UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

ÉCOLE Doctorale SCIENCES DU NUMÉRIQUE ET DE L'INGÉNIEUR

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : ***AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE***

Spécialité :Automatique et Traitement de Signal

Présentée et soutenue publiquement par

IMANE TAHIRI

Le 28 août 2020

Contribution à la conception d'une commande reconfigurable et tolérante aux fautes pour les Systèmes Automatisés de Production

---

Thèse dirigée par **VERONIQUE CARRE-MENETRIER**

JURY

Mr Armand TOGUYENI,	Professeur des Universités,	CRIStAL – Centrale Lille Institut	Rapporteur
Mr Said AMARI,	Maître de conférences HDR	LURPA – École Normale Supérieure Paris-Saclay	Rapporteur
Mr Bernard RIERA,	Professeur des Universités	CReSTIC – Univérsité de Reims Champagne-Ardenne	Examinateur
Mr Ahmed NAIT SIDI MOH,	Maître des conférences HDR	LTI – Université de Picardie Jules Verne	Examinateur
Mr Alexandre PHILIPPOT,	Maître des conférences HDR,	CReSTIC – Université de Reims Champagne-Ardenne	Examinateur Co-encadrant
Mr Abdelouahed TAJER,	Professeur des Universités,	LGeCOS – Université Cadi Ayyad	Examinateur Co-encadrant
Mme Véronique CARRE-MENETRIER	Professeur des Universités	CReSTIC – Université de Reims Champagne-Ardenne	Examinatrice Directrice de thèse

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les systèmes sont devenus complexes et vulnérables en raison des contraintes induites par un environnement incertain, changeant et dominé par une forte concurrence internationale. L'impact de ce changement dans l'industrie se traduit par la nécessité de disposer de systèmes flexibles capables de s'adapter aux changements de production pour répondre à des critères de productivité et de qualité tout en tant réduisant les risques de défaillances. Pour relever ce défi, et pour obtenir un fonctionnement productif et continu, l'étude du *système de contrôle* représente la solution fondamentale pour l'amélioration des performances opérationnelles et l'augmentation de la réactivité.

Dans le domaine du contrôle automatique, l'un des axes scientifiques intéressants est l'étude du contrôle des systèmes à événements discrets (SED) qui est présent dans différents domaines d'application tels que les systèmes de transport, les systèmes électriques, les systèmes automatisés de production (SAP), etc. Dans le cadre des systèmes automatisés de production, les approches industrielles traditionnelles se basent sur une implémentation directe d'un programme de contrôle qui doit être « raffiné » pour obtenir la réalisation du contrôle cible du SAP. Ce raffinement est obtenu grâce à des procédures de test et de validation, mais ne garantit pas l'optimalité et la validité du contrôleur résultant. Quant aux approches issues du monde académique, elles consistent à garantir formellement le résultat. Ces méthodes formelles reposent sur une description mathématique de la problématique à résoudre tout en exploitant des outils permettant une résolution automatique.

Introduite pour la première fois par Ramadge et Wonham en 1989 (Ramadge et Wonham 1989), la théorie de contrôle par supervision (SCT) est considérée comme l'une des théories académiques majeures pour le contrôle des SED. Avec les réalisations scientifiques des dernières décennies, le cadre de la SCT forme un paradigme systématiquement formel pour synthétiser les contrôleurs des SED. Cependant, l'application de la SCT aux problèmes industriels présente un inconvénient majeur qui est l'explosion combinatoire de l'espace d'états. Par conséquent, différentes architectures de contrôle telles que le contrôle modulaire et distribué ont été proposées pour surmonter ce problème et pour rendre la phase de l'implémentation des lois de commande calculées dans des automates programmables industriels (API) moins complexe.

Par ailleurs, afin de faire face aux dysfonctionnements qui peuvent entraver le bon fonctionnement des SAP, il s'avère nécessaire d'élaborer des méthodes de contrôle tolérant aux fautes ou reconfigurable. Ces méthodes consistent à assurer la continuité et la poursuite des tâches pour lesquelles le SAP a été conçu, tout en garantissant la sûreté de fonctionnement. L'utilisation des systèmes de contrôle reconfigurable est une solution prometteuse qui améliore les performances et la productivité des systèmes et prend en charge la réactivité exigée dans les systèmes de contrôle. Un système de contrôle reconfigurable est capable de changer les caractéristiques de l'architecture de contrôle pour s'adapter aux variations du processus contrôlé, tout en maintenant autant que possible la stabilité et les performances du système d'origine (Moor 2016).

Les travaux présentés dans ce mémoire entrent dans le cadre de la reconfiguration de la commande des systèmes automatisés de production vus comme une classe de SED. La méthodologie de reconfiguration adoptée dans ce manuscrit est basée sur la théorie de contrôle par supervision SCT et déclenchée suite à une détection des défauts sur la partie opérative (commande tolérante aux fautes). La contribution de la thèse cherche à répondre à la question suivante : comment assurer un processus de reconfiguration afin d'améliorer les performances globales des systèmes automatisés de production ? Le cadre de la thèse repose sur les travaux réalisés par les membres de l'équipe Commande et Diagnostic des SED autour des problématiques de synthèse de commande [(Tajer et al. 2013), (Philippot et al. 2005)], du contrôle distribué (Qamsane et al. 2016) et de l'implémentation de la commande dans des automates programmables industriels (Zaytoon et Riera 2017).

Contribution de la thèse

L'objectif des travaux de ce mémoire est, par conséquent, de fournir une méthodologie de conception d'une commande reconfigurable et tolérante aux fautes, implantable dans un API et utilisable dans le domaine industriel. L'approche est basée selon le cycle en V présenté figure 1. Le contrôleur reconfigurable/tolérant aux fautes est obtenu à partir d'un modèle de comportement de la partie opérative, d'un ensemble de spécifications de sécurité (ce que le système ne doit pas faire), de vivacité (ce que le système doit faire) et de reconfiguration.

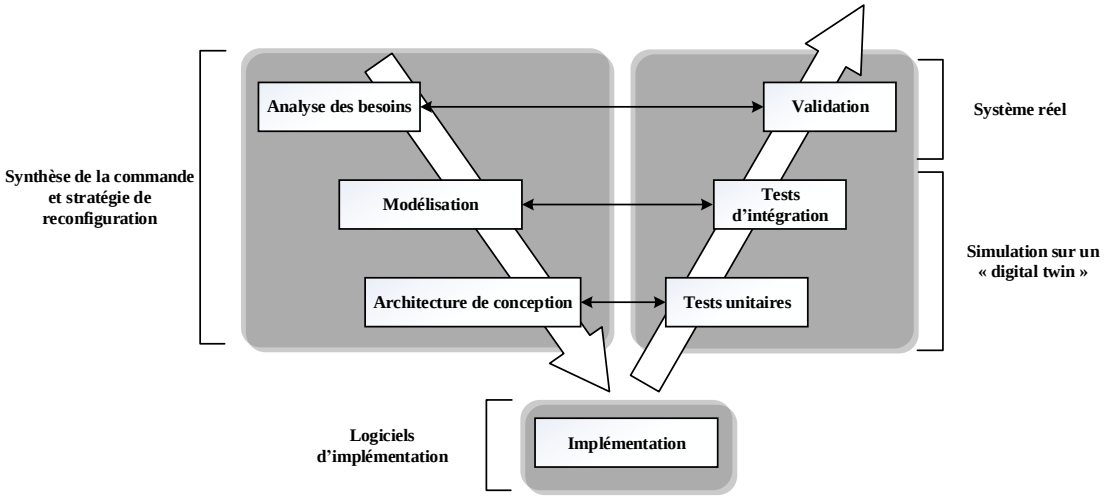


Figure 1 : Cycle de conception d'une commande reconfigurable et tolérante aux fautes

Dans l'objectif de supporter la démarche proposée, différents apports théoriques sont nécessaires et sont illustrés figure 2.

- Pour créer un contrôleur de mode de fonctionnement dégradé lors d'une demande de reconfiguration, la modélisation du comportement de la partie opérative présentée dans des travaux précédents (Qamsane et al. 2016) et (Philippot 2006) est étendue à une modélisation temporisée où des événements sont intégrés pour compenser le fonctionnement des éléments fautifs ❶.

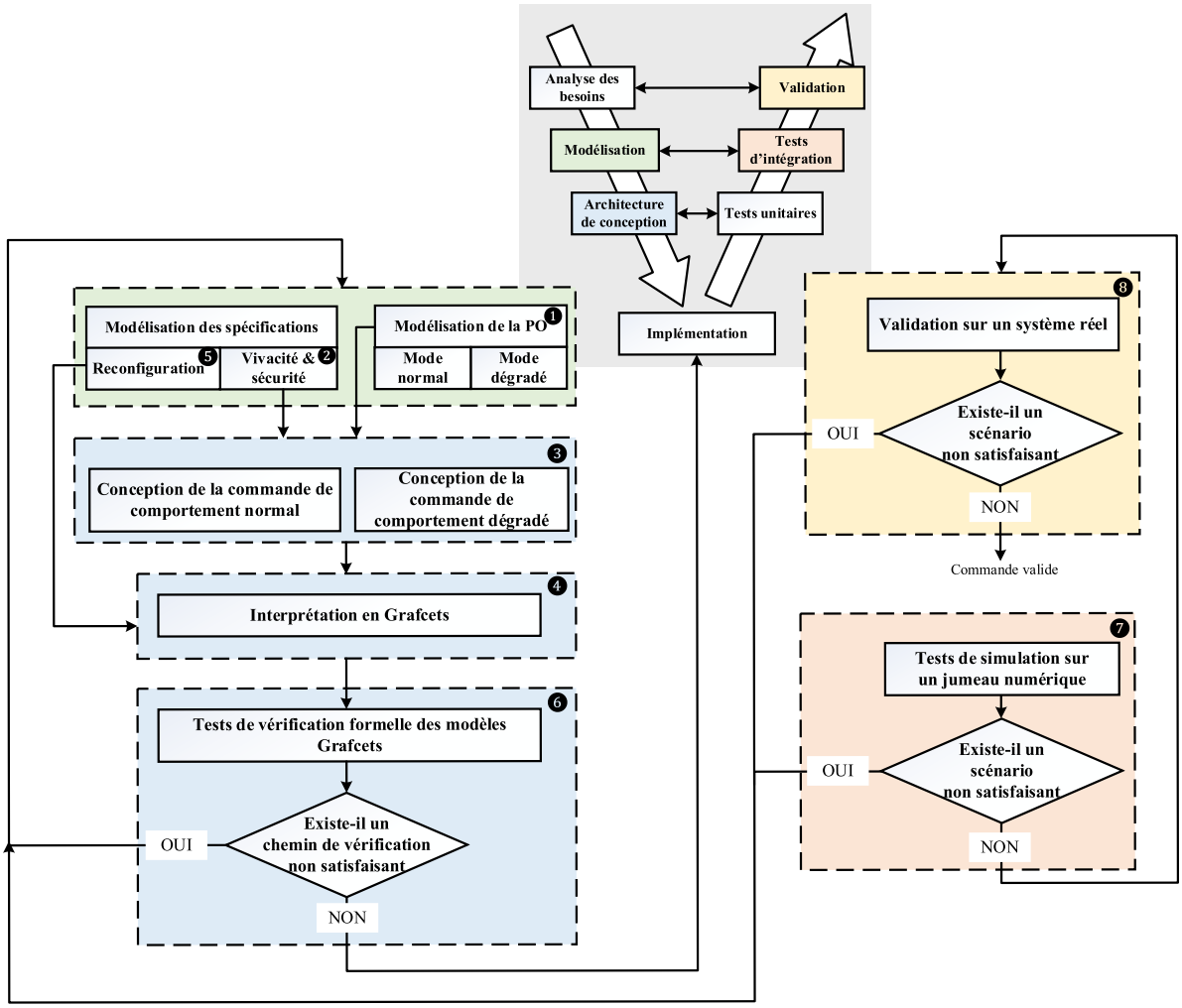


Figure 2 : Architecture formelle pour la synthèse d'une commande reconfigurable et tolérante aux fautes

- Les spécifications de vivacité et de sécurité ❷ sont représentées formellement d'abord par des équations logiques booléennes puis traduites en automates à états finis étendus, facilitant la phase de la synthèse de la commande et diminuant l'explosion combinatoire des modèles résultants.
- La théorie de contrôle par supervision (SCT) est adaptée pour synthétiser deux types de contrôleur ❸ : un pour le comportement normal et le deuxième pour le comportement dégradé, afin d'assurer une redondance au niveau de la commande pour garantir la continuité du fonctionnement lors de la détection des défauts. Une architecture distribuée est utilisée afin d'éviter l'explosion combinatoire liée à l'exploitation d'une architecture centralisée.
- Les modèles des contrôleurs sont interprétés en grafcet ❹ selon des règles de traduction et dans l'objectif d'être compris par un opérateur puis implantés dans un des langages normalisés d'automates programmables industriels.
- Un ensemble de spécifications de reconfiguration sous forme d'équations logiques est intégré ❺ pour assurer la reconfiguration et la commutation entre les deux modes de fonctionnement : mode normal – mode dégradé. Ces spécifications de reconfiguration sont traduites également en grafcet permettant de gérer les deux grafcets de commande.

- Les propriétés de non-blocage et d'atteignabilité de l'ensemble des grafcets résultants de l'application de l'approche de la reconfiguration sont vérifiées formellement par Model-Checking ❻. Si la phase de vérification est satisfaisante, les grafcets sont traduits dans un langage de programmation et testés sur un jumeau numérique ❼ avant d'être implémentés dans un API réel ❽. Si la phase de vérification formelle ou de simulation n'est pas satisfaisante, le concepteur doit apporter des modifications au niveau de la modélisation des spécifications et/ou de la partie opérative.

Organisation du mémoire

Le chapitre 1 effectue un tour d'horizon sur le contexte générale associé aux systèmes automatisés de production et à la sûreté de fonctionnement. Plongé dans un contexte lié à l'existence d'aléas de fonctionnement, nous montrons la nécessité d'un système sûr de fonctionnement. L'étude de ce type de systèmes permet, d'une part, de définir les entraves à la sûreté de fonctionnement et, d'autre part, de déterminer les moyens pour assurer la sûreté parmi lesquels nous retrouvons les notions de reconfiguration et de tolérance aux fautes.

Le chapitre 2 a pour objectif de présenter les travaux de recherche sur les systèmes de contrôle reconfigurable. Tout d'abord, nous abordons le concept de la reconfiguration des SAP et sa relation avec le temps réel puis les caractéristiques associées aux systèmes de contrôle reconfigurable, leurs objectifs et les moyens à mettre en œuvre pour atteindre les objectifs. Enfin, nous présentons une étude bibliographique sur la reconfiguration et la tolérance aux fautes des SED.

Le chapitre 3 présente les différentes contributions à la synthèse et la reconfiguration de la commande des SAP. Tout d'abord, les problèmes liés aux approches de commande basées sur la SCT, les limitations et les adaptations pour pallier ces limitations sont présentés. Ensuite, un nouveau cadre pour le contrôle tolérant et reconfigurable des SAP est proposé. Puis, la formalisation des éléments de la méthodologie de la reconfiguration distribuée de la commande est présentée. Enfin, une approche de vérification avant l'implémentation de la commande est proposée. Les concepts utilisés dans ce chapitre sont illustrés sur un exemple didactique.

Le chapitre 4 est consacré à une application de l'approche sur un système de production réel. Ce système est constitué de plusieurs sous stations contrôlées chacune par un automate programmable industriel. L'ensemble de la méthodologie de l'approche de la reconfiguration est appliqué à une seule station pour en faciliter la compréhension mais aussi à son jumeau numérique dans la phase de test.

Enfin, le mémoire se termine par une conclusion générale résumant l'ensemble du travail réalisé et précisant des directions de recherches futures.

CHAPITRE I

Comment établir un système automatisé de production sûr de fonctionnement ?

1 Introduction

L'introduction fréquente de nouveaux produits, les changements imprévisibles de la demande et les changements technologiques obligent les entreprises à réagir rapidement et à moindre coût. Dans cet environnement, les entreprises doivent privilégier des systèmes automatisés de production (SAP) capables de réagir rapidement et efficacement aux changements. Par conséquent, le concept de systèmes automatisés de production reconfigurables (SAPR) a été inventé en prenant en compte des propriétés d'ajustement dynamique de la capacité/fonctionnalité de production en fonction des fortes fluctuations du marché pour des changements rapides dans la structure du système (Bortolini et al. 2018).

Cette thèse traite de la reconfiguration de la commande des systèmes automatisés de production tout en préservant la sûreté de fonctionnement. Dans ce contexte, ce chapitre présente, dans une première partie, la définition des systèmes automatisés de production (histoire, objectifs et structure) et, dans une seconde partie, la sûreté de fonctionnement (étude de propriétés, entraves et moyens conduisant à la sûreté de fonctionnement).

2 Les systèmes automatisés de production : Définition et problématique

2.1 Définition d’un système

Dans la littérature, de nombreux travaux abordent le concept de système. (Sokolowski et Banks 2012) définissent un système comme une collection de différents éléments qui produisent ensemble des résultats que les éléments seuls ne peuvent pas obtenir. Ces derniers peuvent être des personnes, du matériel, des installations, des structures politiques, des documents ou tout ce qui est requis en tant que qualités, propriétés, caractéristiques, fonctions, comportement et performances. Jennings (Jennings 2000) présente une définition plus courte « un système est un réseau de composants créés pour résoudre des problèmes qui dépassent leurs capacités individuelles » en mettant l’accent sur l’objectif du système. Sayama (Sayama 2015) inclut dans sa définition les résultats attendus des systèmes : « un système est un réseau constitué de plusieurs composants qui interagissent entre eux, qui peuvent évoluer par une auto-organisation et qui permettent le développement d'un comportement émergent à des échelles macroscopiques». Dans (Whitehead et al. 2012), les auteurs ont considéré un système comme étant une structure organisée qui a pour but la résolution d’un problème et un ensemble constitué des éléments interdépendants (composants, entités, facteurs, agents, membres, etc.) influencés les uns par les autres (directement ou indirectement) pour maintenir leur activité et l’existence du système pour atteindre le but commun.

2.2 Nature d'un système

Les systèmes peuvent être divisés en deux types : les systèmes naturels et les systèmes conçus par l'homme (Wallner 1999). Le système respiratoire du corps humain, par exemple, est un système naturel. Au contraire, l'ensemble des opérations d'un aéroport est un système conçu par l'homme. Les deux sont composés d'un ensemble d'éléments qui interagissent les uns avec les autres pour un objectif spécifique. La perception d'un système est différente selon le référentiel adopté pour le caractériser. En effet, un système peut être décrit d'un point de vue externe ou interne (Trentesaux 2002) (figure 3) :

- Vue externe : Une analyse d'un point de vue externe considère le système comme une boîte noire. Le système est donc analysé au niveau des objectifs pour lesquels il est donné et au niveau de son interaction avec l'environnement.
- Vue interne : Une analyse d'un point de vue interne décrit un système par sa structure (ou arrangement), son comportement (ou ses fonctionnalités) et la dynamique en cours d'exécution (ou son progrès). Cette approche de caractérisation correspond à la compréhension du système selon sa composition (quel système est-ce ?) et ses actions (quelle est son utilisation ?) (Moigne 1994). La figure 3 illustre ces propos.

