Approche distribuée pour la reconfiguration de la commande des systèmes manufacturiers

Imane TAHIRI[1,2], Alexandre PHILIPPOT[1], Véronique CARRE-MENETRIER[1] et Abdelouahed TAJER[2]

[1] CReSTIC, Université de Reims Champagne-Ardenne, Reims, France.

[2] LGECoS, Université Cadi Ayyad, Marrakech, Maroc.

Imane.tahiri@univ-reims.fr , alexandre.philippot@univ-reims.fr, veronique.carre@univ-reims.fr, a.tajer@uca.ma

Résumé

Dans cet article, nous proposons une approche pour la reconfiguration de la commande des systèmes à événements discrets. La contribution est basée sur une synthèse de contrôle sûr de fonctionnement en exploitant une architecture distribuée incluant des événements temporisés. Lorsqu'un défaut est détecté sur un capteur, le contrôleur du comportement normal est reconfiguré en un contrôleur de comportement fautif où des informations temporisées compensent les informations perdues par le capteur défectueux. L'approche est appliquée sur un système simple de transfert de caisses.

1 Introduction

Les systèmes manufacturiers soumis à de fortes contraintes induites par un environnement incertain, changeant et dominés par une solide concurrence internationale (Lee 2006) nécessite d'être flexibles (Terkaj, Tolio, et Valente 2009), (Bordoloi, Cooper, et Matsuo 1999) et robustes. L'objectif est de faire face à la diversité, à l'augmentation de la productivité (Bévan 2013) (Rawat, Gupta, et Juneja 2018) et de la qualité, à l'optimisation des coûts d'exploitation, et à la réduction des risques de défaillances.

Le concept des systèmes manufacturiers reconfigurables (SMR) est apparu en 1999 (Y. Koren et al. 1999). Il est considéré comme étant une solution pour gagner en compétitivité et répondre aux demandes d'un marché en perpétuelle mutation. En effet, concevoir un système qui peut être reconfiguré (Yoram Koren et Shpitalni 2010) d'une manière précise, rapide et peu coûteuse en fonction de l'évolution du marché offre un avantage économique important aux entreprises manufacturières. L'objectif majeur du SMR est de concevoir des systèmes avec des machines et des contrôleurs capables de respecter les coûts minimaux et de répondre aux nouvelles spécifications du marché. Le processus de reconfiguration est un processus de réorganisation du matériel et/ou du logiciel du système dans le but d'assurer la production en réalisant un compromis entre les objectifs de production et l'état du système. Ce processus de reconfiguration peut être déclenché par deux catégories d'événements liés aux produits ou aux ressources de la production. Un changement de production peut être lié à la nature de la production, à la qualité ou à la quantité des produits. En général, ces changements peuvent entraîner l'ajout et/ou la suppression de certaines ressources matérielles liées à son ensemble engagé dans la production en cours. D'autre part, un changement d'état d'une ressource de production est caractérisé par deux événements majeurs : les pannes et les réparations. En cas de défaillance, le processus de reconfiguration doit d'abord chercher à remplacer la ressource défaillante par une autre. L'objectif dans ce contexte est d'utiliser des redondances matérielles pour remplacer l'élément défectueux.

L'implémentation d'un processus de reconfiguration dépend de deux paramètres : l'événement de déclenchement et les contraintes de temps qui s'exercent sur le système lorsque cet événement se

produit. Deux situations complémentaires peuvent être envisagées : le lancement d'une nouvelle production lorsque le système est dans une situation d'arrêt et l'occurrence d'une défaillance dans un système en cours de fonctionnement. Le travail proposé dans cet article est positionné dans ce deuxième cas. Les nombreuses solutions proposées dans l'industrie et la recherche reposent sur une redondance matérielle pour remédier à la défaillance d'un composant du système, solutions qui s'avèrent coûteuses en raison de la technologie et la complexité des composants des systèmes de fabrication.

Pour éviter cette redondance, nous proposons dans cet article, une approche de commande qui soit reconfigurable, basée sur une information temporisée d'une classe spéciale de systèmes : les Systèmes à Evénements Discrets (SED). L'idée est de concevoir une commande reconfigurable capable d'adapter et d'exploiter les services encore disponibles offerts par la partie opérative (PO) du système en cas de détection d'un défaut de type capteur. Le processus de reconfiguration consiste à faire évoluer l'état courant du système issu du contrôleur de comportement normal vers un état cible appartenant au contrôleur de comportement fautif afin de maintenir le fonctionnement du système en dépit des défauts (commande tolérante aux fautes) qui peuvent l'entraver. Les informations perdues sur un capteur défectueux sont remplacées par des informations fournies par des estimateurs temporisés qui décrivent le fonctionnement estimé du capteur en utilisant un apprentissage (Tahiri et al. 2019).

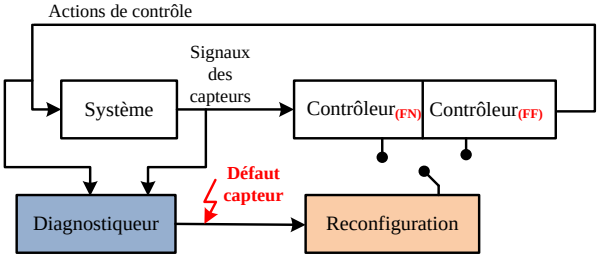


Figure 1 : La boucle de la reconfiguration de la commande

La boucle de reconfiguration (figure 1) de la commande repose sur trois éléments : (1) Le contrôleur obtenu à partir du superviseur élaboré dans le cadre de la théorie de contrôle par supervision (SCT) initiée dans (Ramadge and Wonham 1989). (2) Le diagnostiqueur qui consiste à détecter et à isoler les défauts. Le diagnostic n'est pas le but de cet article, quelques travaux de recherche connexes sont présentés dans (A. Philippot et Carré-Ménétrier 2011), (Blanke et al. 2016), (Hélouët et al. 2014). Dans ce travail, nous traitons le cas des défauts de capteur non observables qui sont définis par un capteur bloqué à 0 ou à 1. (3) Et enfin, le bloc de reconfiguration qui consiste à prendre la décision de passer du contrôleur de comportement normal au contrôleur de comportement fautif.

