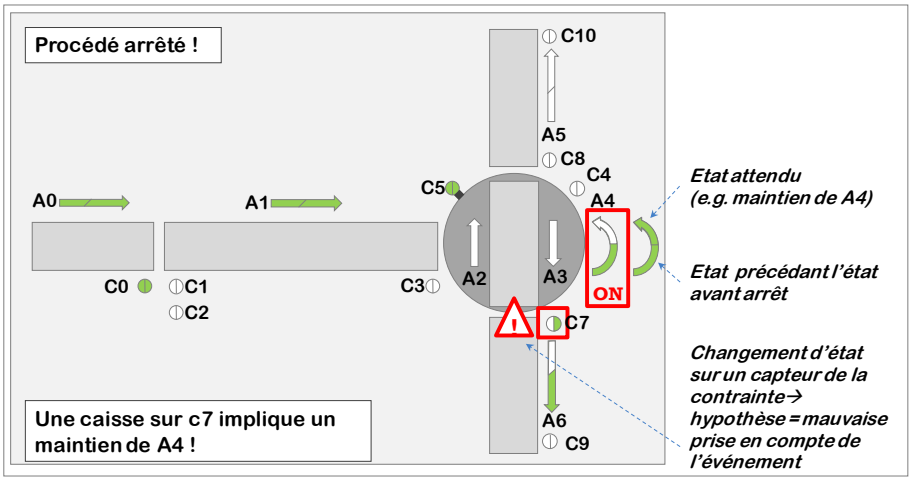

Figure 17 : Interface correspondant au niveau « Hypothèse »

Jusqu'à présent, l'ensemble des informations présentées sur les interfaces peut être collecté de manière automatique. Le dernier niveau, qualifié d'hypothèse (Figure 17) vise à montrer à l'opérateur quels sont les événements qui ont pu conduire la commande à produire une situation interdite et quel était l'état attendu des actionneurs. Dans l'exemple, A4 aurait dû être maintenu activé et il y a un changement d'état sur un des capteurs faisant partie de la contrainte. On peut légitimement penser que ce changement d'état a pu conduire à l'erreur. Si le front montant sur c7 a fait évoluer correctement A6, ce n'est pas le cas de A4 et ces éléments peuvent conduire l'opérateur à cibler l'erreur dans la partie commande.

Si un changement d'état se produit sur un capteur qui ne fait pas partie de la contrainte, l'interface listera tous ces capteurs de manière spécifique. L'erreur proviendra certainement d'une mauvaise réceptivité impliquant un de ces capteurs. Enfin, s'il n'y a aucun changement d'entrée, on peut supposer que l'erreur provient de traitements internes à l'automate telle qu'un déclenchement lié à une temporisation.

Un des problèmes lié à cette approche est la génération de ce dernier niveau d'aide. Il faut étudier pour chacune des contraintes tous les changements possibles des actionneurs et capteurs, en l'occurrence 122 cas possibles. On se rend bien compte que pour une PO plus complexe, le coût de l'étude de l'interface risque d'être important, d'où la nécessité de vérifier expérimentalement qu'un niveau inférieur ne serait pas suffisant pour aider l'opérateur.

6.4.2. Evaluations expérimentales

Les expérimentations réalisées n'avaient pour objectif que d'évaluer les interfaces conçues pour les quatre niveaux d'assistance. Aucune comparaison avec

un système conventionnel sans filtre n'a été réalisée, le gain d'un système avec filtre étant lié au fait qu'aucun endommagement de la PO et des produits n'est possible. L'objectif est de définir le niveau d'aide minimal permettant à un opérateur de diagnostiquer correctement une erreur de commande, sachant que plus le niveau d'aide est important, plus il risque d'être difficile à comprendre et plus il est coûteux en termes d'études et de développement.