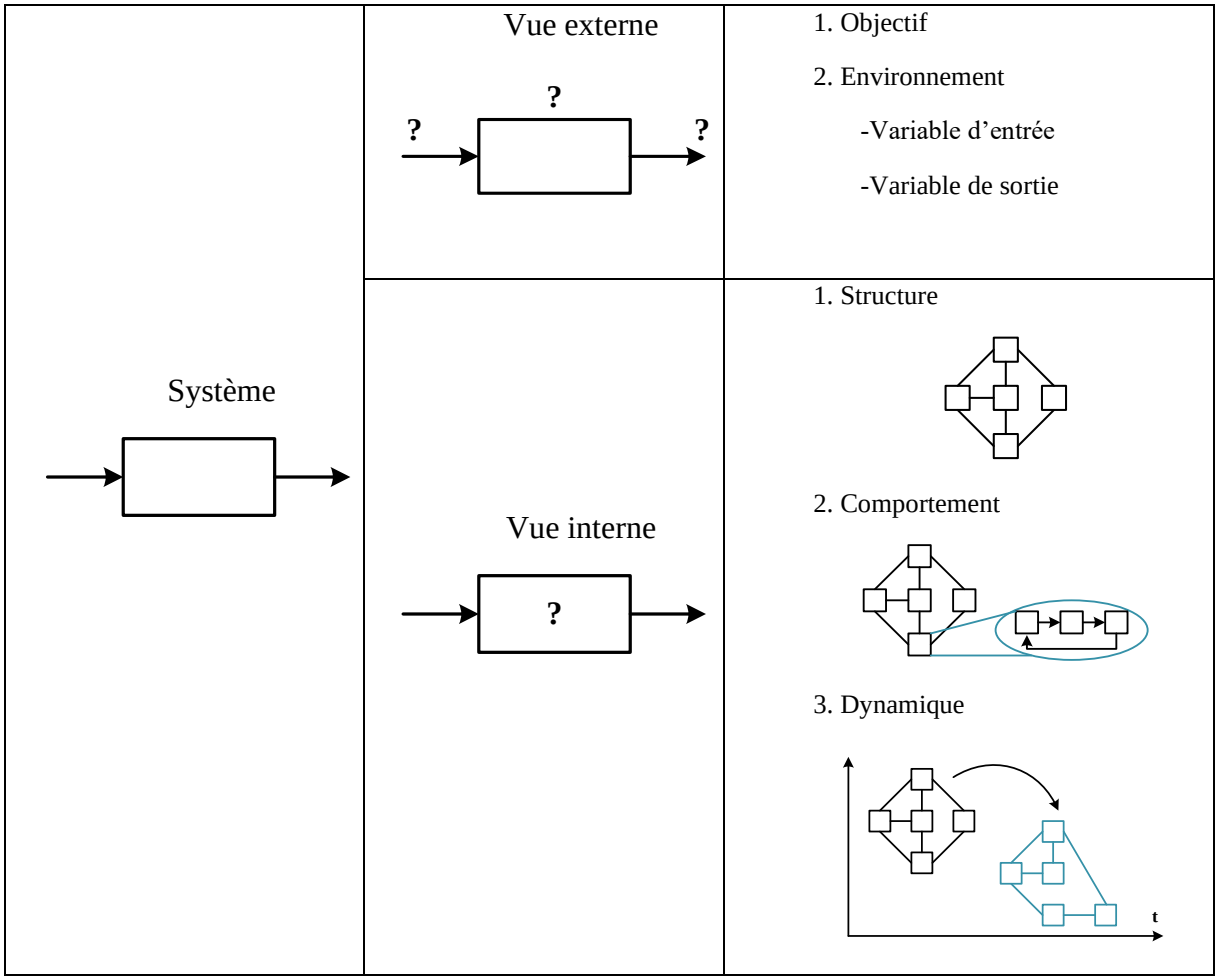


Figure 3: Nature et caractéristiques d'un système

2.3 Histoire et définition des systèmes automatisés de production

Les systèmes de production (figure 4) ont connu une grande évolution au cours des derniers siècles. La mécanisation a remplacé l'artisanat pendant la 1ère révolution industrielle. Elle se situe au moment de l'exploitation du charbon et de la mise au point de la machine à vapeur par James Watt en 1769. Cette révolution correspond à l'utilisation de ce type de machine comme moteur pour actionner les machines permettant des cadences accrues entraînant une fabrication plus importante, et donnant naissance à des produits en petites séries.

La 2ème révolution pendant le 19ème siècle est caractérisée par un considérable développement des techniques et des méthodes de production de biens matériels grâce à l'apparition du pétrole et de l'électricité. Cette révolution a permis la production de séries de produits à une plus grande échelle pour répondre à une demande du marché très élevée.

La 3ème révolution a été concomitante avec l'émergence de l'électronique (Chiron 2008). Ce sont les débuts de la robotique, de la flexibilité des outils de production et de la production en grandes séries. Ces différents concepts ont conduit à l'essor de l'automatisation soulageant ainsi les ouvriers des tâches les plus difficiles. Cette révolution a engendré un vaste domaine de recherche contenant différents champs d'application (Rohée 2008).

Après les révolutions industrielles de la mécanisation, de la production de masse et celle de l'automatisation, une 4ème révolution industrielle est annoncée depuis 2011. Appelée *Industrie 4.0, Usine du futur*, *Usine Connectée* ou bien encore *Smart industry,* elle représente l'arrivée de la numérisation (Beier et al. 2020) (Angelopoulou et al. 2020). L'industrie devient un système global interconnecté dans lequel les machines, les systèmes et les produits communiquent. Cette révolution a pour but de conserver et développer une activité industrielle forte, innovante et créatrice d'emploi. L'interconnexion permet aux usines de devenir plus compétitives grâce à un niveau de performance et de sûreté accru. Elles produisent ainsi à la fois des produits personnalisés et les services associés. L'industrie dite « flexible » s'engage à répondre à ces exigences de produits uniques et personnalisés tout en conservant des coûts équivalents, et cela malgré les faibles volumes de production engendrés.

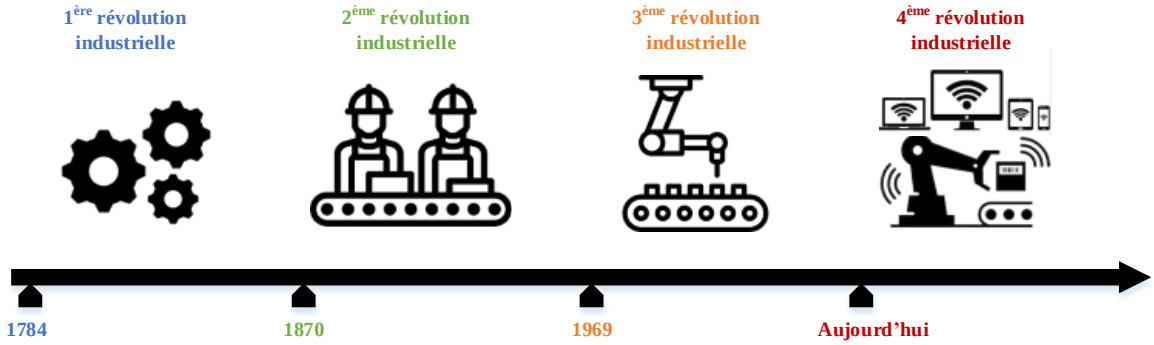


Figure 4: Etapes d'évolution des systèmes de production

Avec l'évolution des technologies, les systèmes automatisés de production (SAP) ne se retrouvent pas dans une seule définition. Plusieurs travaux de recherche dans ce domaine ont tenté de les définir et de les caractériser. Selon Staroswiecki (Staroswiecki et Bayart 1994) , le SAP est conçu pour vérifier les transformations effectuées par le processus physique. Le contrôle est effectué à l'aide des capteurs et des actionneurs. Le concept adopté par Verlinde (Verlinde 1989) pour les SAP est basé sur les fonctions du système qui sont : diriger, maintenir,

surveiller, et sécuriser. Cependant, Bayart (Bayart 1994) ne prend pas en compte les opérateurs dans son modèle. Le SAP selon l'auteur est considéré comme étant une interface entre le processus physique et les opérateurs. Pour Cauffriez (Cauffriez 2005), le SAP prend en charge les principes d'un système physique, qu'il soit interne ou externe. Bien que ces définitions caractérisent un SAP, une autre définition adaptée au contexte de nos travaux est celle de Panetto (Panetto 1991) et de Chiron (Chiron 2008), qui considèrent qu'un SAP est composé d'une partie opérative (PO) et d'une partie commande (PC).

2.4 Objectifs et structure d'un système automatisé de production

2.4.1 Objectifs d'un SAP

L'automatisation a permis aux entreprises d'améliorer leur compétitivité (coûts des produits, qualité, adaptabilité à la demande, ...). Elle a pour objet d'associer les moyens de production et les moyens de commande automatique qui permettent d'assurer la reproductibilité du résultat de la manière la plus autonome possible (plus au moins indépendante des interventions humaines). L'automatisation s'exprime selon les objectifs suivants (Berruet et al. 2007) :

- Augmenter la productivité : fabriquer le maximum de produits pendant le minimum de temps,
- Améliorer la flexibilité de production : fabriquer le maximum de variétés de produits avec le même équipement. Ceci nécessite l'utilisation de systèmes de production ayant la capacité, d'une part, de s'adapter rapidement aux changements des caractéristiques des produits à fabriquer en reconfigurant la circulation des produits et des opérations, et, d'autre part, de répondre dans les plus brefs délais aux variations du volume des commandes, sans créer des stocks inutiles,
- Améliorer la qualité des produits,
- Intégrer gestion et production : contrôler le flux de production, disposer des données technico-économiques sur la production, et simuler des programmes de production,
- Améliorer les conditions de travail du personnel : supprimer la pénibilité et améliorer la sécurité,
- Permettre de réaliser des travaux dans des milieux hostiles et suppléer l'homme dans des situations de conduite dangereuses (centrales nucléaires, usines chimiques, ...).

La compétitivité des entreprises se joue sur deux points principaux, le premier est de gagner un marché par des produits moins chers et de meilleure qualité, et le deuxième est de conserver le marché par des produits irréprochables en constante évolution et toujours concurrentiels. La compétitivité est fondée alors sur :

- La qualité,
- Les coûts où un compromis entre le coût du produit, sa qualité, la quantité et les délais des demandes doit être trouvé,
- L'innovation pour laquelle la recherche et l'adaptation aux nouvelles technologies doivent être omniprésentes,
- La flexibilité qui représente la capacité à effectuer des opérations différentes lors du changement de processus opératoire ou des conditions,
- Le management qui nécessite une organisation structurée de l'entreprise permettant le travail en groupe (responsabilité repartie, précise, communication, ...).

2.4.2 Structure d'un SAP

Dans ce mémoire, nous prenons comme référence le fait que le système automatisé est constitué d'une partie opérative (PO) et d'une partie commande (PC) qui dialoguent ensemble. La PO regroupe l'ensemble des opérateurs techniques qui assurent et contrôlent la production des effets utiles pour lesquels le système automatisé a été conçu. C'est la PO qui agit directement sur la matière d'œuvre pour lui apporter une valeur ajoutée. La PC est l'ensemble des moyens de traitement de l'information qui assure la commande et la coordination des tâches c'est-à-dire de la succession des actions de la PO, à la place de l'opérateur et à partir des programmes préétablis. Deux types d'échanges d'information existent entre la PC et la PO (figure 5) :

- L'émission des ordres aux signaux de commande vers les pré-actionneurs de la PO.
- La réception des comptes rendus de la PC par l'intermédiaire des organes de saisie d'information (capteurs).

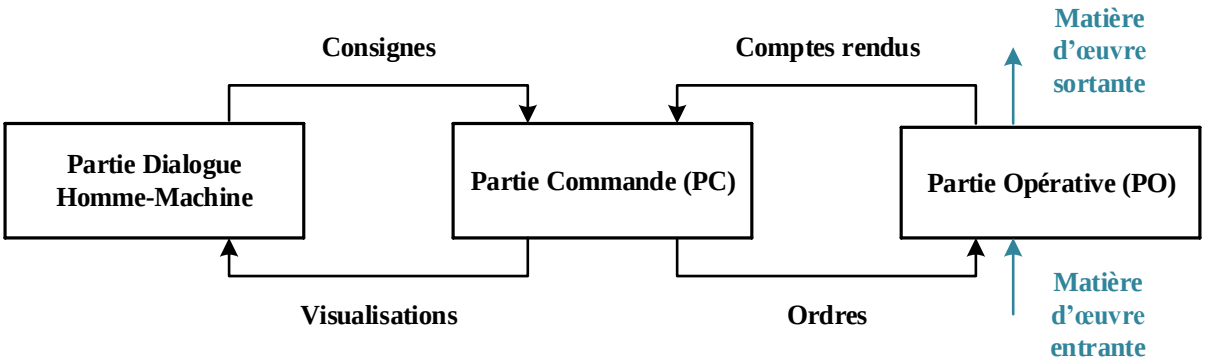


Figure 5 : Structure d'un système automatisé de production

2.4.3 Problématique liée aux systèmes automatisés de production

Avec la complexité croissante et l'automatisation de plus en plus poussée pour atteindre les performances souhaitées, les systèmes de production deviennent plus vulnérables aux erreurs pouvant entraîner des comportements indésirables et endommager les systèmes. Par conséquent, la gestion des défauts joue un rôle important dans les systèmes de fabrication actuels où de nombreux composants automatisés interagissent de manière complexe, de sorte qu'un défaut sur un seul composant peut entraîner un dysfonctionnement du système dans son ensemble. Il ne convient plus seulement de détecter et de localiser une défaillance dans un système manufacturier, mais également de fournir une solution permettant à l'opérateur d'interagir avec son système.

Afin de répondre aux problèmes de sécurité opérationnelle dans le domaine du contrôle des systèmes, la mise en œuvre de méthodes formelles est nécessaire. Dans ce contexte, il est important de surveiller le système et de proposer une solution alternative pour maintenir la production et assurer le fonctionnement des systèmes. Cette problématique fait d'ailleurs l'objet de plusieurs travaux de recherche ces dernières années.

3 Sûreté de fonctionnement

3.1 Définition

La sûreté de fonctionnement ou « la science des défaillances » est une notion générique qui mesure la qualité de service délivré par un système (Avižienis et al. 2004). Elle inclut l'évaluation des risques potentiels, la prévision de l'occurrence des défaillances et la minimisation des conséquences des situations catastrophiques lorsqu'elles se présentent, tout en permettant aux utilisateurs de placer une confiance justifiée dans le service délivré par le système. Pour atteindre les objectifs visés par la sûreté de fonctionnement, on distingue deux niveaux (Cauffriez 2005):

Niveau 1 : Il est lié à la sûreté de fonctionnement *prédictive* et consiste à évaluer au préalable les objectifs de la sûreté. Ceci correspond à une démarche formelle, technique, objective et quantitative qui a pour but de définir les différentes situations possibles et d'évaluer la probabilité de leurs occurrences. Face à une situation déterminée, il est nécessaire de définir les différents chemins conduisant à un événement interdit, les modes d'isolement des défaillances, le calcul de la durée d'occurrence et sa probabilité, l'identification des procédures de sécurité et les conséquences vis-à-vis du matériel, de l'humain et de l'environnement.

Niveau 2 : Il est lié à la *gestion* des objectifs de la sûreté, il s'agit d'un niveau d'ordre qualitatif, subjectif, et social. Les prises de décisions sont réalisées par rapport à une fonction de coût qui peut être : risque/coût, gain/perte ou risque/coût/ bénéfice.

3.2 Les fondamentaux de la sûreté de fonctionnement (attributs FMDS)

Dans les systèmes automatisés de production ou les systèmes manufacturiers, les préoccupations sont plutôt liées à la **disponibilité**. Dès lors que la **sécurité** ou la disponibilité d'un système est mise en défaut, c'est la **fiabilité** qui est incriminée. Enfin, en cas de dysfonctionnement, il convient de remettre le système dans les conditions de fonctionnement initial, c'est à ce niveau qu'intervient la **maintenabilité**. Ces quatre paramètres Fiabilité – Maitenabilité – Disponibilité – Sécurité (**FMDS)** constituent les bases de la sûreté de fonctionnement d'un système automatisé de production (figure 6). Ils permettent de définir les objectifs attendus d'un système et/ou d'évaluer la qualité du service délivré par le système afin de cibler les points critiques à améliorer (Clarhaut 2009b). La norme européenne NF EN 13306 précise leurs définitions.

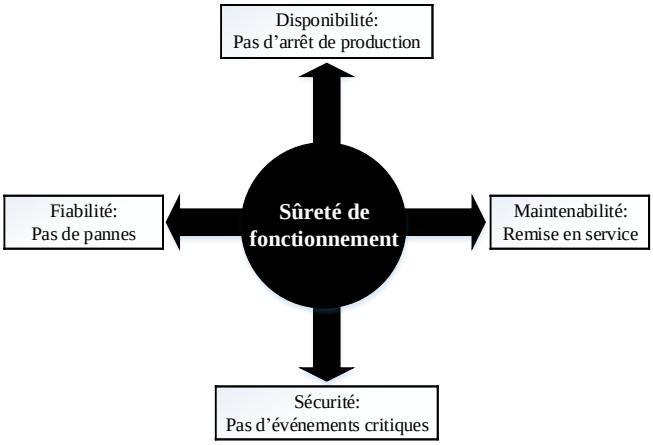


Figure 6 : Attributs de la sûreté de fonctionnement (inspirée de (Moïsio 2016))

3.2.1 Fiabilité (Reliability)

La fiabilité R(t) correspond à l'aptitude d'un bien à accomplir dans des conditions données, une fonction requise durant un intervalle de temps déterminé. Selon la norme, un bien signifie au sens large tout composant, élément, sous-système, unité fonctionnelle, système ou équipement qui peut être considéré individuellement. La fonction requise est la ou les fonctions qu'un bien doit accomplir pour remplir pleinement la tâche qui lui est associée.

Soit « T » l'instant d'occurrence de la défaillance, cette variable aléatoire permet de définir la notion de fiabilité qui est interprétée comme étant la probabilité que l'entité « E » considérée tombe en panne après un instant t donné ou bien comme la probabilité qu'elle ne tombe pas en panne avant l'instant t. La fiabilité (Procaccia et al. 2011) est la probabilité définie par : R (t) = P (T > t) (l'entité E est non défaillante sur la durée [0, t]).

3.2.2 Disponibilité (Availability)

Selon la norme, la disponibilité est l'aptitude d'être dans un état qui permet d'accomplir une fonction requise dans des conditions données, à un instant donné, en supposant que la fourniture des moyens extérieurs nécessaires soit assurée.

La disponibilité est la probabilité A(t) qu'une entité soit en état d'accomplir une fonction requise dans des conditions données à l'instant « t ». Cette probabilité ne fait pas appel à l'histoire de l'entité E, qu'elle ait été réparée ou non avant l'instant « t »: A(t)= P (l'entité E est non défaillante à l'instant 't').

3.2.3 Maintenabilité (Maintainability)

Dans des conditions données d'utilisation, la maintenabilité est définie par l'aptitude (la probabilité) d'une entité à être maintenue ou remise en service sur un intervalle donné de temps, dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données avec des procédures et des moyens prescrits.

La maintenabilité d'une entité E qui peut être réparée est caractérisée par une probabilité M(t) que la maintenance d'une entité accomplie dans des conditions données, avec des démarches et des moyens définis, soit achevée au temps t, sachant que l'entité E est défaillante à t = 0. La maintenabilité est alors une équivalence de la fiabilité, mais elle est appliquée à la réparation est non pas à la défaillance :

Tableau 1: Les trois items de la maintenance

	Méthodes de maintenance	**Evénements déclencheur**	**Actions de maintenance**
Maintenance	**Corrective**	Défaillance	Dépannage, réparation
	Préventive, Prédictive et Prescriptive	Date/ échéance, franchissement limite ou seuil, dérives	Remplacement systématique, remplacement sous condition, intervention ciblée

- M(t) = P (la maintenance de l'entité E est achevée au temps t)
- M(t) = 1 - P (l'entité E est non réparée durant [0, t]).

Les différentes méthodes de maintenance et les actions liées à la maintenance sont présentés dans le tableau 1 et explicités à la suite (Ribot 2009).