L'exemple qui illustre les travaux présentés dans cet article est un système de transfert de caisses (figure 2) construit à l'aide du simulateur 3D FACTORY I/O (https://factoryio.com/) qui offre à l'utilisateur, la possibilité de créer son système tout en permettant la génération de différents défauts au niveau des actionneurs et/ou des capteurs. Le système à commander est constitué de deux poussoirs A et B représentés par deux vérins monostables simple effet avec leurs capteurs de fin de course associés ({*a0, a1*} pour A et {*b0, b1*} pour B) comme le montre la figure 3. Deux convoyeurs transportent les caisses et les évacuent vers un stock. Deux capteurs de position *c* (resp. *e*) permettent de détecter les boites présentes devant le poussoir A (resp. B). Le démarrage de l'installation s'effectue par appui sur un bouton poussoir *dcy*. Dans cet article, nous étudions le comportement des poussoirs A et B et ignorons le fonctionnement des deux convoyeurs. Le cahier des charges à respecter est le suivant : le système ne doit pas envoyer les ordres de sortie des vérins A et B en même temps, l'ordre de sortie d'un poussoir ne peut être réalisé que si le vérin est en position rentrée (a0/b0) et l'ordre de sortie du vérin B ne peut être exécuté qu'après la sortie du vérin A.

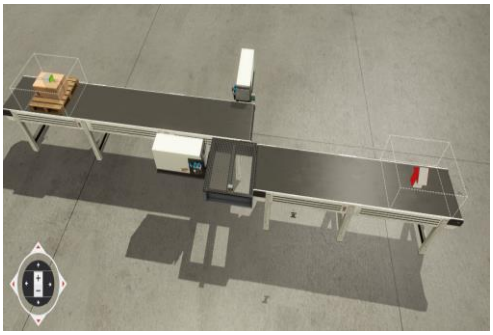


Figure 2 : Système de transfert des caisses

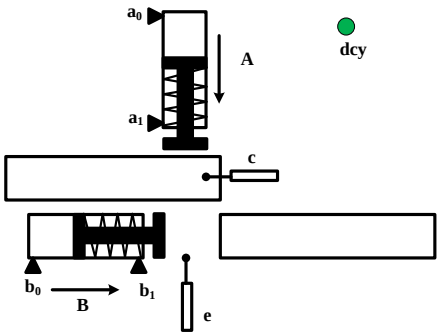


Figure 3 : Actionneurs et capteurs du système de transfert

La suite de cet article est organisée en trois sections. Nous présentons en section 2 les principes retenus pour élaborer un contrôleur de commande. La section 3 décrit l'approche de reconfiguration de la commande distribuée avec l'application sur l'exemple de transfert de caisses. La dernière section conclut et propose des perspectives au travail proposé dans cet article.

2 Principe d'élaboration du contrôleur de commande

2.1 Rappel sur la synthèse de contrôle par supervision

Selon la théorie de contrôle par supervision SCT (figure 4), le superviseur est calculé à partir de deux automates distincts : le premier représentant un système G donné et le second la spécification K décrivant le comportement contrôlé souhaité du système. Les algorithmes de synthèse génèrent automatiquement le superviseur le plus permissif qui doit fonctionner avec le système pour restreindre le comportement de ce dernier afin de respecter les spécifications. Le système est modélisé comme étant un générateur d'événements qui peuvent appartenir à l'ensemble des événements contrôlables ($z_i \in \Sigma_c$) ou incontrôlables ($e_i \in \Sigma_{nc}$). Le superviseur agit en autorisant ou inhibant les événements contrôlables qui peuvent être générés par le système dans un état donné. En revanche, les événements incontrôlables ne sont pas soumis à l'influence du superviseur. Les algorithmes de synthèse basés sur la SCT génèrent le superviseur correspondant au comportement le plus permissif contrôlable et non-bloquant du modèle du système qui satisfait les spécifications. Un comportement est dit contrôlable ou commandable (Kumar, Garg, et Marcus 1991) si l'occurrence de tout événement non contrôlable possible pouvant être généré par le système à un moment donné maintient le comportement du système dans K.

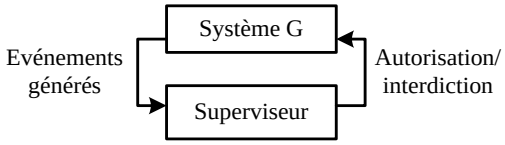


Figure 4 : Schéma de contrôle par supervision

2.2 Démarche de construction du contrôleur en mode normal

Elaborer le contrôleur (CONT) à partir du superviseur (SUP) pour obtenir une commande sûre de fonctionnement, c'est-à-dire qui tient compte de ce que le système doit faire et de ce que le système de doit pas faire, repose sur trois étapes principales (figure 5) :

1. Etape 1 : Modélisation du comportement de la partie opérative du système (PO), des spécifications du système sous forme de contraintes de sécurité (ce que le système ne doit pas faire) et de contraintes de vivacité (ce que le système doit faire)
2. Etape 2 : Synthèse, à partir des contraintes de sécurité et du modèle de PO, pour obtenir le superviseur (SUP) qui correspond au comportement maximal admissible du procédé
3. Etape 3 : Intersection du superviseur avec les contraintes de vivacité. Cette opération restreint le superviseur (SUP) et permet d'obtenir le contrôleur (CONT). Ce contrôleur pourra être traduit en Grafcet de commande si l'objectif est d'implanter de la commande dans un automate programmable industriel (API) à travers un langage normalisé (Norme IEC61131-3).

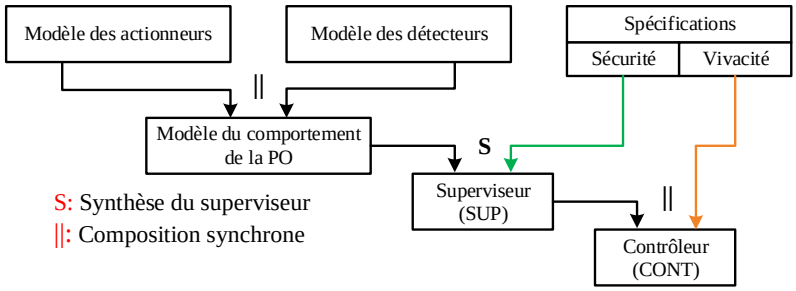


Figure 5 : Démarche d'élaboration du contrôleur en exploitant la SCT

2.2.1. Modélisation

2.2.1.1. Modélisation du comportement normal de la partie opérative (PO_N)

La modélisation du comportement normal de la PO est basée sur l'utilisation d'automate à états finis recevant des événements en provenance de la partie commande et agissant sur ses événements d'entrée du système de façon à exprimer les réactions de sa partie opérative. En outre, les modèles doivent tenir compte de la technologie du matériel utilisé, ainsi que de l'environnement du procédé. Ces modèles peuvent être obtenus à partir des relations entre les entrées et sorties du procédé à commander.