6.4.2.1. Protocole expérimental

Le protocole expérimental a été conçu dans le but d'évaluer les niveaux d'explication. Chaque participant évalue les aides individuellement au cours de session de deux heures maximum. À leur arrivée, les participants sont informés sur les objectifs du projet et le protocole expérimental. Ils lisent et signent un document de consentement éclairé et remplissent un bref questionnaire afin de déterminer notamment leur savoir-faire et niveau de compétences en matière d'automatisme. Ensuite, ils ont une courte formation sur le procédé lui-même (le simulateur de tri de caisses) et apprennent à le contrôler en mode manuel. Les participants sont ensuite formés au système d'aide. Ils apprennent à utiliser les différentes interfaces et ont à leur disposition une documentation technique. Ils sont entrainés aussi à la correction d'une commande erronée.

Les participants ont effectué chacun trois passations. Pour chacune des passations, ils reçoivent une photo du processus arrêté comme s'ils étaient sur site, et la vue « état courant » (Figure 13). Dans la première passation qui correspond à la situation de référence, les participants ne disposent d'aucune autre information. La deuxième passation porte sur le premier niveau d'explication fournie avec la « vue projection ». Dans la dernière passation, un des trois niveaux plus avancés d'assistance est donné. Les participants testent donc un seul niveau avancé de l'aide afin d'éviter les effets d'apprentissage. Pour la même raison, les programmes de la commande et les erreurs de commande sont différents pour chaque essai. Scénarios et niveaux d'aide sont croisés selon un carré greco-latin afin d'éviter tout effet d'ordre et d'apprentissage.

Trois types de mesures ont été effectués. Les participants ont rempli des questionnaires sur leur perception du système d'aide : ils devaient choisir une réponse entre 1 et 7 pour évaluer la lisibilité, l'intelligibilité et la pertinence des explications fournies par le système d'aide. Ensuite, la conscience de la situation des participants a été évaluée, d'abord sur l'état de la PO, puis sur les informations apportées par les aides quand elles étaient disponibles. Le but est de déterminer sur quelle partie du procédé leur attention se focalisait. Enfin, les performances en termes de localisation et de correction de l'erreur dans les programmes ont été mesurées. Un exemple de correction de GRAFCET de l'automate par un participant est proposé sur la Figure 18. Le participant a écrit en rouge la correction et l'explication de l'erreur de commande sur le GRAFCET. Le système d'aide lui a permis de comprendre le problème qui est ici que l'actionneur A4 doit être maintenu actif.