Les méthodes de maintenance :

- La maintenance corrective (Grusenmeyer 2000) a pour objectif de rétablir le système après l'apparition d'une défaillance de manière à ce qu'il soit capable de fournir à nouveau ses fonctions.
- La maintenance préventive (Thomas et al. 2009) a pour objectif de prévenir les défaillances durant le fonctionnement du système. L'aspect préventif est important pour des raisons de sûreté de fonctionnement mais aussi pour des raisons économiques. La maintenance préventive a pour but de supprimer les causes d'accidents graves en réduisant la probabilité des défaillances d'un système ou la dégradation des équipements de ce dernier.
- La maintenance prédictive ou prévisionnelle (Thomas 2009) est une maintenance conditionnelle basée sur l'anticipation du franchissement d'un seuil prédéfini qui permet de donner l'état de dégradation du bien avant sa détérioration complète. Le but de cette maintenance est d'agir sur l'élément défaillant au plus près de sa période de dysfonctionnement. Elle permet aussi de suivre une dégradation dans le cas d'une durée de vie variable d'un élément. Toutes ces actions permettent donc de réduire la fréquence des pannes tout en optimisant la fréquence des interventions préventives.
- La maintenance prescriptive (Iung 2019) intègre la prise de la décision et englobe à la fois les événements susceptibles de survenir et la mise en œuvre de stratégies de maintenance optimisées. Ce type de maintenance utilise aussi des analyses avancées sur les données pour faire des prédictions mais la différence est qu'elle ne propose pas seulement des recommandations, mais propose des actions à faire sur la base de ces recommandations. Ainsi, la différence entre l'analyse prédictive et l'analyse prescriptive est le résultat de l'analyse. La première fournit des éléments pour prendre des décisions éclairées, tandis que la seconde offre des options de décision étayées par des données qu'il est possible ensuite de comparer les unes aux autres.

Les actions de maintenance : Parmi ces opérations[1], on peut distinguer la réparation, le dépannage, et l'inspection, ...

- La réparation correspond à une action définitive de maintenance corrective après la détection d'une panne ou une défaillance. Son application est assurée immédiatement à la suite d'une défaillance, après un dépannage, ou après une visite de maintenance préventive,
- Le dépannage correspond à une action sur un équipement en panne pour le remettre en état de fonctionnement. Cette action est exploitée par les services de maintenance pour abaisser le coût de leurs dépenses,
- L'inspection est une activité de surveillance portant à relever d'une manière périodique les anomalies sans avoir besoin d'un outillage spécifique, ou d'un arrêt de l'outil de production ou des équipements.

[1] http://tpmattitude.fr/methodes.html

3.2.4 Sécurité (Safety)

La sécurité est l'aptitude d'une entité à ne pas connaître de pannes considérées comme catastrophiques ou critiques pendant une durée donnée (Tuptuk et Hailes 2018). Pour élaborer la sécurité au sein des SAP, il faut (Durka et Brun 1993) tout d'abord élaborer le référentiel d'évaluation pour ensuite effectuer l'évaluation fonctionnelle et l'évaluation des actions de l'opérateur :

Référentiel d'évaluation : Il est nécessaire de construire un référentiel d'évaluation avant d'activer un système. Le référentiel se compose de la définition du système et de ses limitations : environnement, caractéristiques de fonctionnement, condition d'utilisation, fonctions à réaliser, actions assurées par l'opérateur, cycle de vie, limite du système étudié. Il contient également l'analyse préliminaire des risques réalisée selon les éléments du dossier technique fourni par le constructeur, en tenant compte des agressions possibles de l'environnement. L'analyse des risques permet de définir les principales fonctions associées aux contraintes de sécurité, ainsi que les évènements redoutés tels que la génération d'un ordre inapproprié et la non-génération d'une alerte.

Evaluation fonctionnelle : Avant d'étudier les caractéristiques de sécurité d'un système, il est nécessaire de s'assurer que ce dernier satisfait certains critères grâce à des évaluations basées sur des tests/validations. Ces tests visent d'une part à valider les fonctions de l'équipement par rapport aux performances annoncées dans la documentation technique pour assurer une conformité fonctionnelle. D'autre part, ils visent à valider le bon fonctionnement du matériel vis-à-vis des contraintes d'environnement auxquelles ce dernier est susceptible d'être confronté telles que : la température, l'humidité, la poussière, les vibrations, la variation de la tension de l'alimentation, etc .

Evaluation des actions de l'opérateur : Lorsqu'un système industriel devient complexe, l'intervention d'un opérateur de maintenance ou de conduite peut menacer les dispositifs de sécurité mis en place. En effet, il n'existe pas de technique ou de méthode pour être sûr qu'un opérateur ne contribuera pas à une erreur. Pour limiter ce risque, des précautions et des interventions doivent être prises en compte telles que : la vérification des qualités des tests, des documentations techniques, la qualité des signalisations, etc.

3.2.5 Discussion

La fiabilité, la disponibilité, la maintenabilité et la sécurité, piliers de la sûreté de fonctionnement, sont dépendants entre eux (figure 7) de la manière suivante (Clarhaut 2009b) :

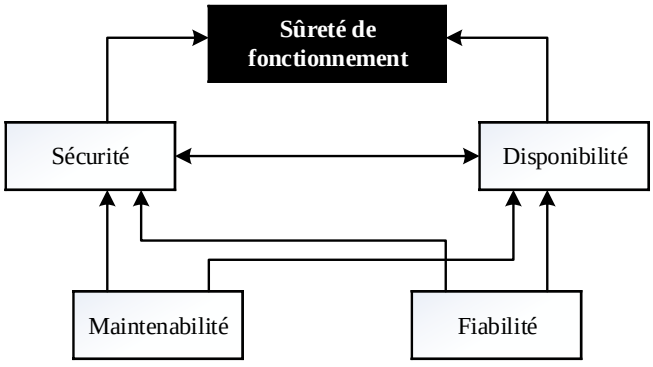


Figure 7 : Les interdépendances entre les fondamentaux de la sûreté de fonctionnement (inspirée de (Clarhaut 2009b))

- La diminution de la fiabilité du système entraîne une faible disponibilité due à la présence de nombreuses défaillances. Cette diminution a également un impact sur le niveau de sécurité du système, en effet, une défaillance peut conduire le système à un état dangereux,
- Pour des systèmes réparables, une maintenabilité inexacte peut affecter la disponibilité du système à cause de l'augmentation du nombre de défaillances et la sécurité du système à cause de l'augmentation du risque d'accident,
- En ajoutant plusieurs éléments de sécurité, le niveau de la sécurité augmente et le système arrête le fonctionnement dès la première défaillance détectée. Ce qui entraine une réduction de la disponibilité du système.

4 Entraves à la sûreté de fonctionnement

Une entrave est l'événement qui peut affecter la sûreté de fonctionnement du système. Les entraves à la sûreté de fonctionnement sont représentées par la figure 8 :

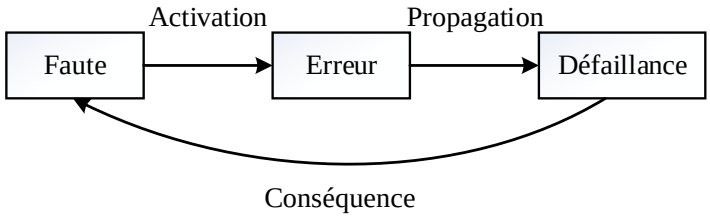


Figure 8 : Enchaînement des entraves à la sûreté de fonctionnement (inspirée de (Besseron 2010))

4.1 Faute

Une faute est considérée comme étant la cause (événement, action, circonstance) interne provoquant une défaillance. Elle peut être introduite par l'environnement, l'opérateur ou le concepteur. C'est un événement naturel, inévitable considéré dans le cadre de nos travaux de thèse comme tolérable.

4.2 Erreur

Une erreur est un état ou partie de l'état d'un système susceptible de conduire à une défaillance ou pas selon qu'il existe une redondance et selon sa nature car la redondance permet la continuité du service et selon l'activité du système, car la partie erronée peut ne pas être utilisée ou peut être éliminée avant la défaillance. Une erreur est dite *latente* tant qu'elle n'a pas conduit à une défaillance. Le temps entre l'apparition d'une erreur et une défaillance est appelé *délai de latence*. Plus ce dernier est long, plus la détermination des causes de la défaillance est difficile.

4.3 Défaillance

Une défaillance selon la norme NF EN 13306 est l'altération ou la cessation de l'aptitude d'une entité à accomplir une fonction requise avec les performances définies dans les contraintes et spécifications techniques. L'entité est alors indisponible suite à la défaillance, ce qui la conduit à être dans un état appelé *panne*. On considère un système Z constitué de deux

composants X et Y en cascade, les trois entraves à la sureté de fonctionnement sont présentées par la figure 9.

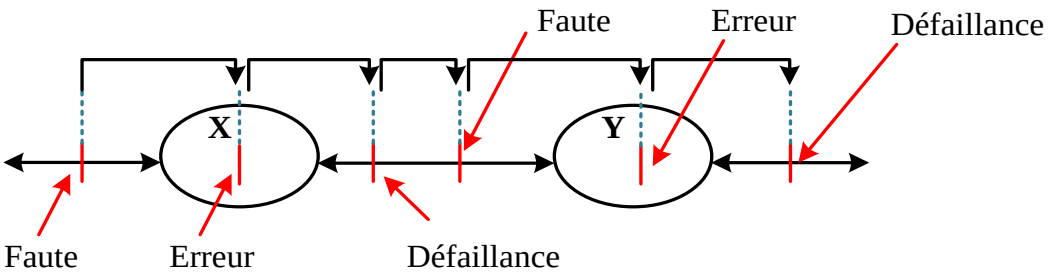


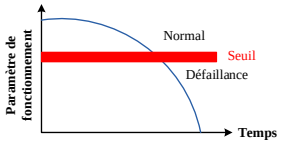
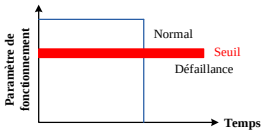
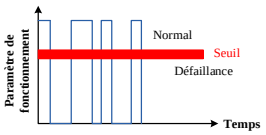
Figure 9 : Enchaînement des entraves de la sûreté de fonctionnement système Z

Le bon fonctionnement de Y dépend du bon fonctionnement de X, une défaillance de X entraîne l'apparition d'une faute pour Y et une faute de Y peut, à son tour, provoquer une erreur interne de Y, entrainant ensuite une défaillance.

4.4 Classification des défaillances

Selon la norme NF X 06-501, les défaillances n'ont pas un comportement identique dans le temps. Elles peuvent être classifiées selon plusieurs caractéristiques. Le tableau 2 rassemble les sept critères pour classifier une défaillance.

Tableau 2 : Critères de classification des défaillances selon la norme NF X 06-501

Critère de classification	Types de défaillance	
Vitesse d'apparition	**Défaillance progressive** : Elle est due à une évolution dans le temps de certaines caractéristiques d'un système. Elle aurait pu être prévue par un examen ou une surveillance.	
	Défaillance soudaine : elle est brutale et due à une évolution quasi instantanée des caractéristiques d'une entité. Elle n'aurait pas pu être prévue par un examen ou une surveillance.	
	Défaillance aléatoire : C'est une défaillance dont l'instant exact d'apparition n'est pas prévisible.	

Degré d'importance	**Défaillance partielle** : Défaillance qui entraîne l'inaptitude d'une entité à accomplir certaines fonctions requises. Elle résulte d'une déviation d'une ou de plusieurs caractéristiques au-delà des limites spécifiées, mais de telle façon qu'elle n'entraine pas une disparition totale de la fonction requise. **Défaillance complète** : Elle provoque l'inaptitude totale de l'entité à accomplir toutes les fonctions requises. **Défaillance intermittente** : C'est la défaillance d'un dispositif pour une période de temps limité, après laquelle celui-ci retrouve son aptitude à accomplir la fonction requise sans avoir été soumis à une action corrective extérieure. Ce type de défaillance est souvent répétitif.
Instant d'apparition	**Défaillance en fonctionnement** : C'est une défaillance qui se produit sur l'entité, alors que la fonction requise est utilisée. **Défaillance à l'arrêt** : Une défaillance qui se produit sur l'entité, alors que la fonction requise n'est pas utilisée. **Défaillance à la sollicitation** : Une défaillance qui apparaît au moment où la fonction requise est sollicitée.
Cause	**Défaillance due à une faiblesse inhérente** : Elle est attribuée à une faiblesse inhérente à l'entité elle-même lorsque les spécifications ne dépassent pas les niveaux prévus lors de la conception. Elle peut être due à une conception inappropriée ou à une fabrication non conforme à la conception de l'entité ou aux procédés de fabrication spécifiés. C'est à dire faiblesse due à la conception ou à la réalisation de l'entité. **Défaillance par un mauvais emploi** : C'est une défaillance due à l'application des contraintes en utilisation qui dépassent les possibilités spécifiées de l'entité. **Défaillance primaire** : C'est la défaillance d'une entité dont la cause directe ou indirecte n'est pas une défaillance d'une autre entité. **Défaillance secondaire** : C'est la défaillance d'une entité dont la cause directe ou indirecte est due à une défaillance d'une autre entité. **Défaillance par fausse manœuvre** : C'est la défaillance d'une entité causée par une opération incorrecte dans son utilisation.
Conséquences	**Défaillance critique** : Elle est considérée comme susceptible de causer des blessures corporels et matériels importants ou de conduire à d'autres conséquences jugées inacceptables. **Défaillance majeure** : Elle est susceptible d'affecter de façon importante une fonction considérée comme d'importance majeure **Défaillance mineure** : Elle n'est pas susceptible d'affecter de façon importante une fonction considérée comme d'importance majeure.
Origines	**Défaillance externe** : C'est une défaillance dont l'origine est considérée et attribuée à des facteurs externes à l'entité. **Défaillance interne** : C'est une défaillance dont l'origine est affectée à l'entité elle-même.

5 Etude d'un système sûr de fonctionnement

La sûreté de fonctionnement représente un « outil » indispensable à la conception d'un système industriel sûr. Elle aide à la prise de décision (vérifier et analyser d'une manière rapide les informations afin de pouvoir prendre la décision la plus adaptée à un instant donné) grâce à

l'identification des risques, l'optimisation de l'architecture et les moyens de support, la justification et la démonstration des choix adoptés et enfin la vérification des objectifs. La sûreté de fonctionnement est considérée aussi comme un outil d'optimisation qui aide à diminuer les pannes observées durant le cycle de vie d'un système, à rendre la maintenance plus efficace et ciblée, à dimensionner les moyens nécessaires pour le support, et enfin à optimiser les coûts de conception.

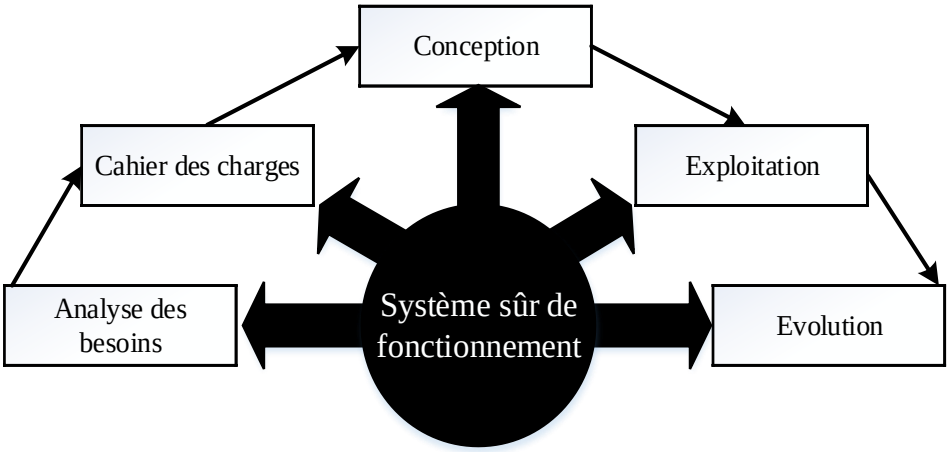


Figure 10 : Etude d'un système sûr de fonctionnement [2]

La figure 10 illustre les cinq étapes essentielles pour étudier un système sûr de fonctionnement. Avant toute étude, il faut débuter par l'étape d'*analyse des besoins*. Cette première étape se base sur l'analyse et la compréhension de la situation et du besoin pour identifier et prendre en compte les contraintes et les risques pour assurer un processus répondant aux besoins du client. À ce niveau, les actions suivantes doivent être accomplies :

- Connaître le contexte, c'est-à-dire, comprendre les besoins exprimés par le client. La source principale de l'information étant le client lui-même, des informations supplémentaires peuvent être aussi utiles en documentation
- Déterminer les besoins et les contraintes en les organisant par degré d'importance afin de mieux cerner à quelle étape de la conception, ces besoins doivent être pris en compte.
- Déterminer les paramètres de conception à partir des mesures prises sur le site du processus.

Cette connaissance permet, dans une deuxième étape, la rédaction d'un *cahier des charges* du système étudié. Il décrit d'une manière claire et précise les tâches et les opérations à effectuer et aussi une évaluation du temps pour les accomplir. Il comporte en général les éléments suivants :

- Mise en situation
- Etat des lieux
- Description détaillée des travaux
- Contraintes techniques particulières
- Clauses administratives
- Mandat
- Description générale du projet
- Spécifications techniques
- Liste des biens livrables

Après l'écriture du cahier des charges du système, il faut passer à la phase de *conception* en se basant sur les solutions techniques et les architectures proposées dans le cahier des

[2] https://sitelec.org/

charges. Dans cette phase de conception, les solutions doivent être quantifiées d'un point de vue sûreté de fonctionnement, comparées entre elles et, si nécessaire, optimisées.

Une fois les solutions retenues déterminées, l'identification des conditions de leur *exploitation* devient nécessaire. Il faut spécifier les opérations de maintenance préventive nécessaires pour conserver les caractéristiques souhaitées pour la sûreté de fonctionnement sans aucune dégradation. Cela permettra, par la suite, une *évolution* sûre de fonctionnement en ajustant la politique de maintenance adoptée, si nécessaire.

La conception d'un système sûr de fonctionnement s'effectue grâce à une analyse fonctionnelle et une analyse dysfonctionnelle du système (figure 11).

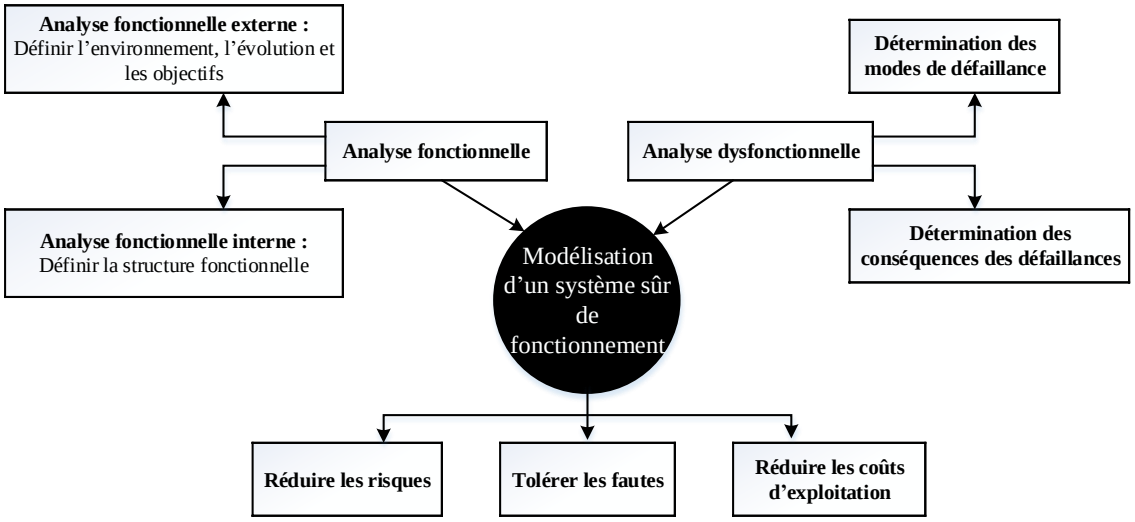


Figure 11 : Analyse fonctionnelle et dysfonctionnelle pour modéliser un système automatisé de production sûr de fonctionnement[3]

5.1 Analyse fonctionnelle

L'analyse fonctionnelle (figure 11) consiste à rechercher et à caractériser les fonctions d'un produit placé dans un système pour satisfaire les besoins de son utilisateur (Fougères 2005).