Nous utilisons l'interprétation de Balemi (Balemi et al. 1993) où l'ensemble des événements commandables Σ_c représente l'ensemble des sorties de la PC (les actionneurs), « $\uparrow z$ » pour l'activation des ordres de la commande et « $\downarrow z$ » pour leur désactivation ; l'ensemble des événements non commandables Σ_{nc} représente l'ensemble des entrées de la PC (les capteurs), « $\uparrow e$ » pour la lecture à « 1 » des états des capteurs et « $\downarrow e$ » pour la lecture à « 0 ». Nous avons alors $\Sigma_c = \uparrow z_i \cup \downarrow z_i$ et $\Sigma_{nc} = \uparrow e_i \cup \downarrow e_i$.

La détermination du modèle de comportement normal de la partie opérative du système noté (PO_N) est basée sur la modélisation proposée dans (Alexandre Philippot 2006). L'idée principale consiste à diviser la partie opérative en plusieurs éléments de PO appelé EPO et de définir ensuite le modèle des détecteurs et le modèle des actionneurs. Le modèle des détecteurs (détecteurs $_N$) décrit le comportement normal de tous les détecteurs qui constituent chaque EPO du système et le modèle des actionneurs (actionneurs $_N$) décrit le comportement normal de chaque actionneur avec ses détecteurs Le modèle de la partie opérative est alors obtenu par la synchronisation de ces deux modèles. Formellement, le modèle (PO_N) est défini par l'automate $A\_{N} = (Q\_{N}, \Sigma\_{N}, \delta\_{N}, q_{0}\_{N}, Q_{m}\_{N})$:

- Q_{N} : est l'ensemble fini des états de A_{N}
- Σ_{N} : est l'ensemble des événements de A_{N} tel que $\Sigma\_{N} = \Sigma\_{NT}$, avec Σ_{NT} est l'ensemble des événements non temporisés

- δ_N : est la fonction de transition. Une transition est définie par : $\delta_N (q_N, \sigma) = q'_N$, σ est l'occurrence d'un événement de Σ_N
- q_{0_N} : est l'état initial de l'automate A_N, tel que $q_{0\_N} \in Q_N$
- Q_{m_N} : est l'ensemble des états marqués dans A_N, tel que $Q_{m\_N} \subseteq Q_N$

2.2.1.2. Modélisation des spécifications

Les spécificités du système représentent le comportement des opérations à effectuer par le système, elles sont exprimées sous forme de contraintes de sécurité et de vivacité. L'intégration des contraintes de spécifications a pour objectif d'inhiber les actions et/ou à organiser et séquencer l'exécution des ordres envoyés au système. Une contrainte ne peut pas provoquer d'actions supplémentaires dans un modèle, mais peut exprimer une restriction ou une inhibition de ces actions. La modélisation peut être réalisée soit par des automates (Wonham, Cai, et Rudie 2018), soit par des équations logiques (Riera et al. 2015) . Les contraintes peuvent être appliquées globalement à l'ensemble du processus ou localement à chaque élément de partie opérative.

2.2.2. Synthèse du superviseur (SUP)

Étant donnés les modèles G et K d'un procédé et des spécifications à lui imposer, la synthèse de la commande par supervision est basée sur l'algorithme de Kumar qui calcule l'automate qui accepte le langage suprême commandable du comportement désiré pour le système en boucle fermée. Les étapes retenues pour cet algorithme sont les suivantes :

- Etape 1 : Construire le composé synchrone A de G et de K (A = G || K).
- Etape 2 : Déterminer l'ensemble des états défendus
- Etape 3 : Déterminer l'ensemble des états faiblement défendus
- Etape 4 : Supprimer du composé A l'ensemble des états défendus ainsi que l'ensemble des états faiblement défendus ainsi que les transitions associées à ces états

L'application de cet algorithme pour les spécifications de sécurité permet l'obtention de l'automate du superviseur (SUP).

2.2.3. Conception du contrôleur (CONT)

Dans cette étape, c'est le contrôleur de commande qui est recherché ce qui fait intervenir les spécifications de vivacité. L'algorithme présenté précédemment est de nouveau appliqué mais avec les spécifications de vivacité pour obtenir l'automate du contrôleur (CONT). Des modèles d'applications de cet algorithme pour des machines à états finis (MEF) et des équations logiques sont présenté dans (Kumar, Garg, and Marcus 1991), (Tajer, Philippot, and Carré-Ménétrier 2013) et (Qamsane, Tajer, and Philippot 2016).

2.3 Prise en compte de la défaillance des capteurs

Pour mettre en place la reconfiguration de la commande, nous avons besoin d'introduire la notion d'événements fautifs et de fait l'occurrence d'événements temporisés afin de compenser l'élément fautif et éviter l'utilisation des redondances matérielles. Prendre en compte ce type d'événements conduit à un changement de modélisation au niveau du comportement de la PO. En effet, l'ensemble des événements présenté précédemment constitué d'événements non-temporisés devient un ensemble d'événements constitués de deux types d'événements : les non-temporisés et les temporisés. Cela conduit notamment à un changement dans la détermination des fonctions de transitions.

2.3.1. Modèle du comportement de la PO (PO_F)

Le modèle de comportement de la partie opérative présenté précédemment ne prenant pas en compte les événements temporisés, nous proposons d'étendre la méthodologie en intégrant la notion d'événements temporisés pour déterminer le modèle de comportement fautif du système (PO_F).

Dans un travail précédent (Tahiri et al. 2019), nous avons proposé une méthode permettant d'inclure le temps pour les SED, on parle alors des systèmes à événements discrets temporisés. Le temps est ainsi représenté par une horloge et il est considéré comme un événement. Cette hypothèse est intéressante car elle facilite la tâche de modélisation par utilisation des machines à états finis. Le modèle du comportement fautif (PO_F) est alors obtenu par la synchronisation des deux modèles temporisés des détecteurs (détecteurs_F) et des actionneurs (actionneurs_F).