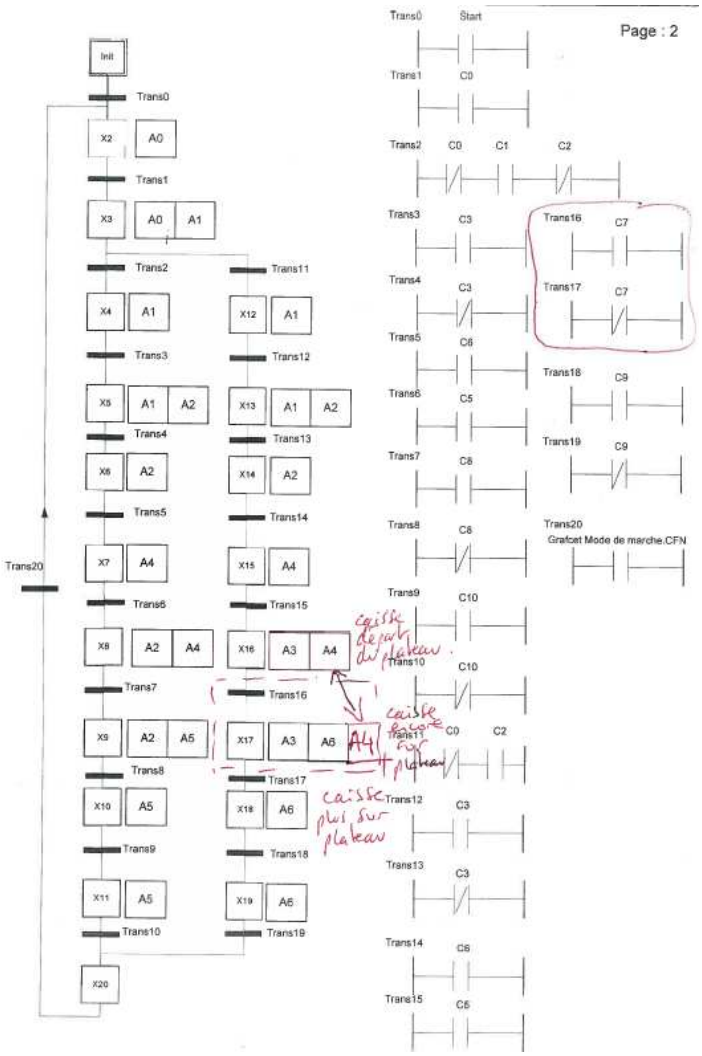

Figure 18 : exemple de corrections apportées par un opérateur

6.4.2.2.Résultats

Les expériences ont été menées avec 48 participants. L'échantillon était composé de 45 hommes et 3 femmes, avec une moyenne d'âge de 21,94 années, allant de 20 à 28 ans (± 2,02). Les résultats à partir des données subjectives sont issus des réponses aux questionnaires. En ce qui concerne l'auto-évaluation de leur niveau de compétences en automatisme, GRAFCET et Ladder, ces participants sont relativement confiants dans leur savoir-faire (réponse entre 1 et 7; Automatisme: 4,70 ± 1,19; GRAFCET: 5,12 ± 1,19; Ladder: 4,96 ± 1,50). Les participants ont

considéré qu'ils ont été suffisamment formés sur la PO utilisée pour cette expérimentation (5,88 ± 1,42), ainsi qu'avec le système d'assistance (5,59 ± 1,48)). Par contre, ils ont trouvé que la détection des erreurs de commande était difficile (réponse comprise entre 1 et 7; 3,50 ± 1,29). Néanmoins, ils ont considéré que leur niveau en automatisme était suffisant pour détecter et corriger les erreurs (5,37 ± 1,33).

Données subjectives :

Les résultats issus des données subjectives concernent l'évaluation des systèmes d'assistance par les participants. Ils ont évalué deux types de systèmes d'assistance, la vue " projection" (PV) (Figure 7) et une des vues avancée (AV) (Figure 8, Figure 9, **Erreur ! Source du renvoi introuvable.**). Ils ont estimé (échelle entre 1 et 7) que les informations proposées par les systèmes d'aide sont :

- lisibles : PV: 6,07 ± 1,23; AV: 6,11 ± 1,05,
- compréhensibles : PV: 5,81 ± 1,17; AV: 5,70 ± 1,24,
- proposent des explications pertinentes: PV: 5,61±1,19; AV: 5,44±1,19,
- donnent une analyse compréhensible: PV: 5,59 ± 1,15; AV: 5,48 ± 1,37,
- permettent de localiser les erreurs de commande : PV: 5,42±1,58; AV: 5,72±1,35,
- permettent de corriger les erreurs de commande : PV: 5,29±1,75; AV: 5,31±1,58.

Données objectives :