Lorsque l'analyse fonctionnelle ne s'intéresse qu'aux fonctions relatives à l'usage d'un produit pour satisfaire le besoin du client, alors le produit est considéré comme une boîte noire et seules les fonctions sortantes de cette boîte sont concernées. On parle d'une *analyse fonctionnelle externe* qui sert à exprimer le point de vue du client et ne met en évidence que les fonctions de service.

Lorsque l'analyse fonctionnelle s'intéresse au produit lui-même dans un objectif d'amélioration de son comportement, de sa fiabilité, ou de la diminution de son coût, le produit n'est plus vu comme une boîte noire. Il est alors considéré comme un ensemble de composants interagissant entre eux. On parle d'une *analyse fonctionnelle interne* ou analyse fonctionnelle technique qui met en évidence les fonctions techniques du système. Elle permet ainsi la réalisation d'une transition entre l'analyse fonctionnelle du besoin et la conception détaillée en tenant compte des considérations technologiques.

[3] http://star-engineering.fr/surete-de-fonctionnement-maitrise-des-risques/

5.2 Analyse dysfonctionnelle

L'analyse dysfonctionnelle (figure 11) consiste à déterminer les modes de défaillances pouvant entraver le fonctionnement du système et à étudier les conséquences de chaque mode de défaillance des fonctions décrites dans l'analyse fonctionnelle (Demri 2009). Par mode de défaillance, on entend :

- Absence de fonction à la demande : le système ne fonctionne pas,
- Perte de fonction durant le fonctionnement : le système ne fonctionne plus,
- Fonctionnement dégradé : le système ne fonctionne pas de manière normale,
- Fonctionnement intempestif : le système fonctionne sans le demander,
- Fonctionnement intermittent : le système s'arrête et se remet en fonctionnement tout seul.

Un système automatisé de production peut évoluer selon deux modes principaux de fonctionnement : un mode normal et un mode anormal (Rayhane 2004). Selon le mode de fonctionnement dans lequel le système évolue, l'objectif pour lequel le système a été conçu peut-être atteint totalement, partiellement ou non atteint. Partant de ces deux modes de fonctionnement, plusieurs modes doivent être différentiés (figure 12) :

- Mode de fonctionnement normal : Il correspond au fonctionnement nominal du système tout en répondant à l'objectif pour lequel il a été conçu.
- Mode de fonctionnement anormal : Il correspond au mode dans lequel le système ne peut pas atteindre d'une manière totale ou partielle ses objectifs. On distingue quatre modes de fonctionnement anormaux :
    - Mode défaillant : Il correspond à un mauvais fonctionnement du système suite à l'apparition d'une défaillance.
    - Mode dégradé : Il correspond à l'accomplissement de l'objectif du système d'une manière partielle ou avec des performances moindres.
    - Mode critique : Il correspond au mode dans lequel le système, suite à la détection des symptômes critiques, présente un fonctionnement particulier et non souhaité.
    - Mode interdit : Il correspond au mode dans lequel le système, suite à des critères de sécurité, ne doit pas fonctionner.

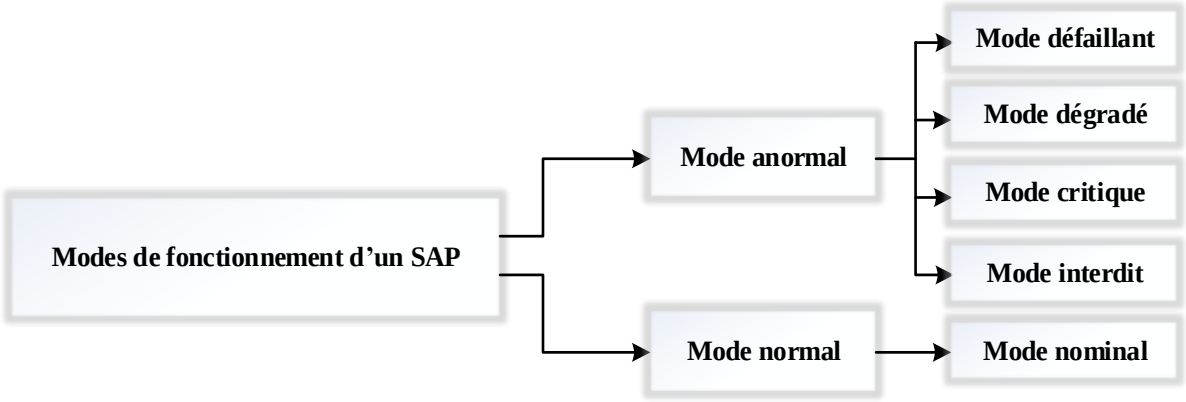


Figure 12 : Modes de fonctionnement d'un système automatisé de production

En tenant compte des différents types de défaillances cités ci-dessus, un mode de défaillance est défini pour chaque fonction du système. Il devient alors possible de confirmer ou non une solution technique, optimiser les choix des architectures, et remplacer les éléments

défaillants dans l'objectif de tolérer, dans la limite du possible, certaines fautes en autorisant un mode de fonctionnement dégradé conditionné.

Dans la cadre de cette thèse, nous nous intéressons aux fonctionnements normal, défaillant et dégradé du système comme le montre la figure 13. Après la détermination de la défaillance et le passage du système à un mode de fonctionnement défaillant, un scénario de reconfiguration peut être déclenché avant la réparation de l'élément défectueux. Le système passe alors en mode de fonctionnement dégradé équivalent au fonctionnement normal. Cette équivalence sera assurée par des informations temporisées pour pallier les éléments défaillants, la démarche est détaillée dans le chapitre 3.

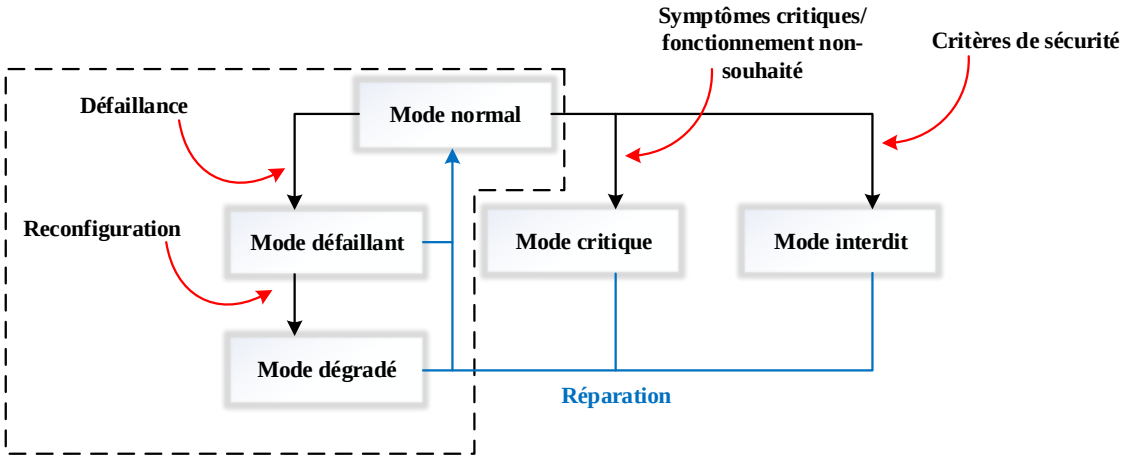


Figure 13 : Modes de fonctionnement d'un système automatisé de production en prenant en compte la reconfiguration

5.3 Outils utilisés pour la sûreté de fonctionnement

L'analyse et l'évaluation d'un système pour la sûreté de fonctionnement peuvent être réalisées à l'aide de nombreux outils, on distingue deux catégories d'outils : ceux utilisés pour l'analyse fonctionnelle et ceux exploités pour l'analyse dysfonctionnelle.

5.3.1 Outils utilisés pour l'analyse fonctionnelle pour la sûreté de fonctionnement d'un système automatisé de production

Les outils les plus utilisés par les industriels sont :

BDF (Blocs Diagrammes Fonctionnels) (Gonzva et al. 2016) : Il s'agit d'une méthode de découpage fonctionnel du système dans l'objectif de faciliter l'analyse en traitant chaque sous-système tout seul.

SADT (System Analysis and Design Technique) (Cauffriez et al. 2006) : C'est une méthode d'analyse par niveaux successifs d'approche descriptive d'un ensemble. Les fonctions (figure 14) sont représentées par des boites connectées entre eux à travers des liens symbolisant les différentes interactions. Les connexions peuvent être des entrées, des sorties, ou une contrainte.

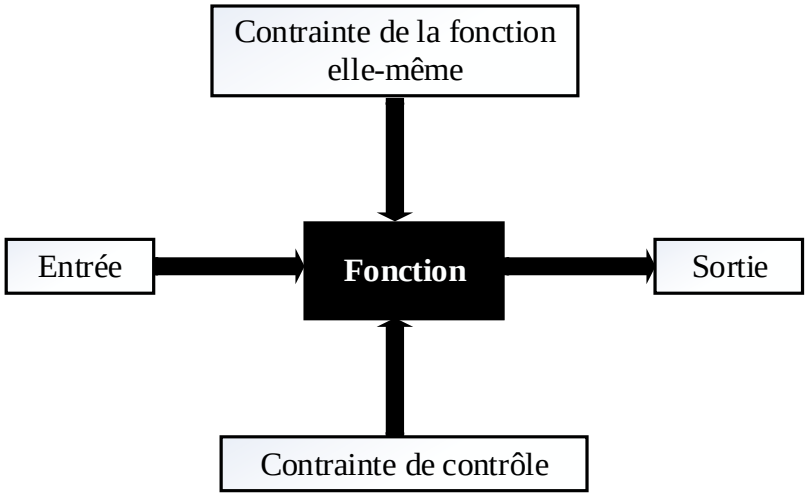


Figure 14 : Principe du SADT

MISME (Méthode d'Inventaire Systématique des Milieux Extérieurs ou Environnants) (Ziv 2018) : Cette méthode considère l'ensemble des composants du système avec leurs interactions, ainsi que les milieux environnants. Elle se base sur deux étapes : la validation du besoin en utilisant le diagramme de bête à cornes, et l'analyse du besoin grâce au diagramme pieuvre.

5.3.2 Outils utilisés pour l'analyse dysfonctionnelle de la sûreté de fonctionnement

APR (Analyse Préliminaire des Risques) (Guenab et al. 2008) : L'APR fournit l'ensemble des événements redoutés prévisionnels (fuites de matières dangereuses toxiques, explosion, incendie, affaissement de barrage, erreurs humaines, conditions climatiques extrêmes, séismes, pannes électriques, pandémie, …) dans tout le cycle de vie du système dès les premières étapes de la conception, en passant par la mise en service, et enfin l'exploitation et la maintenance. Elle consiste aussi à déterminer les causes et les conséquences d'une situation de danger et à mettre en lumière les barrières de sécurité existantes de prévention et/ou de protection (les mesures de traitement des risques) et proposer des améliorations si besoin.

AMDEC (Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité) (Bironneau, et al. 2010): Il s'agit d'une méthode exhaustive qui vérifie, à un niveau de détail choisi à l'avance, les potentialités des dysfonctionnements de chacun des éléments composant le système. Elle permet de quantifier la probabilité d'apparition des défaillances et de classer ses effets par ordre de gravité. C'est une méthode inductive qui permet de réaliser une analyse qualitative de la fiabilité d'un système depuis un niveau bas jusqu'à un niveau élevé. L'AMDEC joue un rôle important dans la procédure d'assurance de la fiabilité. Elle peut s'appliquer à un large intervalle de problèmes qui peuvent se produire dans un système industriel. Elle peut être plus ou moins approfondies ou modifiées selon l'objectif à achever. Cette méthode d'analyse est peu employée dans les phases d'étude, de planification et d'identification. Elle est, par contre, largement utilisée pendant la conception et la mise en œuvre. Il faut toutefois noter que l'AMDEC n'est qu'une phase du processus de fiabilité et de maintenabilité qui exige la réalisation de multiples tâches dans des domaines variés.

AEEL (Analyse des Effets des Erreurs Logicielles) (Hadj-Mabrouk 2007) : Cette analyse est une adaptation de l'AMDEC décrite ci-dessus. Le but de cette méthode est de mettre en évidence des points critiques relevés durant les phases de développement d'un logiciel, et de

proposer aux personnes chargées des tests de validation, une synthèse de la criticité des modules du logiciel analysé, afin d'affiner leur démarche. L'AEEL se déroule en trois phases. La *première* consiste à définir les travaux d'analyse qui évaluent les thèmes d'études par rapport aux spécifications et aux contraintes exprimées dans le cahier des charges, qui définissent les échelles de gravité vis-à-vis des thèmes, et qui déterminent les différents types d'erreurs. La *deuxième* phase se base sur un classement préparatoire des modules et fonctions du logiciel pour vérifier la couverture de la validation et des tests. La *troisième* phase se base sur la réalisation des AEEL : déterminer les effets au niveau local et au niveau global pour chaque erreur envisageable, classer les erreurs selon l'échelle de gravité précédemment fixée à la phase 1 et recommander des actions pour pouvoir détecter et éliminer les erreurs ou éviter leur propagation.

ADD (Arbre De Défaillance ou arbre de pannes ou arbre de fautes) (Clarhaut 2009a) : C'est une méthode qui présente l'ensemble des combinaisons des sous-événements qui peuvent amener le système à une défaillance dite événement de tête, soit par une occurrence simultanée de tous les sous-événements ou par une apparition d'un sous-événement quelconque. Ce dernier est ensuite décomposé de la même façon jusqu'à l'obtention des éléments suffisants pour évaluer la probabilité de l'apparition de ces sous-événements. Cet enchaînement est réalisé grâce à un schéma logique de l'arbre de décomposition et ensuite la probabilité de l'occurrence de l'événement de tête ou de défaillance est déduite. Un ADD est généralement présenté de haut en bas (figure 15). La première ligne au sommet de l'arbre, contient uniquement la défaillance que l'on cherche à analyser. Chaque ligne détaille la ligne supérieure en présentant la combinaison ou les combinaisons susceptibles de produire l'événement de la ligne supérieure auquel elles sont rattachées. Ces relations sont présentées par des portes logiques « OU » ou « ET ».

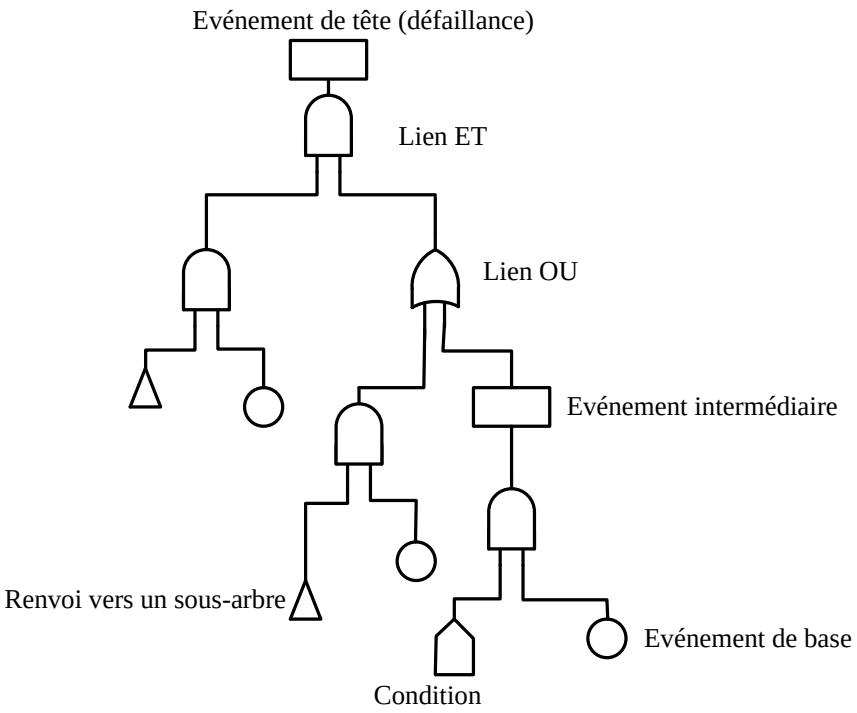


Figure 15 : Exemple d'un ADD[4]

[4] https://www.onera.fr/

L’avantage principal de l’utilisation de la méthode de l’ADD est la possibilité de visualiser l'ensemble des combinaisons d'événements conduisant à une défaillance, c'est-à-dire qu'elle permet d'avoir une vision globale et logique du fonctionnement et des dysfonctionnements d'un système industriel.

Il faut savoir distinguer entre les méthodes d’ADD, l’Arbre de Causes (AC) et l’Arbre d’Evénements (AE) (Mortureux 2016). Certes, la représentation logique des trois méthodes est commune mais le besoin de réponse de chaque méthode est différent. Le tableau 3 représente les différences entre les trois méthodes.

Tableau 3 : Différences entre les méthodes d'ADD, d'AC, et d'AE

Méthode	**ADD**	**AC**	**AE**
Définition	Détermination de toutes les combinaisons de causes	Organisation d’un enchaînement particulier	Organisation de tous les enchaînements des conséquences
Type de démarche	Menée a priori	Menée par un retour d’expérience	Menée a priori ou par un retour d’expérience.
Quantification des probabilités d’occurrence	Oui	Non	Oui

La figure 16 montre les différences à retenir aussi du point de vue des représentations.

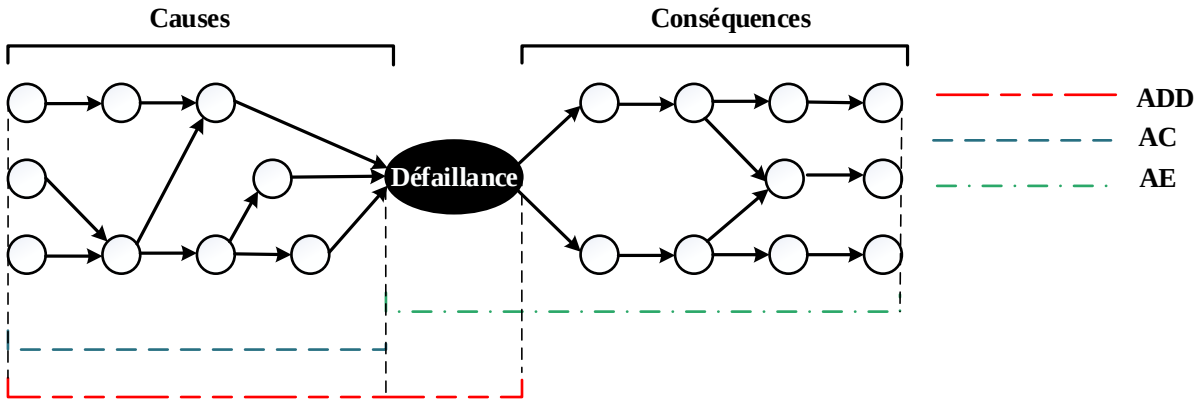


Figure 16 : Représentation des méthodes d'ADD, AC, et AE

GM (Graphe de Markov) (Quagliaro 1993) : Bien que la méthode d’ADD permette une description statique du système, elle ne prend pas en compte sa reconfiguration (réparation). La méthode du GM est développée pour des systèmes réparables ou reconfigurables dont l’objectif principal est d’analyser les caractéristiques de la fiabilité et de la disponibilité de ces systèmes. Trois états sont distingués : l’état de fonctionnement normal dit nominal, l’état de fonctionnement dégradé et l’état de panne. Ces états sont alors modélisés par des cercles reliés par des flèches (transitions), chaque transition est conditionnée par des défaillances et des remises en état nominal dont le taux de défaillance ou de reconfiguration est indiqué.

Le nombre d'états est défini par 2^n avec « n » est le nombre des éléments constituants un système. Un système est dit markovien si tous les taux associés aux transitions sont indépendants du temps. On considère un système constitué de deux éléments X et Y, le nombre d'états possible est 2^2=4 états : (1) X et Y fonctionnent, (2) X fonctionne et Y en panne, (3) X en panne et Y fonctionne, (4) X et Y en panne. Les événements conduisant à un état de panne sont p_X et p_Y, et les événements de réparation conduisant à l'état nominal sont r_X et r_Y. La figure 17 présente le GM du système.