Formellement, le modèle (PO_F) est défini par l'automate $A\_F = (Q\_F, \Sigma\_F, \delta\_F, q_{0\_F}, Q_{m\_F})$ tel que :

- Q_F : est l'ensemble fini des états de A_F
- Σ_F : est l'ensemble des événements de A_F, tel que $\Sigma\_F = \Sigma_{NT} \cup \Sigma_T$, avec Σ_T est l'ensemble des événements temporisés tel que : $\Sigma_T = C \cup D$ avec :
    - C : est l'ensemble des horloges, chaque horloge est définie par ses activations et désactivations : $C = \uparrow cki \cup \downarrow cki$
    - D : est l'ensemble fini des durées « di » associées à chaque horloge cki, tel que $D=\{d_1, d_2, \ldots, d_i\}$
- δ_F : est la fonction de transition. Une transition est définie par : $\delta\_F (q\_F, \sigma)=q'\_F$. σ est l'occurrence d'un événement temporisé ou non de Σ_F
- q_{0_F} : est l'état initial de l'automate A_F, tel que $q_{0\_F} \in Q\_F$
- Q_{m_F} : est l'ensemble des états marqués dans A_F, tel que $Q_{m\_F} \subseteq Q\_F$

Un exemple d'un automate A_F contenant des événements temporisés et non temporisés est donné dans la figure 6.

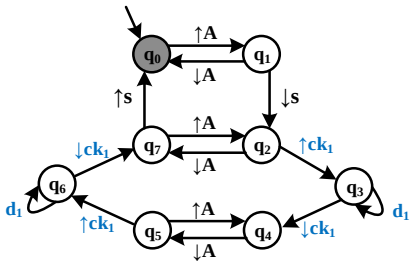


Figure 6 : Modèle d'un automate A_F

2.3.2. Modèle des spécifications

Chaque spécification de sécurité et/ou de vivacité est exprimée pour le comportement normal de la PO et il en est de même pour le comportement fautif. Les spécifications du comportement normal contenant des événements fautifs sont transformées en des spécifications temporisées. En effet, le comportement de la PO n'est plus spécifié par des événements associés aux capteurs fautifs, mais par des horloges (événements temporisés) qui spécifient le même comportement souhaité de la PO.

Par exemple, la spécification en comportement normal « **si** $\uparrow e_1$ **alors** act=1 » signifiant l'action « act » est forcée à 1 lorsqu'un front montant du capteur e1 est détecté, est transformée à la spécification « **si** d1 **alors** act=1 », avec d1 est l'événement temporisé qui traduit la durée nécessaire pour détecter un front montant de e1.

2.4 Architecture du contrôleur

Les approches classiques pour l'obtention d'un contrôleur sont basées sur des architectures centralisées. Cependant, ces approches s'exposent à un problème d'explosion combinatoire lié à la complexité de la phase de modélisation et des modèles par eux-mêmes. Pour pallier cet inconvénient, différentes approches sont proposées dans la littérature : hiérarchique, décentralisée, modulaire et distribuée. Le lecteur peut retrouver des détails dans (Wonham, Cai, et Rudie 2018) et (Zaytoon et Riera 2017).

L'approche centralisée en raison de l'explosion combinatoire est difficilement utilisable dans des applications complexes et le contrôleur résultant est très peu souvent implanté dans un API. Par conséquent, nous avons retenu pour cet article une approche distribuée basée sur les travaux de (Qamsane, Tajer, et Philippot 2016).

3 Contrôleur distribué et reconfiguration

Nous proposons dans cette section d'expliciter l'approche de reconfiguration de la commande en considérant une approche distribuée. Le schéma proposé pour la reconfiguration distribuée de la commande (figure 7) comporte sept étapes :

- ❶ Décomposition de la PO en plusieurs éléments de la PO (n×EPO) et modélisation pour chaque EPO du comportement normal et du comportement fautif
- ❷ Modélisation de deux ensembles de spécifications du système : les spécifications locales de sécurité et de vivacité propres à chaque EPO et les spécifications globales incluant les règles de reconfiguration
- ❸ Synthèse locale pour obtenir des contrôleurs locaux des deux comportements pour chaque EPO (❸b) issue des superviseurs locaux (❸a)
- ❹ Synthèse globale pour obtenir des contrôleurs distribués des deux comportements pour chaque EPO
- ❺ Interprétation des contrôleurs distribués en Grafcet (Norme IEC60848)
- ❻ Interprétation d'un ensemble des règles de reconfiguration en grafcet afin d'assurer la commutation entre les grafcets des deux comportements normal et fautif de chaque EPO
- ❼ Implémentation dans un API dans les langages de programmation (Norme IEC IEC61131-3), non illustré dans cet article

L'intervention du concepteur s'effectue essentiellement hors ligne, elle est en effet nécessaire dans les phases de spécifications et de modélisation. Quant à l'opérateur de surveillance, c'est en ligne qu'il visualise différents types d'informations : détection d'un défaut lui permettant de procéder au changement d'un capteur, situation de la commande (grafcet de fonctionnement normal ou grafcet de fonctionnement fautif), ...

3.1 Les modèles des EPO_N et EPO_F

La définition des deux modèles du fonctionnement normal ($EPO_{\_N}$) et du fonctionnement fautif ($EPO_{\_F}$) de chaque EPO du système est basée sur le même principe que la modélisation évoquée précédemment. Les modèles pratiques de chaque EPO ne sont pas synchronisés pour réaliser une reconfiguration distribuée de la commande. Soit G l'ensemble des modèles des EPO tel que : $G= G_{\_N} \cup G_{\_F}$, avec :

$G\_{N} = \bigcup_{i=1}^{n} A_{(i)N}$ ensemble des comportements normaux des EPO

Et

$G\_{F} = \bigcup_{i=1}^{n} A_{(i)F}$ ensemble des comportements fautifs des EPOoù n est le nombre des EPO constituant le système manufacturier