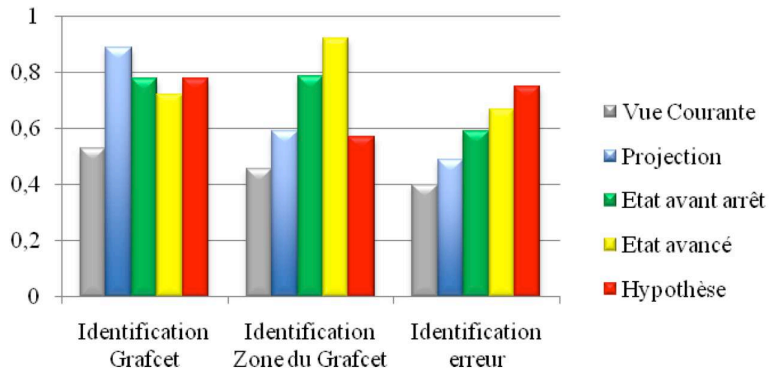

Figure 19 : Pourcentage de détection d'erreur (identification du GRAFCET concerné par l'erreur, identification de la zone du GRAFCET contenant l'erreur, identification de l'erreur).

Les résultats issus des données objectives concernent les corrections des programmes. Les différents programmes utilisés comportaient plusieurs GRAFCETs

et les réceptivités et le post-traitement pour les actionneurs sont exprimés en Ladder. Les erreurs introduites concernaient uniquement ces réceptivités et les actionneurs. Les résultats issus des données objectives concernent les corrections des programmes (GRAFCET et Ladder). Ces résultats montrent que les performances s'améliorent avec le niveau des aides proposées (Figure 19).

La Figure 19 indique le pourcentage de réussite pour (i) identifier le GRAFCET concerné par l'erreur, (ii) identifier dans ce GRAFCET la zone concernée par l'erreur et (iii) l'erreur en elle-même. Cette figure met en évidence que la vue « projection » a plutôt tendance à soutenir l'identification du GRAFCET concerné par l'erreur. La vue « état avancé » aide le participant à reconnaître quelle est la partie du GRAFCET qui engendre le déclenchement du filtre. Enfin, la vue « hypothèse » est l'aide qui amène le plus de participants à finaliser la correction, même si les deux autres aides avancées fournissent également de bons résultats.

Pour résumer, la tendance générale est que plus le système d'aide fournit des explications sur son analyse du procédé — les participants disposent alors d'informations plus importantes que celles données par la vue état courant et la vue projection — plus ils détectent et corrigent correctement les erreurs dans la commande. Le système d'aide fournit donc les indices nécessaires au diagnostic de l'erreur de commande et à sa correction.

7. Conclusion/ Perspectives

Ce papier a présenté les résultats obtenus dans le projet ADEXEC. Dans ce projet, un filtre robuste est utilisé afin d'éviter des détériorations sur les produits ou sur la PO par exemple dans le cas d'un mauvais code mis en œuvre dans un API, ou une mauvaise commande envoyée par l'opérateur humain.

La première contribution autour du filtre et du diagnostic de la PO a produit deux résultats. Le premier a proposé une formalisation de l'obtention des contraintes logique dans le filtre et le deuxième a montré que la collaboration entre le filtre et le diagnostic permet de simplifier la construction du diagnostiqueur et rend le filtre plus robuste aux défaillances de la PO.

La deuxième contribution se trouve au niveau de la génération d'explication lors d'une erreur de commande. Lorsque le filtre robuste détecte un état dangereux, le processus est arrêté et l'opérateur n'a pas d'indication pour effectuer un diagnostic et trouver l'erreur dans le programme de l'automate. Basées sur le concept de la coopération homme-machine, de nouvelles IHMs ont été proposées afin d'aider l'opérateur humain et différentes expériences ont été réalisées. Les premiers résultats de ces expériences ont été présentés et ont montré que plus le système d'aide fournit des explications sur l'analyse du processus, plus les agents détectent et corrigent aisément les erreurs dans la commande. Néanmoins, il est nécessaire de réaliser d'autres expériences afin de valider ces premiers résultats et les étendre aux commandes manuelles envoyées par les opérateurs vers la PO.

Ces premiers travaux entre les trois laboratoires le CRAN, le CReSTIC et le

LAMIH ont montré qu'il était intéressant de continuer cette collaboration pour formaliser la détection d'erreur aussi bien au niveau de la commande qu'au niveau des défaillances de la PO pour apporter une explication compréhensible par l'opérateur. Il reste néanmoins un certain nombre de recherches à effectuer pour valider globalement l'approche proposée.