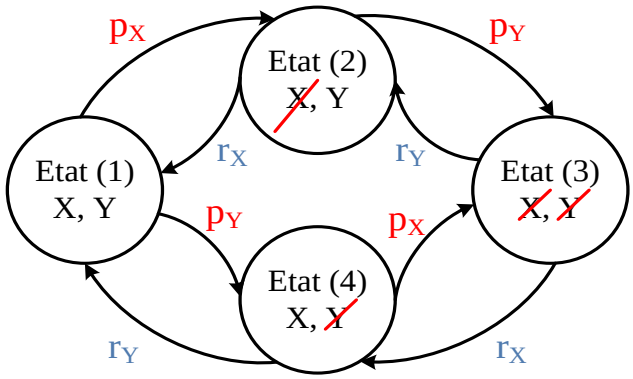


Figure 17 : Graphe de Markov d'un système constitué de 2 éléments

6 Moyens conduisant à la sûreté de fonctionnement

Les moyens conduisant à la sûreté de fonctionnement sont des solutions éprouvées pour casser les enchaînements dans la trilogie faute, erreur et défaillance :

6.1 Prévention et élimination des fautes

La prévention des fautes consiste à empêcher l'apparition des fautes et de diminuer la probabilité de leur occurrence (Potiron et al. 2013). Les principales actions de prévention consistent à :

- Utiliser les composants les plus fiables en respectant les coûts et les spécifications de performance,
- Choisir les meilleures techniques d'assemblage et d'interconnexion des composants,
- Utiliser des langages de programmation qui présentent une abstraction des données, la modularité et de la sécurité,
- Choisir un environnement de développement et une méthodologie structurée permettant la gestion de la complexité et la vérification que rien n'a été oublié,
- Examiner les formes d'interférences,
- Spécifier rigoureusement les besoins.

La prévention des fautes ne peut pas éliminer la possibilité d'occurrence de fautes. Il faut alors les détecter et les éliminer pour maintenir le bon fonctionnement du système.

L'élimination des fautes consiste à détecter, identifier et enlever les fautes du système. C'est une méthode basée sur une manipulation de tests la plus complète possible, dans des conditions de charge réalistes. Un test peut être utilisé uniquement pour démontrer la présence des fautes et pas leur absence complète et qui est parfois impossible en tenant compte des conditions réelles. C'est pourquoi les tests s'effectuent en simulation. L'élimination des fautes se réalise soit par élimination pendant la phase de développement soit par élimination pendant

la phase d'utilisation. Pendant la phase de développement, l'idée est d'utiliser des techniques de vérification avancées de façon à détecter les fautes et les enlever avant envoi à la production. Pendant l'utilisation, il faut tenir à jour les défaillances rencontrées et les retirer pendant les cycles de maintenance

6.2 Diagnostic

La faute est l'une des principales causes de défaillance d'un système de production. Un diagnostic précis est l'une des étapes les plus importantes en traitement de défaillance.

6.2.1 Définition

Le diagnostic (Alexandre Philippot 2006), (Nguyen 2015), (Blanke et al. 2016), est considéré comme étant une des fonctions qui participe à la réactivité d'un système vis-à-vis d'une occurrence de défaillances. Il est basé sur les trois étapes suivantes (Chen et Patton 1998) :

- La détection de faute : C'est une décision qui permet de savoir si un élément ne fonctionne pas bien ou que tout fonctionne dans les conditions normales.
- L'isolement de la faute : Il consiste à identifier l'emplacement de la panne (par exemple, le capteur ou l'actionneur est défectueux)
- L'identification de la faute : Elle consiste à déterminer la taille et le type ou la nature de la faute.

6.2.2 Classification des méthodes de diagnostic

Le diagnostic des fautes analyse les mesures d'un système donné et applique des algorithmes spécifiques pour fournir des informations sur les pannes du système (Korbicz et al. 2004). Diverses méthodes ont été proposées par la communauté scientifique et mises en œuvre dans les applications industrielles. Ces méthodes peuvent être classées en trois catégories (Papadopoulos et McDermid 2001) : (1) méthodes à base de systèmes experts, (2) méthodes fondées sur des données, et (3) approches basées sur des modèles.

- Les méthodes basées sur les systèmes experts utilisent les connaissances spécifiques de l'expert pour effectuer la tâche de diagnostic. Ce sont des méthodes à base de règles qui établissent des associations empiriques entre effets et causes. Pour cette raison, il est nécessaire de formaliser les connaissances pour rendre le système accessible aux algorithmes de diagnostic. Parmi les formalisations les plus utilisées, on trouve les règles « If-Then-Else » qui peuvent être évaluées à l'aide de l'enchaînement en avant ou en arrière des règles (Papadopoulos et McDermid 2001). Dans ce processus, une ou plusieurs règles sont déclenchées par une déviation d'un paramètre du système. Les règles déclenchées peuvent elles aussi impliquer le déclenchement d'autres règles dans une chaîne. Si aucune autre règle ne peut être déclenchée, les informations résultantes sont censées contenir les données nécessaires sur la faute.
- Les méthodes basées sur les données pour le diagnostic des fautes reposent sur l'analyse des données en provenance du système et sur l'extraction des fonctionnalités spéciales. La collection d'information (Dash et Venkatasubramanian 2000) est un exemple de cette catégorie de méthodes. Pour un ensemble de défauts donné, les collections attendues dans les mesures des capteurs doivent être stockées dans une base de données des connaissances.

Les auteurs proposent sept types de collection générales pour classer les différents défauts. Pendant le diagnostic en ligne, les mesures des capteurs sont analysées à l'aide des filtres appropriés pour identifier les collections d'informations importantes correspondant à un paramètre donné de primitives. Pendant ce processus, le temps de détection de défaut et la robustesse de la méthode doivent être équilibrés : la mesure analysée doit être suffisamment longue pour éviter une interprétation erronée du bruit « normal » en tant que collection, mais suffisamment courte pour détecter les collections réelles indiquant un défaut aussi rapidement que nécessaire.

- Les méthodes basées sur des modèles : L'idée de ces méthodes est de comparer le comportement du système observé avec le comportement attendu défini par un modèle de système (Sampath et al. 1995) (Sampath et al. 1996). Une politique de détection des fautes possible dans ce cadre est de détecter si la sortie du système observée et la sortie du système prévue diffèrent de manière importante. La précision du modèle, liée aux besoins de la surveillance et aux critères de performance du diagnostic, définit le choix de l'utilisation de modèles quantitatifs ou qualitatifs.

  Les méthodes quantitatives reviennent à comparer l'entrée et la sortie du système à diagnostiquer. Toutes incohérences exprimées en résidus, peuvent être utilisées à des fins de détection et de localisation (Nguyen 2015). La méthode quantitative permet la gestion des variables d'état d'un système donné pour diagnostiquer rapidement les fautes à travers les résidus. Cependant, plusieurs facteurs tels que la complexité du système, les non-linéarités et/ou le manque de données rend le développement d'un modèle mathématique du système difficile et parfois impossible. Ce type de méthode est limité pour les systèmes présentant une seule défaillance. En effet, si plusieurs défaillances apparaissent en même temps, elle produit le même résultat de diagnostic. En outre si un défaut n'est pas spécifiquement modélisé et a causé l'occurrence d'une nouvelle défaillance, le diagnostic ne fournit pas de résultat pour cette dernière.
  La méthode qualitative se repose sur la cohérence qui provient du domaine de l'intelligence artificielle. La technique du diagnostic de cohérence consiste à comparer le comportement réel du système observé et son comportement attendu tel qu'il peut être prédit grâce à des modèles de bon comportement (Nguyen 2015).

6.3 Tolérance aux fautes et reconfiguration

La tolérance aux fautes et la reconfiguration sont considérées comme des moyens d'augmentation du niveau de la sûreté de fonctionnement (Fayollas 2015) par rapport aux fautes qui peuvent entraver le bon fonctionnement d'un système. Elles consistent à mettre en place des mécanismes qui maintiennent le service fourni par le système, même en présence de fautes. Dans ce cas, le fonctionnement dégradé est accepté. Dans cette section, nous nous intéressons à la tolérance de fautes vue comme une classe de la reconfiguration.

6.3.1 Solutions utilisées par la tolérance aux fautes

Pour masquer une apparition de faute, la tolérance aux fautes utilise le principe de la redondance (Powell et al. 2011) qui peut être sous forme de :

- Redondance matérielle : Elle inclut les composants matériels ajoutés pour supporter la tolérance aux fautes à tous les niveaux (alimentation électrique, processeurs, disques, réseau...). Deux types de redondance matérielle sont définis :

    - La redondance statique pour laquelle des composants redondants sont utilisés pour cacher les effets des fautes.
    - La redondance dynamique utilisée à l'intérieur d'un composant indiquant si le résultat est fautif.
- Redondance logicielle : Elle contient tous les programmes et les instructions utilisés pour tolérer les fautes.
- Redondance temporelle : C'est le temps supplémentaire pour exécuter les tâches (exécuter les instructions plusieurs fois par exemple) pour la tolérance aux fautes.

6.3.2 Niveau de tolérance de fautes

Pour choisir un mécanisme de tolérance aux fautes adapté à un système donné, il est nécessaire de mettre en place des critères de sélection. Parmi ces derniers, on trouve le niveau de tolérance aux fautes (Durand 2011) :

- Tolérance aux fautes complète : Le système continue à fonctionner en présence des fautes, sans perte importante de fonctionnalité ou de performance.
- Dégradation graduelle : Le système continue à fonctionner en présence des fautes, acceptant une dégradation partielle des fonctionnalités ou de performance durant le recouvrement ou la réparation.
- Arrêt sécuritaire : Le système maintient son intégrité tout en acceptant un arrêt temporaire de son fonctionnement.

Le niveau de tolérance aux fautes nécessaire dépend de chaque système. La majorité des systèmes critiques nécessitent une tolérance complète, mais plusieurs se contentent d'une dégradation graduelle.

6.3.3 Phases de tolérance aux fautes

La tolérance aux fautes consiste à diminuer la probabilité de défaillance malgré la présence éventuelle de fautes. La tolérance aux fautes est mise en œuvre par la détection des fautes et le rétablissement du système ou recouvrement des fautes (Powell et al. 2011):

- Phase de détection des fautes : Une présence des fautes est déduite en détectant une erreur, elle se base généralement sur une fonction de diagnostic.
- Phase de recouvrement des fautes : Elle consiste à éliminer les erreurs de telle manière qu'elles ne se propagent pas par des actions futures. On discrimine :
    - Un recouvrement des fautes en aval (figure 18), c'est-à-dire continuer à partir de l'état fautif en faisant des corrections sélectives à l'état du système.

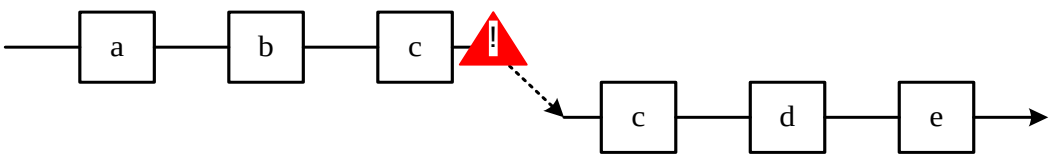


Figure 18 : Recouvrement en aval

    - Un recouvrement des fautes en amont (figure 19) qui consiste à restaurer le système à un état précédent sûr et à exécuter une section alternative.

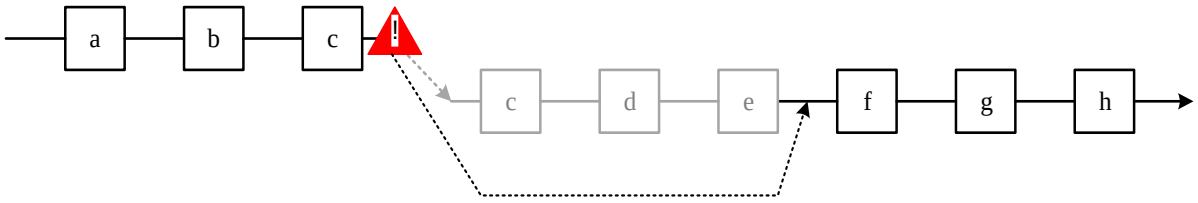


Figure 19 : Recouvrement en amont

6.3.4 Types de tolérance aux fautes

L'apparition des fautes a été souvent évitée en s'appuyant sur une redondance matérielle des actionneurs et des capteurs du système de production. Cette stratégie est non seulement coûteuse, mais elle nécessite également un dispositif de maintenance majeur. Dans ce contexte, et en réponse aux nouveaux défis posés par les coûts et la tolérance aux fautes, de nombreuses méthodes et techniques basées sur des modèles dynamiques ont été développées et analysées de manière analytique (Khelassi 2011). Ces méthodes sont généralement classées en deux grandes familles, tolérance aux fautes passive ou tolérance aux faute active. La distinction entre ces deux catégories dépend de la méthode de contrôle, des fautes considérées, du type de redondance présent et du comportement du système en cas de dégradation.

La tolérance aux fautes passive (Stefanovski 2018), (Badihi et Zhang 2017) permet l'utilisation d'un seul système de contrôle pour le fonctionnement normal et le fonctionnement fautif, elle est étroitement liée à un contrôle robuste. Même si l'idée des méthodes passives est attirante, ce type de solution peut imposer des limitations pour des systèmes de grandes dimensions.

La tolérance aux fautes active (Lan et Patton 2016), (Gao et al. 2017), (J. Wang et al. 2016) nécessite un système de contrôle qui adapte sa loi de contrôle à un événement fautif. Ce type de stratégie permet de concevoir un contrôleur pour le fonctionnement normal et un second contrôleur pour le fonctionnement fautif à l'aide d'un bloc de diagnostic. L'objectif principal est de passer d'un contrôleur à l'autre en fonction du mode de fonctionnement souhaité.

Une étude comparative entre les méthodes actives et passives est présentée dans (Jiang et Yu 2012) et (Blanke et al. 2016). L'étude indique que chaque approche a ses propres avantages et limites. Une approche de tolérance aux fautes active est plus flexible pour traiter différents types de défauts, y compris les scénarios de défaillance au-delà des défauts de conception. Cependant, elle dépend fortement du schéma de détection et diagnostic de fautes (DDF) pour fournir à temps les informations précises de défaut. Tout retard et/ou incertitude dans la DDF peut gravement affecter l'efficacité des approches actives. La complexité de la mise en œuvre est également relativement élevée. En revanche, les approches passives ne se basent ni sur des unités de DDF ni sur des mécanismes de reconfiguration, c'est relativement simple à implémenter. De plus, elles ne produisent aucune commutation transitoire, les actions de contrôle sont toujours engagées. Cependant, la tolérance aux fautes diminue à mesure que le nombre de scénarios de fautes augmente.

7 Conclusion

Nous avons présenté dans ce chapitre le contexte générale associée aux systèmes automatisés de production et la sûreté de fonctionnement. Nous avons précisé leur nature, leur

rôle et leur structure. Plongé dans un contexte lié à l'existence des aléas de fonctionnement nous avons montré la nécessité d'un système sûr de fonctionnement.

L'étude de ce type de systèmes, nous a permis, d'une part, de définir les entraves à la sûreté de fonctionnement et, d'autre part, de déterminer les moyens assurant la sûreté de fonctionnement. La *reconfiguration* est une notion associée généralement à des thématiques de contrôle/commande mais elle est peu utilisée dans le domaine d'analyse de la sûreté de fonctionnement. Un système est dit reconfigurable s'il dispose de mécanismes de commutation permettant la modification de sa structure et/ou son comportement.

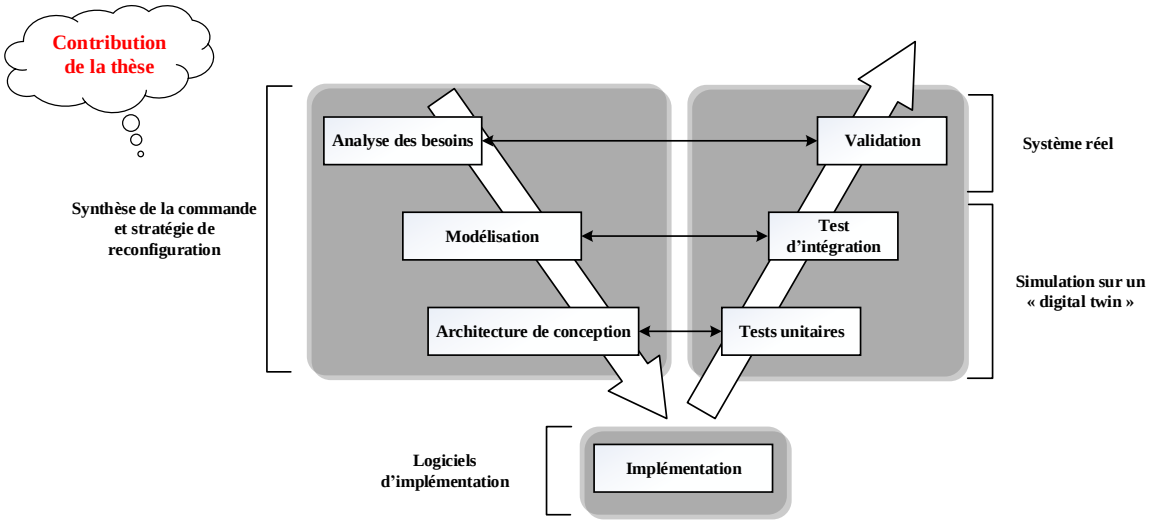


Figure 20 : Cycle en V de conception d'une commande sûre de fonctionnement

Dans le cadre de ma thèse, cette flexibilité de système est exploitée pour mettre en place une stratégie de reconfiguration permettant une sûreté de fonctionnement des systèmes automatisés de production dès la phase de conception de la commande (figure 20). Dans notre cas, cette stratégie est motivée par l'objectif de tolérance aux fautes.

Les phases de spécifications et de tests de la commande suivent la structure du cycle en V (modèle de validation et de vérification) qui permet de revenir aux étapes de conception si les tests effectués ne sont pas concluants comme le montre la figure 20. Les phases de conception et de vérification formelle sont effectuées grâce à l'approche présentée dans le chapitre 3 et la phase de simulation est effectuée sur le jumeau numérique avant la validation sur le système manufacturier réel. Avant d'entamer la contribution de la thèse au chapitre 3, le chapitre suivant détaillent la reconfiguration des Systèmes Automatisés de Production.

CHAPITRE II

Reconfiguration de la commande des SED

1 Introduction

Les entreprises manufacturières doivent faire face à une concurrence mondiale accrue (Koren et al. 1999) et à des clients de plus en plus exigeants. Pour survivre dans ce contexte, la demande de disponibilité, de fiabilité et de sûreté de fonctionnement des systèmes automatisés de productions (SAP) est devenue un enjeu majeur. Face aux limitations actuelles de ces systèmes, à l'évolution rapide des environnements et à la volonté de contrôler les différents paramètres et perturbations qui peuvent affecter leur bon fonctionnement, il s'avère nécessaire de mettre en œuvre des outils assurant une certaine tolérance aux fautes (Gharsallaoui 2010). Les systèmes automatisés de production reconfigurables (SAPR) semblent être une réponse à cette problématique et sont présentés comme un moyen prometteur pour les entreprises dans l'objectif de garantir un contrôle sûr de fonctionnement (Mehrabi et al. 2000). Ce chapitre a pour objectif de présenter les travaux de recherche existants sur les systèmes de contrôle reconfigurable dans un objectif de tolérance aux fautes.

2 La reconfiguration : principes et propriétés

Le terme *reconfiguration* signifie les modifications possibles pour une configuration. D'une manière générale, une configuration est l'environnement courant d'un objet. Au cours du cycle de vie d'un objet, l'environnement peut changer et l'objet peut donc être amené à s'adapter pour continuer ses actions (Lee 2006).