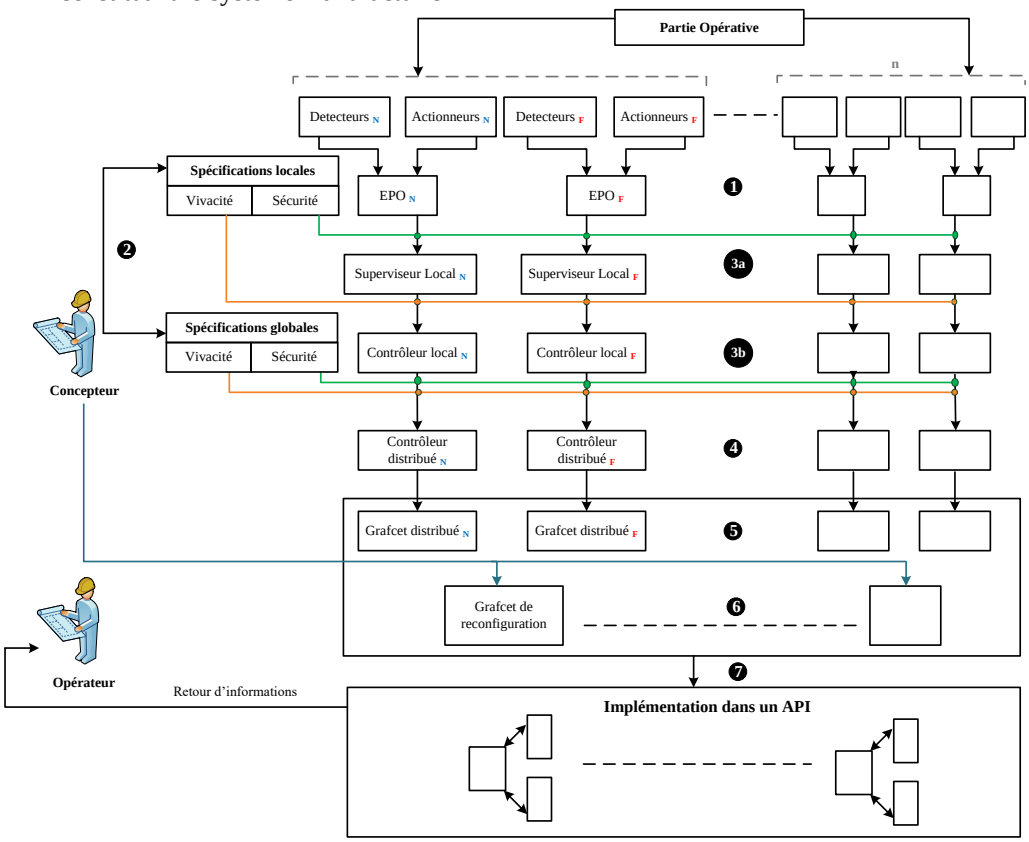


Figure 7 : Architecture de la reconfiguration de la commande distribuée

3.2 Les modèles de spécifications

Pour éviter l'explosion combinatoire, la modélisation des spécifications locales et globales s'effectue par des équations booléennes.

3.2.1. Spécifications locales

Les spécifications locales représentent un ensemble de contraintes de sécurité et de vivacité appliquées localement à un EPO. Elles sont définies par une implication logique donnée par la formule suivante :

$$\mathbf{x} \,.\, \mathbf{y} = \mathbf{0} \qquad \textbf{(eq1)}$$

« x » est un état de l'ensemble des états de G, et « y » est un événement contrôlable.

L'implication ci-dessus signifie que si « x » est vrai, alors « y » est interdit.

3.2.2. Spécifications globales

Les spécifications globales représentent un ensemble de contraintes de sécurité et de vivacité interagissant sur plusieurs EPO. Elles sont définies par une implication logique donnée par l'expression

suivante : **Si (c+d_i) alors {y = 0 sinon y = 1}** **(eq2)**

Suite à la vérification de la condition « c » (vraie), l'action « y » est inhibée (y=0), dans le cas contraire, l'action « y » est autorisée (y=1). Si le capteur associé à la contrainte « c » est défectueux, la durée associée à son état d'activation ou désactivation compense son fonctionnement.

Une condition « *c* » peut prendre trois aspects différents (Qamsane, Tajer, et Philippot 2016) :

- une condition simple utilisant des variables ou des fonctions booléennes
- une condition composée utilisant une séquence de variables ou des fonctions booléennes qui se précèdent
- une condition combinée contenant des conditions simples et composées telles que : $c \in \uparrow\downarrow e_i$ (événements associés aux capteurs) et $/c \in \uparrow\downarrow d_i$ (événements associés aux durées)

3.3 Synthèse locale pour obtenir le contrôleur local

Dans un travail précédent (Tahiri et al. 2018), nous avons proposé une approche pour réaliser une synthèse du contrôleur. Cette approche est basée sur l'extension des modèles des EPO contenant des gardes, des variables et des actions permettant de représenter sous forme d'expressions des restrictions de comportement du système et des fonctions sur des actions (Akesson et al. 2006). L'automate résultant est noté {(A_N)curr} (figure 8-b) pour un comportement normal.

L'équation de la spécification locale donnée dans la section (3.2.1) est transformée en automate, comme l'illustre la figure (8-a). Chaque spécification est composée d'un seul état et d'une transition rebouclante associée à l'événement contrôlable *y* et à la garde exprimée par {(A_N)curr!= x} ou {(A_F)curr!= x}, ce qui signifie que si l'état courant de ((A_N) curr) est différent de *x*, alors *y* est autorisé.

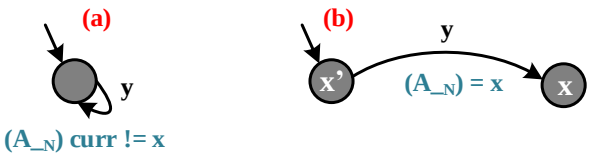


Figure 8 : (a) Modèle d'une spécification locale, (b) modèle de l'automate (A_N)curr

Pour obtenir les différents contrôleurs locaux pour chaque EPO, l'opération de synthèse est réalisée à l'aide du logiciel SUPREMICA entre les modèles {(A_N)curr} et l'automate représentant les spécifications locales. Sur l'exemple de la figure 8, le résultat de l'application de la spécification (a) au modèle (b) est la suppression de la transition associée à y.

3.4 Synthèse globale pour obtenir le contrôleur distribué

Un système manufacturier en cours de fonctionnement impose qu'il y ait des liens de synchronisme et de parallélisme entre les différents EPO. Ainsi, un EPO peut dépendre d'un autre pour garantir le comportement souhaité. Par conséquent, une communication entre plusieurs EPO est nécessaire. Pour se faire, une synthèse de contrôle global est nécessaire pour obtenir des contrôleurs distribués pour les comportements normaux et fautifs pour chaque EPO.