- Les explications proposées ne supportent pour le moment que la phase de diagnostic et de correction d'erreurs de programme. Il reste à proposer un système d'explications pour les opérateurs de conduite lors de l'application d'actions incohérentes directement sur la PO, par exemple lors d'une dégradation de la conscience du mode de fonctionnement de cette PO (production normale, marche de clôture, etc.), ou tout simplement par un manque d'expertise.
- L'approche utilisée pour la vérification du filtre peut être utilisée pour évaluer la diagnosticabilité des diagnostiqueurs.
- La coopération entre un opérateur de conduite et le diagnostiqueur lorsque ce dernier n'est pas capable de lever une ambiguïté doit être aussi investigué. L'opérateur peut même être amené à aller plus loin en invalidant certaines contraintes de sécurité s'il y a lieu.

8.Références

Alanche P, Lhoste P., Morel G., Roesh M., Salim M., Salvi P., (1986). Application de la modélisation de la Partie opérative à la structuration de la commande, *Journée AFCET*, Montpellier.

Alonso-Gonzalez C., Moya N., Biswas G., (2010). Factoring dynamic bayes networks using possible conflicts. *In: Proceeding of the 21th International Workshop on Principles of Diagnosis*, DX10.

Behrmann, G., David, A. and Larsen, K. G. (2004). *A tutorial on Uppaal. In Formal Methods for the Design of Real-Time Systems.* International School on Formal Methods for the Design of Computer, Communication and Software Systems, Vol. 3185 of Lecture Notes in Computer Science, pp.200–236. Springer-Verlag.

Berard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen P., (1999). *Systems and software verification: Model-checking techniques and tools*, Heidelberg, Springer-Verlag Edition.

Caverni J.P., Nguyen-Xuan A., Hoc J.M., Politzer G., (1990). Raisonnements formels et raisonnement en situation. *In C. Bonnet, R. Ghiglione, & J-F. Richard, Traité de psychologie cognitive 2, le traitement de l'information symbolique*. Paris : Dunod, pp. 103-166.

Chaillet-Subias A., (1995). *Approche multi modèles pour la commande et la surveillance en temps réel des systèmes à événements discrets*, Thèse de doctorat de l'Université Paul Sabatier, Toulouse.

Clarke E. M., Grumberg O., Peled D. A., (1999). *Model Checking*. The MIT Press, Cambridge, Massachusetts.

Combacau M., (1991). *Commande et surveillance des systèmes à événements discrets complexes : application aux ateliers flexibles*, Thèse de doctorat de l'Université Paul Sabatier, Toulouse.

Debouk R, Lafortune S and Teneketzis D., (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, Vol.10, pp.33-86.

El-Khattabi S., (1993). *Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets : application aux systèmes de production flexibles*, Thèse de doctorat de l'Université de Lille.

Falzon P., (1987). *Les dialogues de diagnostic : l'évaluation des connaissances de l'interlocuteur*. Rapport INRIA, n°747. Rocquecourt : Institut National de Recherche en Informatique et Automatique.

Faure J-M., Lesage J-J., (2001). Methods for safe control systems design and implementations, *10th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'2001*, Vienna, Austria.

Genc S., Lafortune S., (2003). Distributed diagnosis of discrete-event systems using Petri nets. *Lecture Notes in Computer Science*, 2679, pp.316-336.

Holloway L.E., Krogh B.H., (1990). Fault detection and diagnosis in manufacturing systems: a behavioural model approach, *IEEE International Conference on computer Integrated Manufacturing*, pp 252-259.

Lhoste P., (1994). *Contribution au génie automatique : concepts, modèles, méthodes et outils*, Habilitation à diriger des recherches de l'Université de Nancy.