Pour les systèmes manufacturiers, une configuration d'un système est constituée de composants logiciels et de composants matériels. Des changements peuvent survenir alors au niveau logiciel et matériel et peuvent nécessiter une adaptation pour achever leurs tâches (Walden et al. 2015). On distingue deux types de configuration :

- La configuration *statique :* C'est une configuration du système qui, une fois définie, ne changera pas dans le temps ou, si elle doit être modifiée, le système doit être arrêté. Une configuration statique est souvent utilisée pour les petits systèmes qui n'ont qu'une seule tâche à exécuter.
- La configuration *dynamique* : Elle est utilisée pour des systèmes plus complexes. Ces dispositifs sont construits à partir de divers composants qui interagissent ensemble pour atteindre un objectif. L'application d'une configuration dynamique concerne souvent les systèmes distribués qui ne peuvent pas être arrêtés pour modifier le comportement de l'un de leurs composants. La réalisation d'un SAPR avec une configuration dynamique est plus difficile à obtenir pour plusieurs raisons :
    - ✓ La complexité de l'ensemble du système dépend de divers composants logiciels et matériels,
    - ✓ Les modifications doivent être appliquées pendant l'exécution du système,

- ✓ Puisque les modifications ont lieu pendant l'exécution, les composants du système prendront différents états pendant le fonctionnement ce qui implique que ces états devront être préservés pendant la reconfiguration,
- ✓ Les processus du système peuvent avoir des délais à respecter, ce qui impose une contrainte quant à la fonctionnalité temps réel du système.

2.1 Reconfiguration et temps réel

Pour les systèmes automatisés de production reconfigurables, le temps réel est l'un des facteurs clés de la fiabilité. Les SAPR peuvent être composés de divers sous-systèmes ou composants ce qui conduit à l'interrogation suivante : *Etant donné que les SAPR ont souvent des ressources limitées, comment est-il possible de reconfigurer de tels systèmes dans un temps limité ?* Concernant le temps réel, nous devons définir deux durées différentes :

- La *durée de reconfiguration* : C'est le temps nécessaire pour achever toute la reconfiguration d'un service, à partir du moment où la reconfiguration est déclenchée par ce dernier (figure 21).

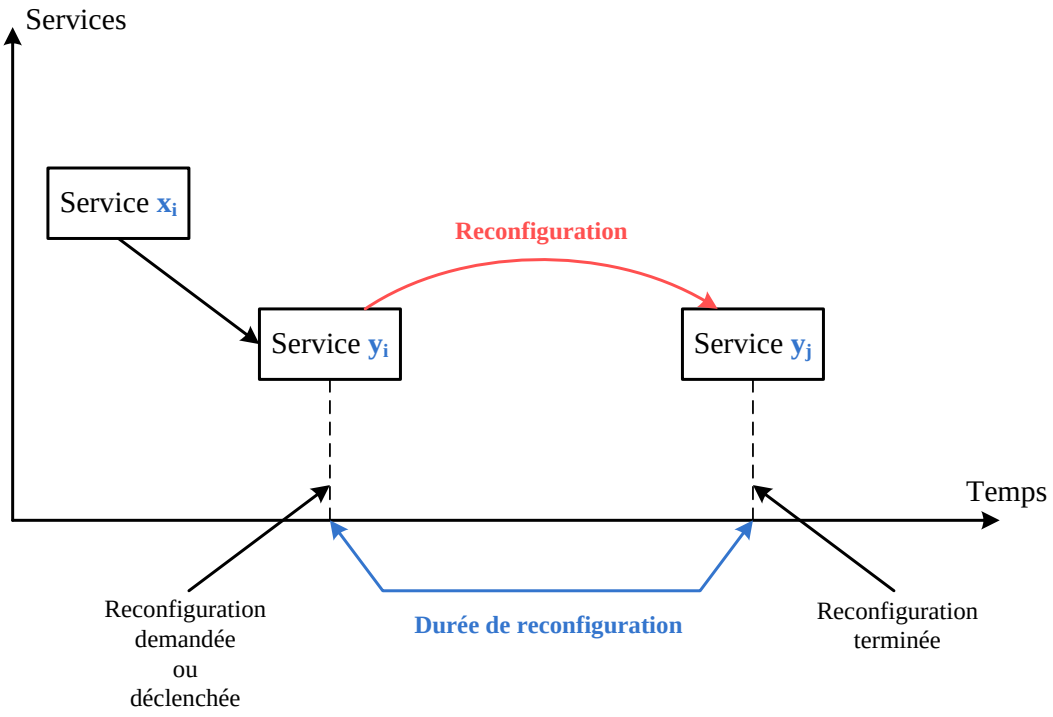


Figure 21 : Durée d'une reconfiguration

- La *durée de coupure* : C'est la durée pendant laquelle le service reconfiguré ne peut traiter aucune tâche demandée. Cette durée ne peut pas dépasser le temps de reconfiguration (figure 22).

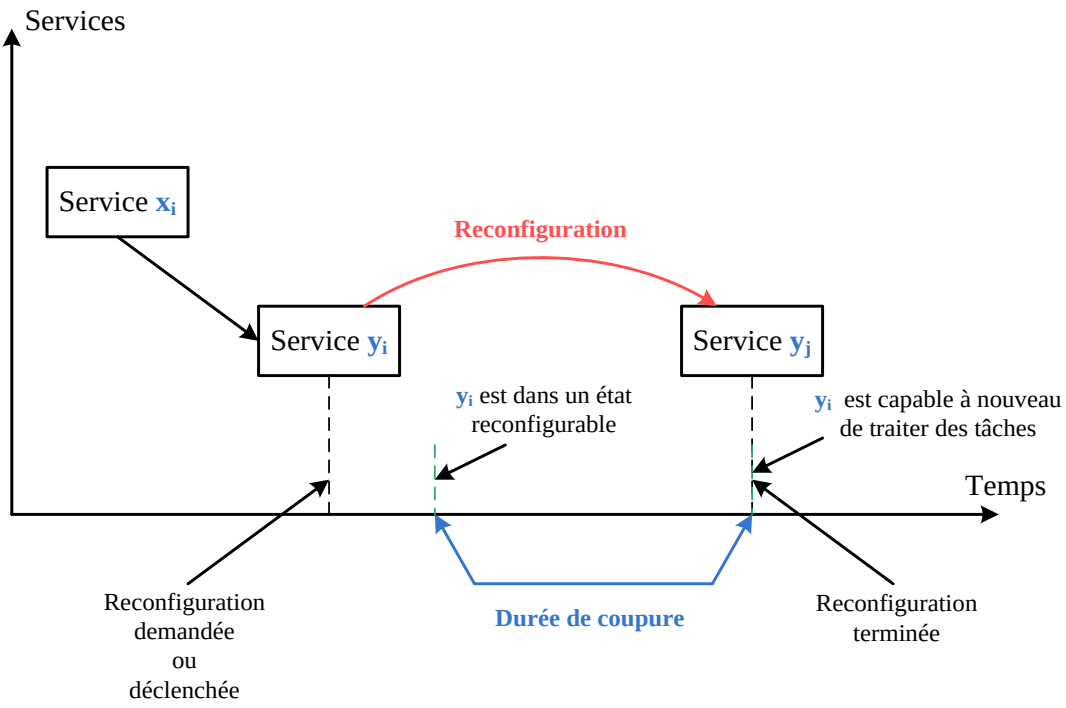


Figure 22 : Durée de coupure

Une reconfiguration se produit en remplaçant une instance d'un service par une version plus récente du même service. Toutefois, au cours de cette procédure, à partir du moment où la reconfiguration est lancée par un troisième service, jusqu'au moment où la nouvelle version du service est active, des événements peuvent s'afficher et affecter l'état de l'ancienne version du service. Pour des raisons de cohérence, cet état doit être le même pour la nouvelle version du service. Ainsi, l'état doit être transféré de l'ancienne version du service à la nouvelle pour maintenir la continuité du fonctionnement. Pour réduire le temps de coupure et, par conséquent, le temps total de reconfiguration, le transfert d'état doit durer le moins longtemps possible. Ce critère est encore plus critique lorsqu'il s'agit de temps réel, car il n'est pas possible d'arrêter le système.

2.2 Classification de la reconfiguration

La reconfiguration est une modification en temps réel d'un système (Hnětynka et Plášil 2006) (Batchkova et al. 2013). Cette modification peut amener le système d'un état courant (décrit par ses propriétés, son emplacement, ses dépendances, ses implémentations, etc.) à un autre état objectif dit état cible. Il s'agit alors de réajuster la structure interne du système sans changer sa fonction principale, ce qui peut être considéré comme étant un processus de recombinaison des cibles. Cela peut faire référence à une recombinaison de la structure d'un système, à la correction des défaillances d'un service, à la mise à jour en ligne des services et des modules du système existant, à l'augmentation et au déploiement dynamique des nouveaux services, …

La reconfiguration est basée sur des actions de reconfiguration simples et/ou complexes. Les actions simples de reconfiguration habituelles sont l'ajout/suppression d'un composant, la modification d'un attribut, l'ajout/suppression d'un lien et l'ajout/suppression des ressources d'une interface de contrôle. Les actions complexes peuvent être représentées comme une combinaison d'un ensemble d'actions simples.

Dans la littérature (Jiménez 2017), la reconfiguration peut être :

- Une reconfiguration *structurelle* : cette reconfiguration entraîne une modification de la structure du système comme par exemple dans le cas de la suppression d'un composant.
- Une reconfiguration *de l'implémentation* : elle conduit à modifier l'implémentation d'un ou de plusieurs composants, mais la structure du système reste inchangée comme par exemple dans le cas de l'ajout d'une interface.
- Une reconfiguration *comportementale* : ce type de reconfiguration entraîne une modification du comportement d'une ou de plusieurs ressources comme par exemple dans le cas de la modification d'un attribut.

La reconfiguration porte alors soit sur la partie opérative soit sur la partie contrôle pour tenir compte de ces trois aspects :

- La reconfiguration de la partie opérative consiste à intervenir sur le plan matériel afin de modifier ses capacités opérationnelles. Il est alors possible de compenser les services perdus par l'introduction de nouveaux services suite à la reconfiguration (Deschamps 2007).
- La reconfiguration du contrôle consiste à posséder une commande reconfigurable capable de s'adapter et d'exploiter les services encore disponibles offerts par la partie opérative (Berruet et al. 2007). On parle alors de systèmes de contrôle reconfigurable SCR.

Notre contribution s'inscrit dans l'axe reconfiguration du contrôle de manière à assurer un contrôle sûr de fonctionnement lorsqu'un défaut est détecté.

3 Les systèmes de contrôle reconfigurable (SCR)

Ce paragraphe présente le cadre théorique qui sert de base pour définir les concepts clés pour le positionnement et l'évaluation des SCR. De manière générale, les SCR sont des systèmes de contrôle dont l'architecture peut changer en fonction des besoins du système (Da Silva et al. 2016). La figure 23 illustre la relation entre le SCR et les besoins du système.

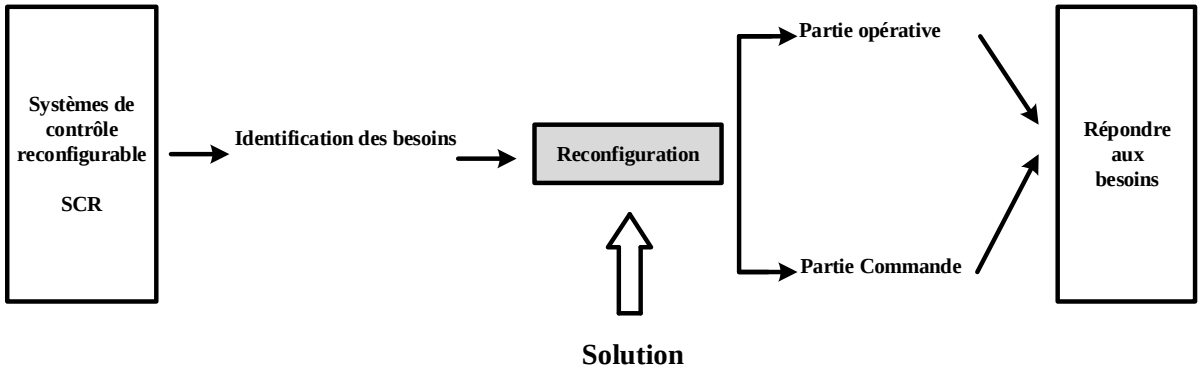


Figure 23 : Principe des systèmes de contrôle reconfigurable

Le processus de la reconfiguration (en gris sur la figure 23) est motivé par des besoins très variés (Piriou 2015). En effet, la reconfiguration d'un SCR peut être souhaitée, entre autres, pour des besoins fonctionnels (suite à un changement de configuration par exemple), pour des besoins de production (pour s'adapter aux nouvelles modifications du cahier des charges et des spécifications) et pour des besoins de sécurité et de tolérance aux fautes (pour conduire le

système dans un état cible tout en maintenant le bon fonctionnement du système). Pour répondre à ces besoins, la reconfiguration peut être appliquée soit à la partie opérative (PO), soit à la partie commande (PC) ou soit aux deux en même temps.

3.1 Définition des systèmes de contrôle reconfigurable

La définition des SCR a évolué au fil du temps. Les définitions changent selon le fonctionnement et le champ d'application (Bortolini et al. 2018). Plusieurs définitions sont actuellement proposées. Dans (Batchkova et al. 2013), les auteurs définissent un SCR comme un système capable de réorganiser dynamiquement les éléments de l'architecture de contrôle afin de prendre en compte l'occurrence des défaillances et des nouvelles exigences. Dans (Konstantopoulos et Antsaklis 1999), les auteurs considèrent les SCR comme étant des systèmes qui se caractérisent par leur capacité à fonctionner en présence de changements radicaux dans la dynamique du système. Ces changements sont liés, par exemple, à des défaillances brusques des composants du système (actionneur/capteur) ou à des changements rapides des spécifications de fonctionnement. Ces définitions montrent que les SCR sont basés sur la modification des éléments ou des composants du système, en particulier sur la modification des éléments de l'architecture de contrôle.

Cependant, il existe d'autres approches qui considèrent que les SCR sont basés sur une modification de l'objectif pour lequel le système de contrôle est conçu et sur une modification de ses interactions avec l'environnement (objectif de contrôle, politique, stratégie, etc). Par exemple, dans (Zhang et Jiang 2008) les auteurs soutiennent l'idée que les SCR sont des systèmes de contrôle capables de gérer les variations de l'environnement en considérant le réarrangement dynamique basé sur l'ajustement de la stratégie de contrôle. Dans (Simon et al. 2002), les SCR sont définis comme des systèmes de contrôle conçus pour réagir aux défaillances en tenant compte des modifications des objectifs de contrôle. Cependant, ces dernières approches ne parviennent pas à définir complètement le concept des SCR, car la notion de *reconfigurabilité* présentée est généralement considérée isolément pour un ensemble d'entrées de contrôle et de sorties contrôlées, sans tenir compte de son intégration dans un processus de prise de décision global (Sachidananda et al. 2016), (Dennis et al. 2010).

3.2 Objectifs des systèmes de contrôle reconfigurable

L'objectif général d'un SCR est d'effectuer les changements nécessaires pour assurer l'efficacité, le bon fonctionnement et la réactivité du système de contrôle. D'une part, la recherche d'une performance globale définie par l'efficacité et la réactivité implique que celles-ci soient construites pour assurer la réalisation des objectifs du système de manière efficace et réactive (Park et Kremer 2015). D'autre part, la réactivité du SCR implique que ce dernier est conçu pour assurer la stabilité du fonctionnement du système de contrôle et récupérer les performances du contrôle en cas de dégradation (Konstantopoulos et Antsaklis 1999).

Les SCR sont considérés comme étant une composition de systèmes liés entre eux et qui agissent ensemble pour résoudre un problème de contrôle global spécifique (figure 24). Ces systèmes se composent alors du mécanisme de reconfiguration, du système de contrôle (partie commande) et du système contrôlé (partie opérative).

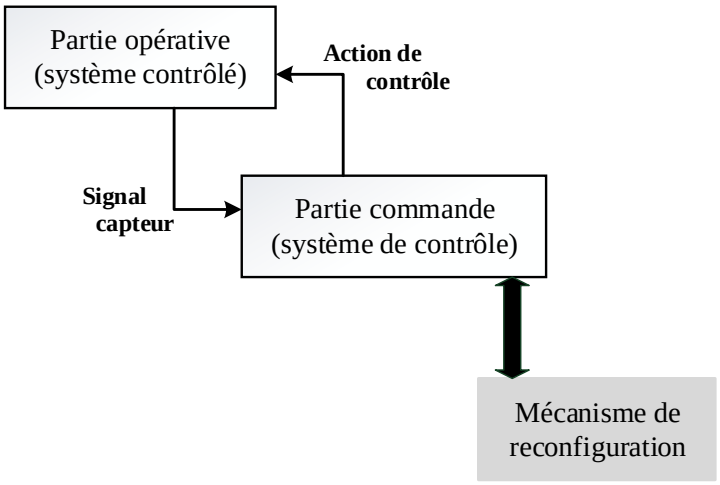


Figure 24 : Principe des systèmes de contrôle reconfigurable

Chacun de ces systèmes (système de contrôle et système contrôlé) possède sa structure et son comportement, chacun étant configuré pour atteindre ses propres objectifs. La communication entre eux est claire car ils sont liés par l'architecture globale de contrôle de la reconfiguration. Néanmoins, il convient de noter que ces systèmes sont fortement interdépendants. Un système automatisé de production avec une architecture de contrôle reconfigurable est un exemple de SCR. En effet, d'une part, le système de contrôle du SAP contribue à la réalisation de l'objectif du système contrôlé (traitement des opérations) et vise, d'une part, à atteindre les objectifs de contrôle (efficacité, bon fonctionnement et réactivité) et, d'autre part, le SCR contribue à la réalisation du but fixé pour le système de contrôle en cherchant également la meilleure configuration.

Un SCR vise à assurer simultanément la récupération et la stabilité des systèmes de contrôle. Ces deux objectifs améliorent les performances du système de contrôle ainsi que les performances globales du système contrôlé (Moor 2016).

3.3 Architecture des systèmes de contrôle reconfigurable

Les dernières années ont été marquées par une tendance croissante à la reconfiguration de l'architecture de contrôle au cours de l'exécution (Manceaux, 2015). Cette nouvelle approche, appelée ici architecture de contrôle dynamique (ACD), contribue à la gestion et à la manipulation des événements qui apparaissent pendant l'exécution du système de contrôle. En d'autres termes, l'ACD modifie la structure et le comportement de l'architecture de contrôle en répondant aux exigences du système suite à l'apparition d'événements. Le système de contrôle commence par une configuration initiale qui change ensuite dans le temps afin de fournir les fonctionnalités et la capacité nécessaires au besoin à l'instant présent. Ensuite, lors de l'occurrence d'une perturbation, le système de contrôle vise à maintenir ses performances pendant l'exécution et à rétablir les performances initiales attendues en apportant les modifications appropriées au sein de l'architecture de contrôle (Terkaj et al. 2009). En résumé, les approches associées aux ACD visent à adapter la solution de contrôle pour faire face à la complexité et à l'incertitude de tout système.

Les avantages et les caractéristiques des SCR reposent sur les modifications de l'architecture de contrôle qui devient alors une architecture de contrôle reconfigurable (ACR). Comme expliqué dans le paragraphe ci-dessus, les ACR présentent deux caractéristiques

principales : d'une part, ce sont des architectures de contrôle capables de modifier leur arrangement structurel et leur fonctionnement comportemental en cas de besoin. D'autre part, ils sont capables d'adapter la configuration de l'architecture de contrôle pour gérer les exigences de complexité et d'incertitude. Pour atteindre ces deux caractéristiques, les SCR contiennent un mécanisme de reconfiguration qui modifie et adapte la configuration de l'ACR. Ce mécanisme de reconfiguration permet d'évaluer les besoins du système de contrôle et d'implémenter les modifications spécifiques et nécessaires pour un changement de configuration. Cependant, pour exécuter cette reconfiguration, les composants des architectures de contrôle ont certaines caractéristiques permettant la possibilité de changement dans l'architecture.