Cette synthèse consiste dans une première étape à agréger les contrôleurs locaux comme suit :

- Les événements contrôlables non-temporisés sont fusionnés dans des macro-états. Les états atteints par les événements contrôlables sont associés dans des macro-états liés par des événements incontrôlables (événements associés aux capteurs $\{\uparrow e_i, \downarrow e_i\}$) ou par des événements temporisés $\{\uparrow ck_i, \downarrow ck_i, d_i\}$. Si l'état du contrôleur local est associé au front montant d'un événement contrôlable, l'ordre est autorisé et appartient à l'ensemble *Ord.* S'il est associé à un front descendant de cet événement, l'ordre est inhibé et appartient à l'ensemble *Inh.*

- Les événements temporisés ↑ ↓ ck sont fusionnés dans des macro-états liés par des événements incontrôlables et des événements temporisés « di ». Si l'état du contrôleur temporisé local agrégé par la première agrégation est atteint par un événement correspondant à l'activation de l'horloge, cet événement appartient à un ensemble noté A_{CK}. S'il est atteint par un événement correspondant à la désactivation de l'horloge, cet événement appartient alors à un ensemble noté D_{CK}. La transition rebouclante devient la transition qui relie les deux macro-états contenant les deux ensembles (A_{CK} et D_{CK}).

La deuxième étape de cette synthèse globale est l'ajout des spécifications globales aux automates résultants afin d'obtenir les différents contrôleurs distribués. Un extrait d'un contrôleur distribué est présenté dans la figure 9.

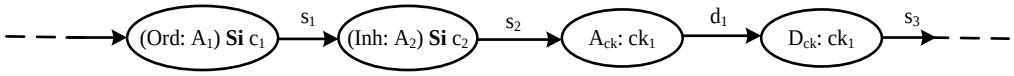
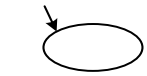
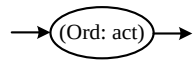
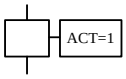
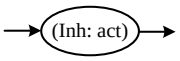
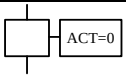
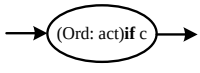
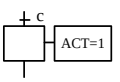
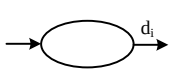
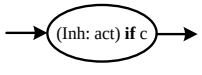
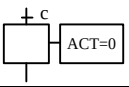


Figure 9 : Extrait d'un contrôleur distribué

3.5 Interprétation des contrôleurs distribués en Grafcet

Pour cette approche, nous avons choisi d'interpréter les contrôleurs distribués non-temporisés (CD) ainsi que ceux temporisés (CDT) en Grafcet (norme IEC 60848) pour une implémentation dans un langage de programmation standard d'un API. Plusieurs règles d'interprétation au grafcet sont données par les auteurs dans (Qamsane, Tajer et Philippot 2016). Un état d'un CD/CDT est interprété par une étape du grafcet, tandis qu'une transition d'un CD/CDT est transformée en une transition du grafcet correspondant. Dans ce travail, nous ajoutons une interprétation de l'événement temporisé. Dans une traduction en Grafcet, l'activation et la désactivation de l'horloge (Ack, Dck) ne sont pas interprétées, seule la durée est présentée. La transition marquée « *di* » sur le CDT est interprétée par une réceptivité de transition sous la forme de (di/Xji), ce qui signifie que si la variable associée à l'étape « *ji* » est activée, celle-ci doit être maintenue jusqu'à ce que la durée s'écoule ce qui provoque la transition à l'étape suivante « *ji + 1* ». L'autorisation d'un ordre (Ord : act) est traduite en action forcée à 1 (act = 1) tandis que l'inhibition de l'ordre (Inh : Act) est transformée en une action forcée à 0 (act = 0). Si une condition globale est associée à l'autorisation ou à l'inhibition de l'acte de commande, elle se traduit par une réceptivité à la transition qui précède l'étape associée à l'action (act = 1) ou (act = 0) (tableau 1).

Tableau 1 : Règles d'interprétation des éléments et des macro-états du CD/CDT en Grafcet

Elément du CD/CDT	Interprétation en Grafcet	Macro-état du CD/CDT	Interprétation en Grafcet
			
			
			
			

En appliquant l'interprétation du tableau 1 aux contrôleurs distribués non-temporisées et temporisés de l'exemple étudié dans cet article, on obtient les grafcets de commande suivants : deux grafcets de commande pour le fonctionnement normal des vérins A et B, et deux grafcets de commande pour leurs

fonctionnements fautifs. Les grafcets de commande du vérin A sont donnés dans la figure 10. Les deux grafcets peuvent être considérés comme étant des modèles génériques pour ce type d'actionneur quel que soit le système dont il appartient. En revanche, la contrainte globale doit être modifiée selon l'environnement de travail de cet actionneur. Par conséquent, ces modèles de grafcet peuvent être des éléments d'une bibliothèque d'éléments de commande.

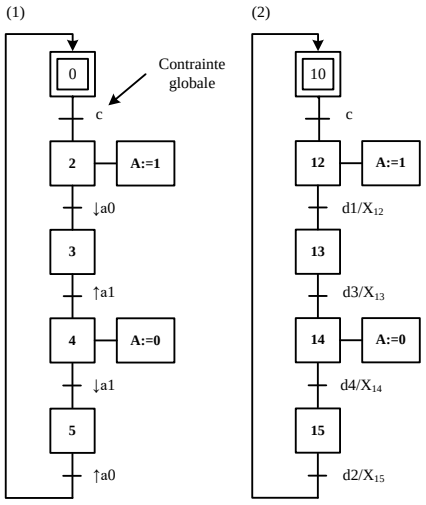


Figure 10 : Grafcets de commande du fonctionnement normal G_N (1) et fautif G_F (2) du vérin A