Magalhaes A., Riera B., Vigario B., (2012) *When Control Education Is the Name of the Game* Computer Games as Educational and Management Tools: Uses and Approaches, IGI Global, pp 185-205, 2012. 10.4018/978

Marangé P., Benlorhfar R., Gellot F., Riera B. (2010) Prevention of human control errors by robust filter for manufacturing system, 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems

Marangé P., Gellot F., Riera B. (2009) Industrial risk prevention by robust filter for manufacturing control system, 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'09)

Marangé, P., Tajer, A., Gellot, F., Carré-Ménétrier, V. (2008). Etude du comportement global d'un SED en vue de la validation de sa commande spécifiée par GRAFCET. *Journal Européen des Systèmes Automatisés*, Vol. *42*(1), pp 63-94.

Marangé P., Gellot F., Riera B., (2007). Remote control of automation systems for D.E.S. course, *IEEE Transaction on Industrial Electronics Special Section*, pp. 3103-3111.

Milne R., Nicol C., Ghallab M., Trave-massuyes L., Bousson, Quevedo J., Dousson C., Aguilar J., Guasch Tiger A., (1994). Real-time situation assessment of dynamic systems. *Intelligent Systems Engineering*. Volume 3, Issue 3, Autumn 1994, p. 103 – 124.

Pandalai D., Holloway L.E., (2000). Template languages for fault monitoring of timed discrete event processes. *IEEE transactions on automatic control*, Vol. 45(5), pp.868-882.

Pencolé Y., Cordier M.O., Rozé L., (2001). Incremental decentralized diagnosis approach for the supervision of a telecommunication network. *Proceedings of the International*

Workshop on Principles of Diagnosis (DX-01), San Sicario, Italy, pp. 151–158

Philippot A., Sayed Mouchaweh M., Carré-Ménétrier V., (2010). Chapter 16: Component models based approach for failure diagnosis of Discrete Event Systems. *Intelligent Industrial Systems: Modelling, Automation and Adaptive Behaviour*, IGI.

Ramadge G., Wonham W. M., (1989). The control of discrete event systems, *Proc. IEEE, Special issue on DEDSs*, 77, pp.81-98.

Rasmussen J., (1986). *Information processing and human-machine interaction*. Amsterdam: Elsevier Science.

Reiter R., (1987). A theory of diagnosis from first principles". *Artificial Intelligence*, Vol.32(1), pp.57-95.

Ribert-Van De Weerdt C., Brangier E., (2000). L'usage et l'efficacité des aides à la maintenance en télédiffusion. *Le travail Humain*, ISSN 0041-1868, Vol. 63, N° 4, pp 331-352, Presse Universitaire de France.

Roussel J.M., Denis B., (2002). Safety properties verification of ladder diagram programs, *Journal Européen des Systèmes Automatisés*, Hermès Editions, 36(7), pp905–917, ISSN 1269-6935.

Riera B., Benlorhfar R., Annebicque D., Gellot F., Vigario B., (2011). Robust control filter for manufacturing systems : application to PLC training. *18th World Congress of the International Federation of Automatic Control*, Milano, Italy.

Riera B., Coupat R., Philippot A., Gellot F., Annebicque D., (2014). Control design pattern based on safety logical constraints for Manufacturing Systems: Application to a Palletizer *12th IFAC - IEEE International Workshop On Discrete Event Systems (WODES'14)*, Paris, France, 2014.

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

Sampath M., (1995). *A discrete Event Systems Approach to Failure Diagnosis*. Thesis, University of Michigan.

Su R., Wonham W.M., Kurien J., Koutsoukos X. (2002). Distributed diagnosis for qualitative systems. *In Proceedings of International Workshop on Discrete Event Systems*

Tzafestas, S. G., & Watanabe, K. (1990). Modern approaches to system/sensor fault detection and diagnosis. Journal of Aircraft, Vol. 31(4), pp. 42–57