Les SCR s'appréhendent selon deux axes (figure 25) : (i) l'axe 1 sur l'analyse des propriétés et des caractéristiques et, (ii) l'axe 2 sur les moyens permettant d'atteindre les objectifs demandés.

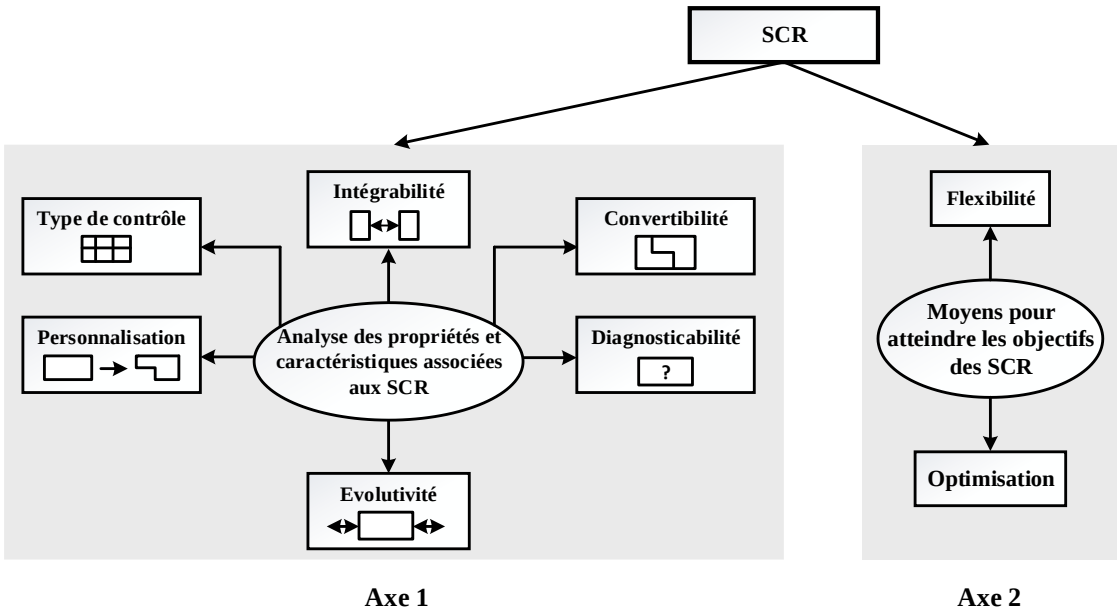


Figure 25 : Les deux axes d'un système de contrôle reconfigurable

4 Analyse des propriétés et des caractéristiques associées aux SCR

Une liste précise des propriétés qui caractérisent un SCR est difficile à déterminer. Plusieurs travaux dans la littérature ont traité ce sujet. Des caractéristiques communes sont proposées dans (Koren et al. 1999), (Rösiö et al. 2019), et (Huang et al. 2018). On peut définir ces caractéristiques selon le type de contrôle, selon des propriétés d'intégrabilité, de convertibilité, de diagnosticabilité, d'évolutivité et selon le concept de personnalisation.

4.1 Le type de contrôle des systèmes de contrôle reconfigurable

4.1.1 Système de contrôle

La préoccupation majeure de tout SCR est la nécessité d'un système de contrôle, c'est-à-dire un système séparé qui surveille et guide le fonctionnement global du SAP pour répondre à ses propres besoins. Le concept de contrôle est basé sur une boucle de retour entre deux systèmes : le système contrôlé (Partie Opérative) et le système de contrôle (Partie Commande) (Morel et al. 2019).

Un SCR peut être modélisé par des systèmes à évènements discrets (SED) (Cassandras et Lafortune 2008). Contrairement aux systèmes continus, un SED est un système qui satisfait les deux propriétés suivantes : (i) l'espace d'états est discret et (ii) la transition entre état est déclenchée par l'occurrence d'un événement. L'utilisation des SED vise, entre autres, à contrôler les SCR sous différentes architectures. Plusieurs démarches existent pour assurer le contrôle des SCR modélisés par des SED. Dans la littérature, on distingue trois méthodes formelles majeures (figure 26) :

- Les approches par vérification formelle et validation (V&V)
- Les approches par filtre logique
- Les approches basées sur la théorie de contrôle par supervision (SCT)

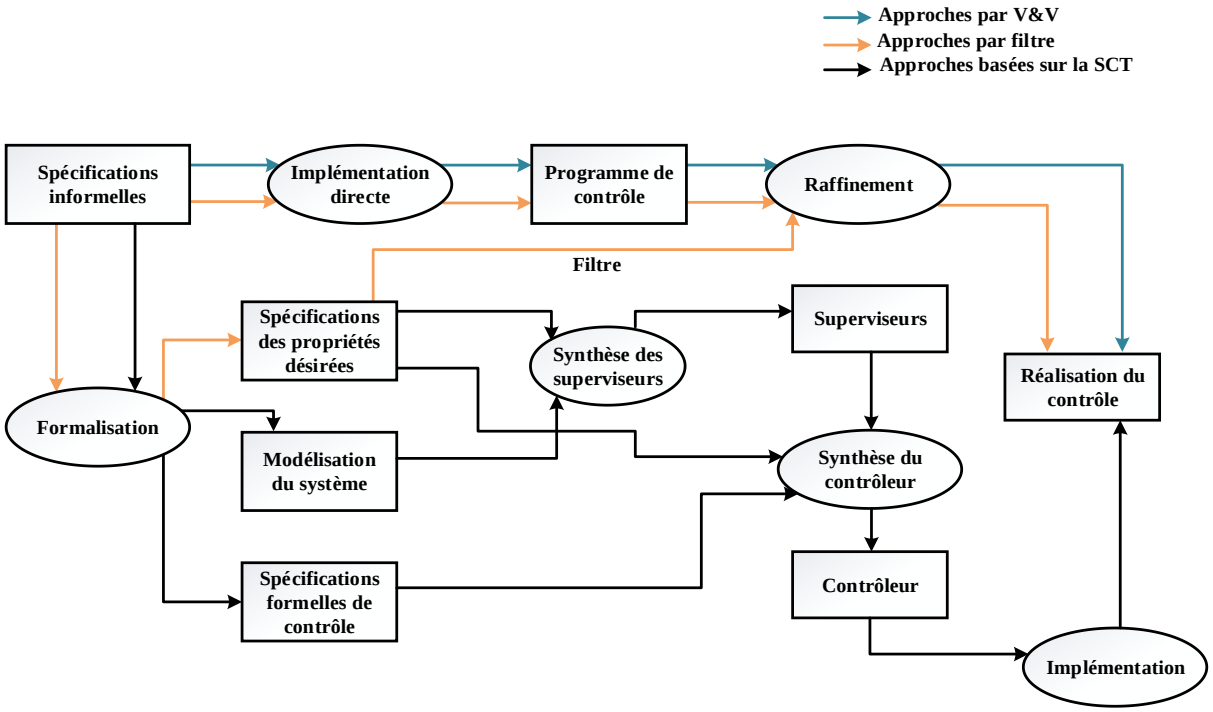


Figure 26 : Activités et modèles impliqués dans la synthèse et l'implémentation d'un contrôleur (inspirée de (Zaytoon et Riera 2017))

Les méthodes de V&V formelle consistent à développer des modèles de contrôle/commande en se basant sur les spécifications, puis ensuite d'analyser automatiquement une représentation formelle de ces modèles (Roussel et al. 2004), (Baier et Katoen 2008). La sécurité, la vivacité, la bornitude, l'atteignabilité et le non-blocage présentent les caractéristiques des modèles à vérifier. Tandis que la validation consiste à déterminer si le modèle déterminé représente bien l'objectif du concepteur. Les techniques de V&V formelles des programmes de contrôle/commande se basent généralement sur les techniques de « model-checking » (Berard et al. 2001).

Les méthodes à base de filtres logiques consistent à implémenter directement le programme de contrôle ou certains fragments structurés du programme de contrôle, puis à utiliser un modèle des propriétés spécifiées ou une abstraction des éléments de la partie opérative comme filtre entre le contrôleur (partie commande) et l'ensemble des capteurs/actionneurs (partie opérative) pour obtenir une réalisation de contrôle qui satisfait les spécifications. D'une part, ces méthodes permettent de s'assurer que les commandes émises soient correctes, et d'autre part que le retour de la partie opérative correspond à un fonctionnement attendu (détection de défauts) (Pichard 2018).

Les méthodes basées sur la théorie de contrôle par supervision (SCT) (Ramadge et Wonham 1989) consistent à synthétiser un superviseur (S) à partir d'une description globale de la partie opérative du procédé étudié (P) et des spécifications formelles. Le superviseur résultant vise à respecter l'ensemble des spécifications tout en offrant une flexibilité comportementale maximale (le plus permissif possible) et garantissant le non-blocage et la sûreté de fonctionnement du système. Par la suite, un contrôleur est synthétisé en raffinant le superviseur (Wonham et al. 2018). C'est une méthode basée sur des modèles et des machines à états finis (MEF).

Pour les travaux dans cette thèse, nous avons besoin de modèles de partie opérative et de spécifications pour exploiter les travaux déjà proposés au sein de notre équipe de recherche. Nous avons retenu alors une méthodologie à base de modèles pour concevoir une commande reconfigurable et tolérante aux fautes. Ce choix est orienté par la motivation de l'exploitation des modèles. Pour garantir par construction le non-blocage et la sûreté de fonctionnement, nous avons choisi de développer ces travaux dans le cadre de la théorie de contrôle par supervision. Pour expliquer et justifier nos choix, nous présentons différentes architectures de contrôle en lien avec la SCT.

4.1.2 Théorie de contrôle par supervision et architecture de contrôle

4.1.2.1 Architecture centralisée

L'approche de contrôle centralisée (figure 27) consiste à délocaliser le superviseur (S) par rapport à la structure physique du procédé (P). Le superviseur centralisé est responsable de la prise des décisions pour la réalisation du contrôle global. Comparée aux autres approches de contrôle, l'approche centralisée offre une application stricte et sécurisée de la commande. Néanmoins, les SAP basés sur ce type de contrôle ont commencé à faire face à la fois à une faible réactivité aux perturbations et aux changements inattendus (défaillance, …) et à la difficulté d'apporter une modification immédiate au moment de la variabilité. Ces limitations ont instancié la proposition de différentes architectures de contrôle que nous citons juste après (Wonham, Cai, et Rudie 2018b) (Zaytoon et Riera 2017).

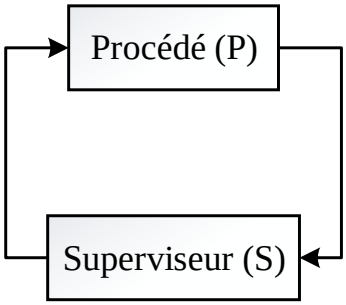


Figure 27 : Architecture de contrôle centralisée

4.1.2.2 Architecture modulaire

L'approche modulaire se base sur une décomposition des spécifications (ensemble d'exigences fonctionnelles à satisfaire exprimées par le cahier des charges) afin de créer plusieurs superviseurs au lieu d'un seul tel qu'il est évoqué ci-dessus (Wonham et Ramadge 1988). Chaque superviseur représente ainsi une spécification unique et tous ces superviseurs locaux agissent simultanément pour restreindre le comportement du procédé (figure 28).

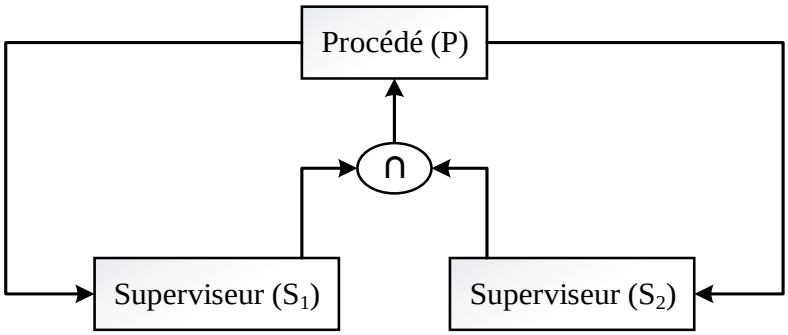


Figure 28 : Architecture de contrôle modulaire

Comme le montre la figure 28, le superviseur S est décomposé en deux sous-superviseurs notés S_1 et S_2. Ils sont construits par rapport au procédé G et leur intersection forme le superviseur S_{12} qui restreint le procédé. Chaque superviseur autorise ou interdit un ensemble d'événements et seul ceux qui ne sont interdits par aucun superviseur peuvent être générés. Formellement, la détermination du superviseur S_{12} est définie par $S_{12}= S_1 \cap S_2$.

4.1.2.3 Architecture décentralisée

L'approche décentralisée (figure 29) reprend en partie l'approche modulaire présentée dans le paragraphe précédent au niveau de la décomposition en plusieurs superviseurs et propose, en plus, de réduire la taille du procédé (Khoumsi et Chakib 2014). Dans un SAP, le contrôle peut être réparti en plusieurs contrôleurs, chacun de ces derniers n'observe qu'une partie du système. Le concept ici est construire, pour chaque superviseur local, une abstraction du système complet qui ne tienne compte que des événements de ce superviseur local. Cette abstraction se fait en utilisant une fonction de projection.

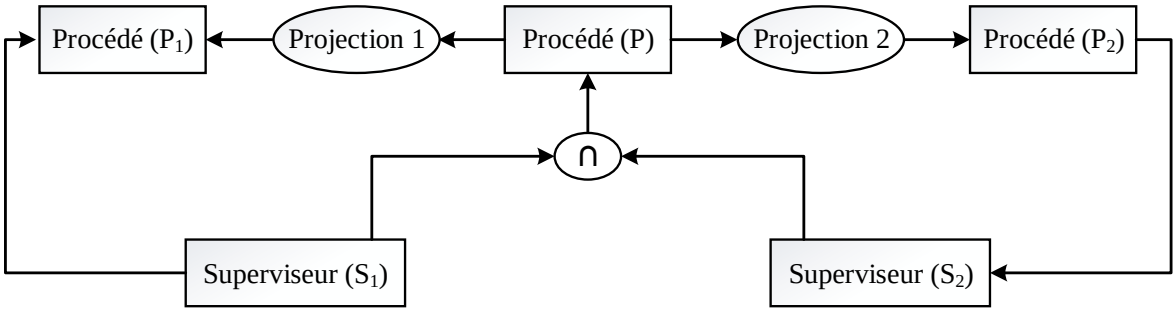


Figure 29 : Architecture de contrôle décentralisée

Comme dans la structure modulaire, un événement du procédé ne peut avoir lieu que s'il est autorisé par tous les superviseurs. Cette structure permet alors de réduire la taille des modèles, y compris les modèles du procédé. Elle facilite donc la compréhension et l'analyse.

4.1.2.4 Architecture hiérarchique

L'approche hiérarchique est fondée sur une conception à plusieurs niveaux des modèles. Dans (Zhong et Wonham 1990), les auteurs ont défini deux niveaux : le premier est composé d'un modèle de procédé réel et de son superviseur vu comme étant un niveau bas. Le deuxième est considéré comme un niveau haut contenant le modèle du procédé abstrait et son superviseur associé (figure 30). Les modèles d'indice *b* sont ceux de bas niveau et ceux d'indice *h* sont ceux de haut niveau. Ces deux niveaux sont couplés par un canal d'information θ qui permet de construire le modèle du procédé de niveau haut à partir de P_b et de synchroniser les actions des

superviseurs S_b et S_h. Ce canal permet aussi de restreindre le comportement de niveau bas à partir des restrictions de niveau haut

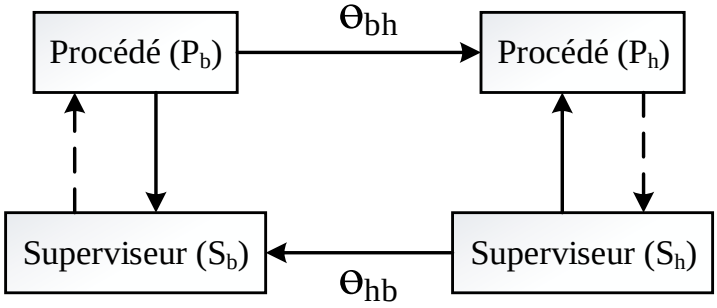


Figure 30 : Architecture de contrôle hiérarchique

Cette structure permet elle aussi de réduire la taille des modèles, sauf celle du procédé de bas niveau qui a la même taille que le procédé dans la structure classique.

4.1.2.5 Architecture distribuée

L'approche distribuée (Mehta et Reddy 2015) repose sur la décomposition du procédé en plusieurs éléments (EP_n), chaque élément de procédé est contrôlé par un superviseur S_n. Le contrôle global est assuré par des systèmes de communication entre les différents superviseurs (figure 31).

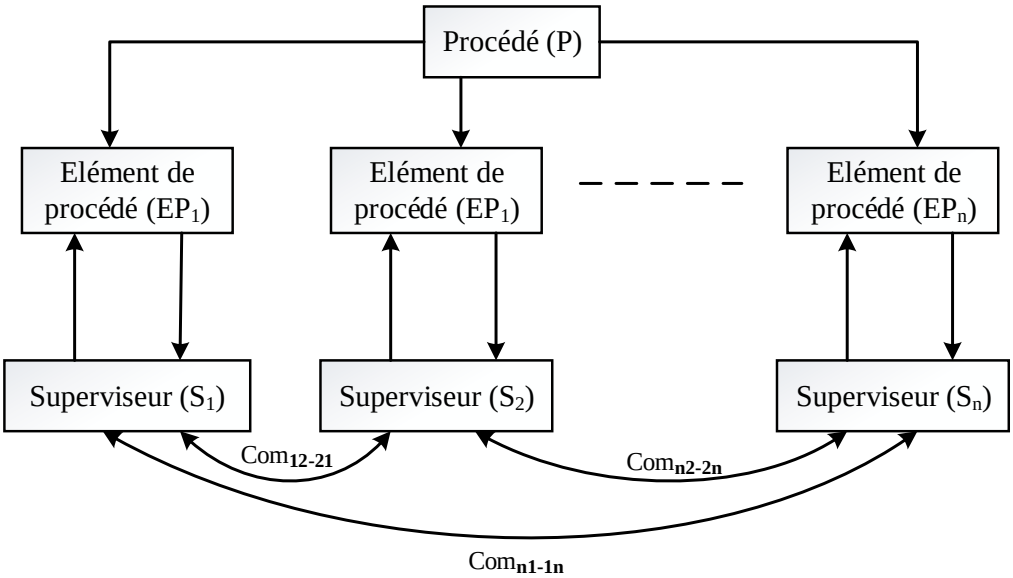


Figure 31 : Architecture de contrôle distribuée

Le système de contrôle décompose alors le problème de contrôle global en sous-problèmes afin de les résoudre collectivement entre différents composants. Cette structure permet de réduire la taille des modèles de contrôle ainsi que celle du procédé, ce qui permet une analyse facile du système.

4.2 Propriété d'intégrabilité (Integrability)

L'intégrabilité est la capacité d'inclure de nouveaux systèmes et composants dans le système de production existant, pour intégrer des nouvelles technologies et ajouter ou

supprimer des ressources (Farid 2017). Dans ce contexte, l'auteur a développé un processus de mesure de la reconfigurabilité afin de fournir une estimation quantitative du potentiel d'intégrabilité. En particulier, une méthodologie de conception est utilisée pour obtenir des mesures du degré de liberté de production qui représentent le potentiel de reconfiguration d'un système de production. Pour modéliser l'intégrabilité, chaque degré de liberté est déduit de l'effort nécessaire pour l'intégrer au reste du système. Dans (Wang et al. 2017), les auteurs modélisent l'intégrabilité comme étant la capacité d'intégrer des composants, par exemple les machines de fabrications et les modules de commande, grâce aux interfaces des composants et à la capacité d'intégrer une nouvelle technique ou un nouveau processus au système actuel. Dans cette étude, les auteurs prouvent l'existence d'une relation inverse entre le temps et le coût d'ajustement de l'interface logiciel/matériel et l'intégrabilité d'un SAPR.