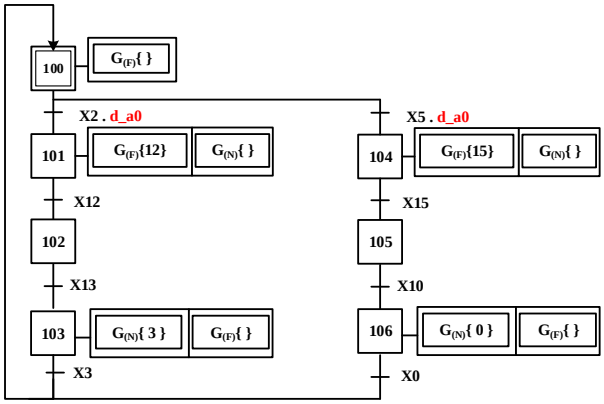


Figure 11 : Grafcet de reconfiguration pour le vérin A

3.6 Grafcet de reconfiguration

Les conditions de passage du mode normal ou mode défaillant sont définies par l'équation logique suivante :

Si X_i **et** f_s alors
(F: $G^*_{(F)}\{X_{ji}\}$ **et** F: $G^*_{(N)}\{\ \}$) **(eq3)**
Sinon Si X_{ji} **et** $f_s = 0$ Alors
(F: $G^*_{(N)}\{X_i\}$ **et** F: $G^*_{(F)}\{\ \}$)

- F est l'opération de forçage
- $G^*_{(N)}$ est le grafcet associé au contrôleur distribué du comportement normal
- $G^*_{(F)}$ est le grafcet associé au contrôleur distribué du comportement fautif
- X_i est la variable booléenne associée à l'étape « i » du $G^*_{(N)}$
- X_{ji} sa variable correspondante associée à l'étape « ji » du $G^*_{(F)}$

L'expression de l'équation (3) signifie que si X_i est active et qu'un défaut de capteur est détecté (f_s), il y a un passage en mode défaillant en forçant le grafcet $G^*_{(F)}$ à s'activer à l'étape X_{ji} et en désactivant le grafcet du mode normal $G^*_{(N)}$.

Un exemple de grafcet de reconfiguration lors de la détection défaut sur la désactivation (d_ ↓ a0) et l'activation (d_ ↑ a0) du capteur a0 est donné dans la figure 11.

3.7 Résultats

L'application de cette approche sur le vérin A de l'exemple donne les résultats présentés dans les tableaux 2 et 3. Ils sont identiques pour le vérin B.

Tableau 2 : Résultats de l'étape de modélisation dans l'approche distribuée pour le vérin A

	$EPO_{N(A)}$	$EPO_{F(A)}$	$SUP_{N(A)}$	$SUP_{F(A)}$	Contrôleur $local_{N(A)}$	Contrôleur $local_{F(A)}$	Contrôleur $distribué_{N(A)}$	Contrôleur $distribué_{F(A)}$
Etats	6	10	6	10	6	10	4	4
Transitions	10	18	10	18	6	14	4	4

Tableau 3 : Grafcets de contrôle pour le vérin A

	$Grafcet_{N(A)}$	$Grafcet_{F(A)}$	Grafcet de $reconfiguration_{(A)}$
Etapes	5	5	7
Transitions	5	5	8

L'approche distribuée proposée permet de s'affranchir d'étapes de composition entre modèles, source d'explosion de l'espace d'états. Le tableau 2 illustre très bien ces propos puisqu'aucun modèle ne dépasse les 10 états. Il est vrai, par contre, que la proposition implique pour chaque EPO, un modèle de contrôleur normal, un modèle de contrôleur du fonctionnement fautif et aussi un grafcet de reconfiguration. Il y a un risque de multiplication des grafcets et donc un risque de perdre la vision globale des spécifications issues du cahier des charges pour l'opérateur dans le cas de systèmes complexes. C'est pourquoi la présentation des données au niveau de l'interface opérateur est capitale.

La démarche distribuée peut être comparée à une démarche centralisée pour montrer tout de même sa pertinence. Nous avons donc réalisé la version centralisée de la démarche de reconfiguration. Elle reprend les étapes génériques décrites en section 2, le comportement normal (CONT_N) et le comportement fautif (CONT_F) étant traités indépendamment l'un de l'autre. La démarche intègre deux étapes supplémentaires : une étape de composition des contrôleurs CONT_N et CONT_F pour obtenir le contrôleur centralisé (CONT) et une étape de composition pour prendre en compte les spécifications de reconfiguration et obtenir le contrôleur reconfigurable global. L'application à l'exemple donne un contrôleur global comprenant 172 800 états et 52 800 transitions, rendant la démarche difficilement praticable.

4 Conclusion et perspectives

Pour répondre aux problèmes de la sécurité opérationnelle dans le domaine de contrôle des systèmes manufacturiers, la mise en œuvre des méthodes formelles est nécessaire. Dans ce contexte, il est important de surveiller ces systèmes et d'offrir une solution alternative pour maintenir la production. Ainsi, une reconfiguration du contrôle des systèmes manufacturiers est demandée. Pour atteindre cet objectif, cet article présente un cadre de reconfiguration de la commande fondé sur un contrôle distribué pour lever les problèmes liés aux approches centralisées. Le principal avantage de cette démarche de reconfiguration est l'exploitation du contrôle distribué qui évite d'une part l'explosion combinatoire récurrente dans les approches centralisées comme le montre la section (3.7). D'autre part, elle permet de reconfigurer le seul EPO défectueux sans reconfigurer tout le contrôleur du système. En outre, la compensation des événements des capteurs défectueux par des événements temporisés qui garantissant le même comportement évite l'utilisation des éléments redondants qui peut être coûteuse.

Plusieurs perspectives de recherche seraient intéressantes pour améliorer les performances de la reconfiguration de la commande des systèmes manufacturiers :

(1) La commande obtenue est sûre de fonctionnement et assure le contrôle du comportement fautif du système, mais elle peut être bloquante. Pour pallier ce problème, une vérification des automates des contrôleurs distribués avant la traduction en Grafcet par model-checking est nécessaire.

(2) L'approche proposée implique une sollicitation forte des grafcets de reconfiguration en cas de détections de défauts mais aussi en cas de remise en marche nominale du système. Ce grafcet représente un « switch » entre les modes de marche. Il convient donc d'assurer le non-blocage également de l'ensemble de la commande implantée mais aussi de l'atteignabilité de chaque étape des grafcets.

(4) Les grafcets des contrôleurs résultants sont obtenus automatiquement à partir de l'étape d'interprétation des contrôleurs distribués. Cependant, une optimisation de cette interprétation semble possible dans le cadre où certaines étapes générées peuvent parfois être supprimables.

(5) Le type de défaut étudié dans cet article est associé aux capteurs du système, tandis qu'un défaut d'un actionneur peut être détecté aussi. Dans ce cas, une redondance matérielle est nécessaire pour assurer l'action souhaitée. Le principe de la démarche proposée dans cet article est relativement similaire : le fonctionnement normal reste le même que pour la solution proposée pour un défaut capteur, par contre, le grafcet de reconfiguration gère la commutation entre les deux grafcets associés à l'actionneur fautif et l'actionneur redondant.

(6) Un autre cas d'étude pour la reconfiguration peut être la perspective de nos prochains travaux en intégrant une nouvelle production à celle en cours.

Références

Akesson, K., M. Fabian, H. Flordal, et R. Malik. 2006. « Supremica - An integrated environment for verification, synthesis and simulation of discrete event systems ». In *2006 8th International Workshop on Discrete Event Systems*, 384-85. https://doi.org/10.1109/WODES.2006.382401.

Balemi, S., G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, et G. F. Franklin. 1993. « Supervisory control of a rapid thermal multiprocessor ». *IEEE Transactions on Automatic Control* 38 (7): 1040-59. https://doi.org/10.1109/9.231459.

Bévan, Romain. 2013. « Approche composant pour la commande multi-versions des systèmes transitiques reconfigurables ». Thesis, Lorient. http://www.theses.fr/2013LORIS311.

Blanke, Mogens, Michel Kinnaert, Jan Lunze, et Marcel Staroswiecki. 2016. « Introduction to Diagnosis and Fault-Tolerant Control ». In *Diagnosis and Fault-Tolerant Control*, 1-35. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-47943-8_1.

Bordoloi, Sanjeev K., William W. Cooper, et Hirofumi Matsuo. 1999. « Flexibility, Adaptability, and Efficiency in Manufacturing Systems ». *Production and Operations Management* 8 (2): 133-50. https://doi.org/10.1111/j.1937-5956.1999.tb00366.x.

Hélouët, Loïc, Hervé Marchand, Blaise Genest, et Thomas Gazagnaire. 2014. « Diagnosis from Scenarios ». *Discrete Event Dynamic Systems* 24 (4): 353-415. https://doi.org/10.1007/s10626-013-0158-2.

Koren, Y., U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, et H. Van Brussel. 1999. « Reconfigurable Manufacturing Systems ». *CIRP Annals* 48 (2): 527-40. https://doi.org/10.1016/S0007-8506(07)63232-6.

Koren, Yoram, et Moshe Shpitalni. 2010. « Design of reconfigurable manufacturing systems ». *Journal of Manufacturing Systems* 29 (4): 130-41. https://doi.org/10.1016/j.jmsy.2011.01.001.

Kumar, Ratnesh, Vijay K. Garg, et Steven I. Marcus. 1991. « On controllability and normality of discrete event dynamical systems ». In . https://doi.org/10.1016/0167-6911(91)90061-I.

Lee, Eun Joo. 2006. « The dynamic reconfiguration control of manufacturing system : approach by the synthesis of control ». Theses, Ecole Centrale de Lille. https://tel.archives-ouvertes.fr/tel-00121727.

Philippot, A., et V. Carré-Ménétrier. 2011. « Methodology to obtain local discrete diagnosers: Submission for special session on diagnosis of DES: Application on a benchmark ». In *2011 3rd International Workshop on Dependable Control of Discrete Systems*, 47-52. https://doi.org/10.1109/DCDS.2011.5970317.

Philippot, Alexandre. 2006. « Contribution au diagnostic décentralisé des systèmes à événements discrets: Application aux systèmes manufacturiers ». Reims, France: Reims champagne Ardenne University.

Qamsane, Yassine, Abdelouahed Tajer, et Alexandre Philippot. 2016. « A Synthesis Approach to Distributed Supervisory Control Design for Manufacturing Systems with Grafcet Implementation ». *International Journal of Production Research*, septembre. http://tandfonline.com/doi/abs/10.1080/00207543.2016.1235804.

Ramadge, P. J. G., et W. M. Wonham. 1989. « The control of discrete event systems ». *Proceedings of the IEEE* 77 (1): 81-98. https://doi.org/10.1109/5.21072.

Rawat, Govind Singh, Ashutosh Gupta, et Chandan Juneja. 2018. « Productivity Measurement of Manufacturing System ». *Materials Today: Proceedings*, International Conference on Processing of Materials, Minerals and Energy (July 29th – 30th) 2016, Ongole, Andhra Pradesh, India, 5 (1, Part 1): 1483-89. https://doi.org/10.1016/j.matpr.2017.11.237.

Riera, B., A. Philippot, R. Coupat, F. Gellot, et D. Annebicque. 2015. « A non-intrusive method to make safe existing PLC Program ». *IFAC-PapersOnLine*, 9th IFAC Symposium on Fault Detection, Supervision andSafety for Technical Processes SAFEPROCESS 2015, 48 (21): 320-25. https://doi.org/10.1016/j.ifacol.2015.09.547.

Tahiri, Imane, Alexandre Philippot, Véronique Carré-Ménétrier, et Abdelouahed Tajer. 2018. « Timed Synthesis Approach for Tolerant-Fault Control of Discrete Event Systems (DES) ». In . marrakech-Morocco.

———. 2019. « Time-Based Estimator for Control Reconfiguration of Discrete Event Systems (DES) ». In . Paris-France.

Tajer, Abdelouahed, Alexandre Philippot, et Véronique Carré-Ménétrier. 2013. « Centralised controller for manufacturing systems through liveness extraction approach ». *International Journal of Systems, Control and Communications* 5 (3-4): 189-213. https://doi.org/10.1504/IJSCC.2013.058175.

Terkaj, Walter, Tullio Tolio, et Anna Valente. 2009. « A Review on Manufacturing Flexibility ». In *Design of Flexible Production Systems: Methodologies and Tools*, édité par Tullio Tolio, 41-61. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-85414-2_3.

Wonham, W. M., Kai Cai, et Karen Rudie. 2018. « Supervisory control of discrete-event systems: A brief history ». *Annual Reviews in Control*, avril. https://doi.org/10.1016/j.arcontrol.2018.03.002.

Zaytoon, J., et B. Riera. 2017. « Synthesis and implementation of logic controllers – A review ». *Annual Reviews in Control* 43 (janvier): 152-68. https://doi.org/10.1016/j.arcontrol.2017.03.004.