4.3 Propriété de convertibilité (Convertibility)

La convertibilité est la capacité d'un système à ajuster la fonctionnalité de la production ou de passer d'un produit à l'autre en répondant aux changements dynamiques du marché. La littérature propose des mesures pour la convertibilité d'un système en considérant la convertibilité de la configuration, de la machine et du matériel dans la même formulation mathématique, dans laquelle un poids est attribué à chacune de ces trois mesures (Bruccoleri et al 2003), (Gumasta et al. 2011). Dans (Farid 2017), l'auteur considère la mesure de convertibilité comme étant la somme de la transformabilité et de transport de la convertibilité. Le premier critère se réfère aux plans de processus, le second au matériel du système. Dans (Wang et al. 2017), les auteurs proposent une formulation mathématique de la convertibilité du système en fonction de sa capacité à produire différentes familles de pièces et différentes pièces de la même famille. Ces travaux proposés dans la littérature tentent d'intégrer la convertibilité dans le développement d'un indice général de reconfigurabilité, dans lequel toutes les caractéristiques du SAPR sont incluses et modélisées.

À titre d'exemple, la figure 32 montre deux lignes de production qui sont comparées. Pour le système (1), le flux de production est linéaire, tandis que pour le système (2), le flux de production contient deux lignes de production en parallèle. L'intégration d'une nouvelle pièce à produire dans la chaîne (1) conduit à un arrêt total du système jusqu'au changement des spécifications selon les nouvelles exigences du produit. Ensuite, la chaîne de production peut redémarrer. Par contre en (2), si un nouveau produit est introduit, une seule ligne sera arrêtée pour reconfiguration tandis que l'autre ligne peut continuer la production (Sohaleh 2017).

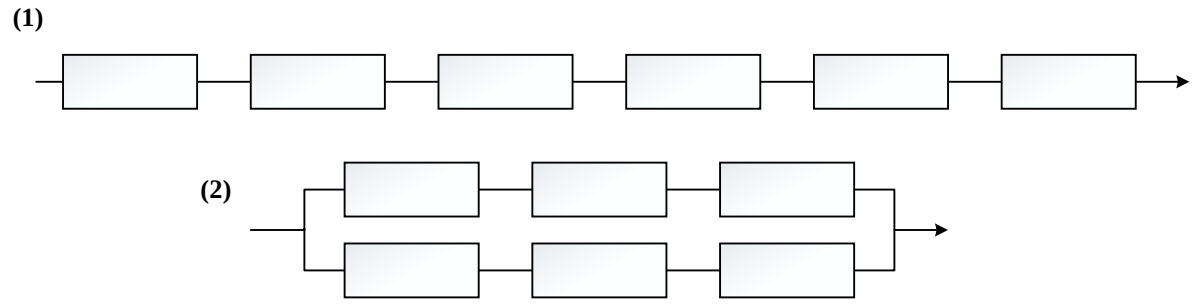


Figure 32 : Comparaison de la convertibilité du système

4.4 Propriété de diagnosticabilité (Diagnosability)

Le diagnostic des pannes d'un SAPR s'effectue soit en ligne pendant le fonctionnement du système, soit hors ligne à travers l'analyse de diagnosticabilité du système. Cette notion de diagnosticabilité est considérée comme étant une propriété importante lors de la conception des systèmes sûrs de fonctionnement. Dans (Sampath et al. 1995), les auteurs définissent la diagnosticabilité comme étant la capacité d'un système à diagnostiquer ses propres défauts apparaissant lors du fonctionnement à l'aide de l'ensemble de ses observations. Suite à ce travail, de nombreuses approches ont été proposées dans cet axe de recherche telles que celles développées par (Lafortune et al. 2018) et (Li et al. 2017), etc.

Si de très nombreux travaux existent autour de la diagnosticabilité, peu s'intéressent à la problématique de la reconfiguration. Dans (Bruccoleri et al. 2003), les auteurs proposent un système de contrôle distribué et intelligent pour prouver que la fonctionnalité de la reconfiguration des machines est adaptée au traitement des pannes des machines. Dans (Bruccoleri et al. 2006), les auteurs ajoutent que des événements non prévus peuvent être gérés par la reconfiguration, en introduisant le concept de reconfiguration pour la gestion des défauts et en développant une architecture de contrôle orientée pour la gestion des fautes prise en charge par cette reconfiguration.

Dans notre étude, la diagnosticabilité est primordiale car elle permet de basculer dans un mode dégradé ou non. Elle est considérée comme connue et par conséquent non développée dans ce manuscrit.

4.5 Propriété d'évolutivité (Scalability)

L'évolutivité est l'aptitude à modifier facilement la capacité de production en ajoutant ou en supprimant des ressources de fabrication et en modifiant les composants du système pour répondre à l'évolution de la demande. Du point de vue technico-économique, cette caractéristique devrait être introduite dans la phase de conception d'un nouveau SCR (Wang et Koren 2012). Le processus de planification de l'évolutivité nécessite de modifier simultanément la configuration du système et de rééquilibrer le système reconfiguré de manière rentable (Deif et ElMaraghy 2007), (Elmasry et al. 2015). La littérature associée à ce sujet de recherche montre que la première étude réalisée dans ce domaine est proposée par (Son et al. 2001), en définissant une procédure mathématique pour mettre à niveau la capacité des lignes en série composées de machines à commande numérique. Dans (Wang et Koren 2012), les auteurs proposent une méthodologie pour la planification de l'évolutivité dans des systèmes sans stock, en considérant comme fonction objective la minimisation du nombre de machines nécessaires pour répondre à la nouvelle demande du marché. Dans une contribution plus récente (Yoram Koren et al. 2017), les auteurs ont développé l'étude précédente en définissant un modèle mathématique appliqué aux systèmes à stocks dans lesquels la productivité après la reconfiguration est maximisée. Un autre travail est proposé dans (Deif et ElMaraghy 2006), un modèle dynamique de la capacité d'évolutivité est défini en se basant sur une approche de contrôle afin d'indiquer la meilleure conception pour l'évolutivité du contrôle. Dans une telle conception, les auteurs doivent prendre des décisions de compromis entre la réactivité du système et le coût. Une version améliorée de ce travail a été proposée dans (Deif et ElMaraghy 2007) où les auteurs définissent un modèle pour la planification de la capacité d'évolutivité pour les SCR en tenant compte des coûts d'investissement totaux. Dans (Gumasta et al. 2011), les auteurs introduisent une formulation mathématique de la mesure de l'évolutivité, considérant celle-ci comme étant

la capacité de maintenir la rentabilité lorsque la charge de travail augmente. De même que pour la mesure de convertibilité, l'évolutivité est incluse dans l'élaboration d'un indice général de reconfigurabilité.

4.6 Notion de personnalisation (Customisation)

La personnalisation fait référence à la sélection des outils des machines et des composants du système en se basant sur le besoin de la flexibilité pour traiter une famille de pièces spécifiques. Dans (Farid 2017) et (Wang et al. 2017), les auteurs définissent un indice de reconfigurabilité comprenant une formulation pour la personnalisation.

La personnalisation est un moyen permettant d'atteindre les objectifs de l'architecture de contrôle. Le terme *personnalisation* est utilisé ici pour désigner la capacité du système de contrôle à définir et à adapter une configuration spécifique de l'architecture de contrôle afin de répondre aux exigences de contrôle spécifiques. La personnalisation permet aussi de piloter l'architecture de contrôle selon les besoins et d'améliorer la contrôlabilité du système. Dans le contexte d'un SCR, la personnalisation offre la possibilité d'obtenir une configuration correcte non seulement au début, mais également lors de l'exécution du système. Le SCR peut ensuite être ajusté, remodelé ou autrement modifié pour répondre aux besoins. Néanmoins, dans la littérature, la personnalisation incombe généralement au concepteur du système. Malgré quelques tentatives méthodologiques, elle reste largement intégrée à la phase de planification hors ligne (Cotta et al. 2008). Cependant, la personnalisation dans un SCR offre un ensemble plus large d'alternatives pour répondre aux défis posés en ligne/temps réel. Pour cette raison, la personnalisation est une tendance importante qui offre de bonnes bases pour atteindre le ou les objectifs du système (Lyke et al. 2015).

5 Moyens pour atteindre les objectifs des SCR

Les moyens souvent employés pour atteindre les objectifs définis par les SCR sont la flexibilité et l'optimisation que nous définissons par la suite.

5.1 Flexibilité

Le degré de flexibilité d'un SCR représente un problème critique durant la phase de conception d'un système. L'analyse de la littérature (Terkaj et al. 2009) sur le thème de la flexibilité des systèmes automatisés de production met en évidence la présence de quatre différentes catégories de travaux scientifiques. Pour la *première* catégorie, plusieurs études ont porté sur l'analyse de la flexibilité des SAP et de la relation avec les problèmes liés à la production. La *deuxième* catégorie comprend des travaux sur la classification des formes de flexibilité existantes. Des approches plus récentes ont porté sur l'élaboration des méthodes et des modèles pour soutenir la conception du système tout en tenant compte des formes de flexibilité du système. Ces études constituent la *troisième* catégorie et, bien qu'elles aient apporté des contributions importantes au sujet de la flexibilité des SAP, l'ensemble de la structure qui soutient le processus de conception du système à partir des classifications de la flexibilité reste faible. Cet aspect représente le noyau de la *quatrième* catégorie, une ontologie sur la flexibilité est brièvement présentée dans le but de systématiser le nombre élevé de définitions de flexibilité, celles-ci pouvant être utiles lors de la phase de conception du système.

5.1.1 Analyse de flexibilité des SAP

L'analyse de la flexibilité des SAP a donné lieu à de nombreux travaux. Dans (Upton 1994), l'auteur présente la flexibilité comme étant la capacité de changer ou de réagir avec une incidence faible en termes de temps, d'efforts, de coût et/ou de performance. Dans de nombreux cas, l'analyse est étayée par des études empiriques, comme dans (Swamidass et Newell 1987). D'autres travaux ont étudié la manière dont les changements externes sont liés aux différentes formes de flexibilité et comment ils peuvent être réduits (Gerwin 1993).

Ces travaux ont mis en évidence la nécessité d'étudier la relation entre les exigences de la production et les formes de flexibilité des SAP. En particulier, le problème majeur consiste à identifier les formes de flexibilité qui répondent à des exigences internes, appelées flexibilité *interne*, et les formes de flexibilité qui gèrent les exigences externes, appelées flexibilité *externe*. Dans (Corrêa et Slack 1996), les auteurs ont souligné cette vision en développant deux grandes catégories d'exigences qui impliquaient un besoin de flexibilité : l'incertitude environnementale et la variabilité de la production requise par le système (par exemple durant la pandémie de covid19, plusieurs entreprises ont réadapté leur production pour fabriquer des masques). Ces deux phénomènes sont appelés *catalyseur* et *impact sur le système de production* par le biais de changements planifiés et non planifiés. Les changements planifiés résultent d'actions de gestion conscientes visant à modifier certains aspects du système ou ses relations avec l'environnement. Les changements imprévus se produisent indépendamment des intentions du SAP, mais ils appellent une réaction. Ce type de changement est appelé *catalyseur* agissant sur le système. Dans (Hyun et Ahn 1993), les auteurs ont introduit alors le concept de *flexibilité proactive* qui prend en compte les évolutions possibles du marché. Cela permet aux entreprises d'envisager également des stratégies différentes de la simple réaction aux changements du marché. En ce sens, la flexibilité influe fortement sur les niveaux de concurrence des entreprises.

5.1.2 Classification des formes de flexibilité

La classification des formes de flexibilité existantes représente un autre sujet qui a donné lieu à de nombreux travaux de recherche. Cet axe insiste sur l'importance de la systématisation des connaissances concernant toutes les formes de flexibilité proposées. De plus, le caractère multidimensionnel de la flexibilité justifie les efforts, au fil du temps, consacrés au développement de classifications où toutes les formes de flexibilité possibles sont classées et caractérisées.

Dans (Sethi et Sethi 1990), les auteurs ont proposé une classification définissant 11 dimensions différentes de la flexibilité. L'approche consiste à déterminer trois groupes principaux (figure 33) : i) la flexibilité des composants ou les flexibilités de base, qui comprennent les éléments de la flexibilité des machines, de la manipulation et l'exploitation du matériel, (ii) les flexibilités du système dont les flexibilités de processus, de routage, de produit, de volume et de développement sont prises en compte ; (iii) les flexibilités globales du programme, de la production, et du marché. En outre, les auteurs ont abordé de manière transversale la structure organisationnelle ainsi que la technologie des microprocesseurs. Pour valider ces dimensions de flexibilité identifiées, un « instrument » permettant la mesure de la flexibilité a été développé et présenté par (Gupta et Somers 1996). Les auteurs ont pu aussi démontrer que les 11 formes de flexibilité peuvent être réduites à neuf formes : flexibilité de la

machine, de la manipulation du matériel, du processus, du routage, du volume, du programme, du produit, de la production, du marché, du développement et la flexibilité du marché.

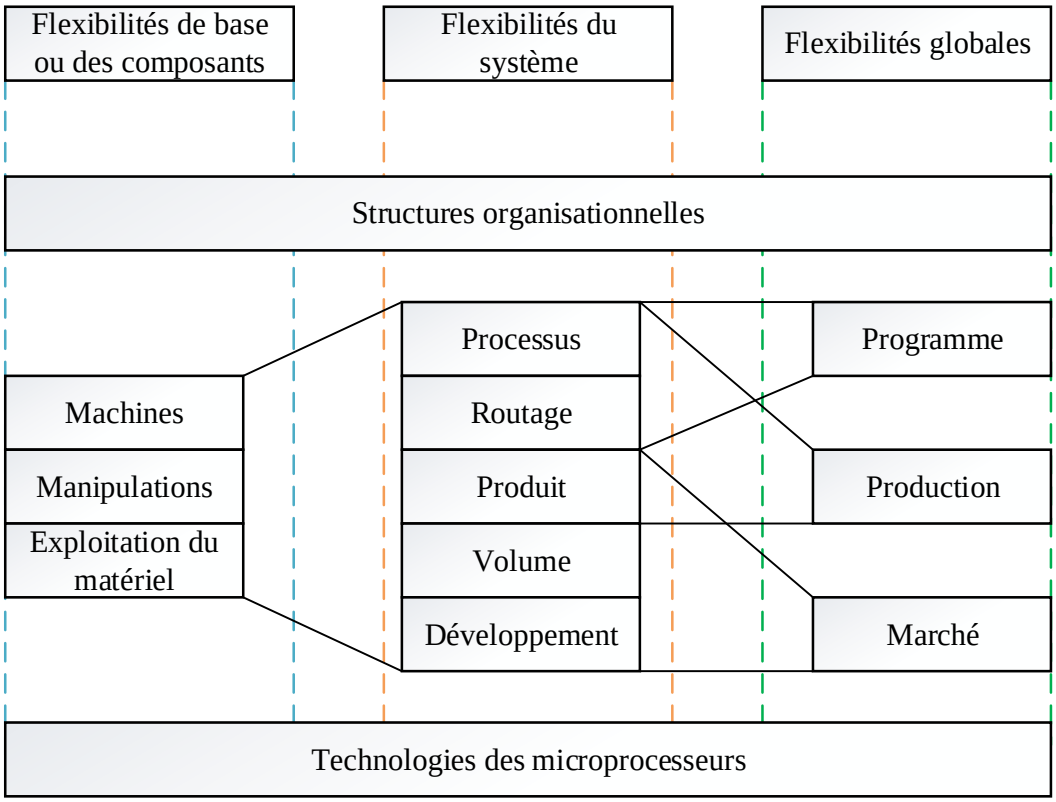


Figure 33 : Liens entre les différentes flexibilités (inspirée de (Sethi et Sethi 1990))

Dans (Kapitanov 2017), l'auteur a proposé une étude des principes de flexibilité pour les SAP, où il considère la flexibilité comme étant la capacité d'un SAP de passer à la fabrication de nouveaux produits caractéristiques en plusieurs, petites ou moyennes séries. En supplément, il a proposé des méthodes d'assurance de flexibilité des produits, des opérations et des routages, ainsi que celle de démarrage et du développement. Afin de choisir une option assurant la flexibilité du système, l'auteur a proposé de diviser l'ensemble d'un SAP en plusieurs classes. Pour chacune de ces dernières, un coefficient de pondération sans dimension est introduit, tout en prenant en compte le degré d'utilisation des capacités des unités de traitement et des machines pour l'évaluation de la flexibilité.

5.1.3 Conception de la flexibilité pour les SAP

La troisième catégorie des travaux traite des approches et des modèles pour implémenter la flexibilité lors de la phase de conception du système. Même si de nombreux efforts ont été consacrés à l'analyse de la flexibilité, le lien entre les connaissances à propos la flexibilité et la conception des SAP reste faible pour le moment. D'une part, l'acquisition de la flexibilité est considérée comme une option stratégique pour réagir aux changements fréquents de volume et aux évolutions des exigences technologiques des produits. D'autre part, de nombreux travaux soulignent la nécessité d'analyser en profondeur le risque associé à l'obtention d'un niveau élevé de flexibilité, en tenant compte des investissements correspondants (Matta et al. 2000). D'autres travaux ont étudié la relation entre le niveau de flexibilité intégré au système et la performance de ce dernier (Koren et al. 1999), (Landers et al. 2001). En effet, la flexibilité fait

référence à la capacité du système à faire face à l'instabilité induite par l'environnement dans lequel il fonctionne. Dans (Bordoloi et al. 1999), les auteurs ont proposé un modèle sur la capacité de développement par lequel une évaluation économique est réalisée pour souligner l'importance de la flexibilité et de l'adaptabilité des SAP. La décision quant au moment de choisir un système flexible est liée à l'analyse de risque des investissements. En effet, la capacité de flexibilité est coûteuse et, en addition, la stratégie visant à concevoir un haut niveau de flexibilité avec des informations incertaines pourrait impliquer des coûts importants. Bien que de nombreux modèles décisionnels traitent des valeurs attendues des coûts ou des bénéfices incertains, la réduction de la période de temps appelle une gestion des risques appropriée.

5.2 Optimisation

L'optimisation présente le deuxième moyen pour atteindre les objectifs fixés par les SCR. En tenant compte de la flexibilité de l'architecture de contrôle reconfigurable, l'agencement du SCR est considérée comme un problème d'optimalité dans lequel la meilleure configuration de reconfiguration doit être sélectionnée en fonction des besoins du système. Par conséquent, le défi du SCR est de sélectionner la configuration optimale pour atteindre les objectifs souhaités.

Plusieurs travaux dans la littérature s'adressent à l'optimalité des systèmes de contrôle et ceux contrôlés tels que (Abbas et ElMaraghy 2018), (Lennartson et al. 2014) et (Wardi et al. 2018). Cependant, très peu de recherches ont été faites sur l'optimalité de la reconfigurabilité. L'optimalité est un concept lié aux systèmes automatisés de production reconfigurables. Une démonstration sur le besoin de l'optimisation de la reconfiguration est donnée dans (Jiménez 2017). L'auteur a associé la définition du niveau d'optimalité d'un processus de reconfiguration aux méthodes de planification proposées dans (Baker 1998), dans lequel le processus est classé en fonction de l'approximation de la solution optimale. Une classe d'optimalité est définie en fonction de l'optimalité du processus de reconfiguration : optimale, quasi-optimale, vers optimal et heuristique. Une classe d'optimalité supplémentaire appelée réaction de transition est créée pour classifier les approches sans intention d'optimalité spécifique (absence d'optimalité) et elle est implémentée uniquement dans le but de poursuivre l'exécution. Dans (Macktoobian et Wonham 2017), les auteurs ont développé un algorithme pour résoudre le problème de reconfiguration : à partir d'un état courant où une reconfiguration est demandée, un ensemble de chemins amenant à l'état cible ou reconfiguré est défini. Ensuite le meilleur chemin est choisi en se basant sur la notion de forçage.

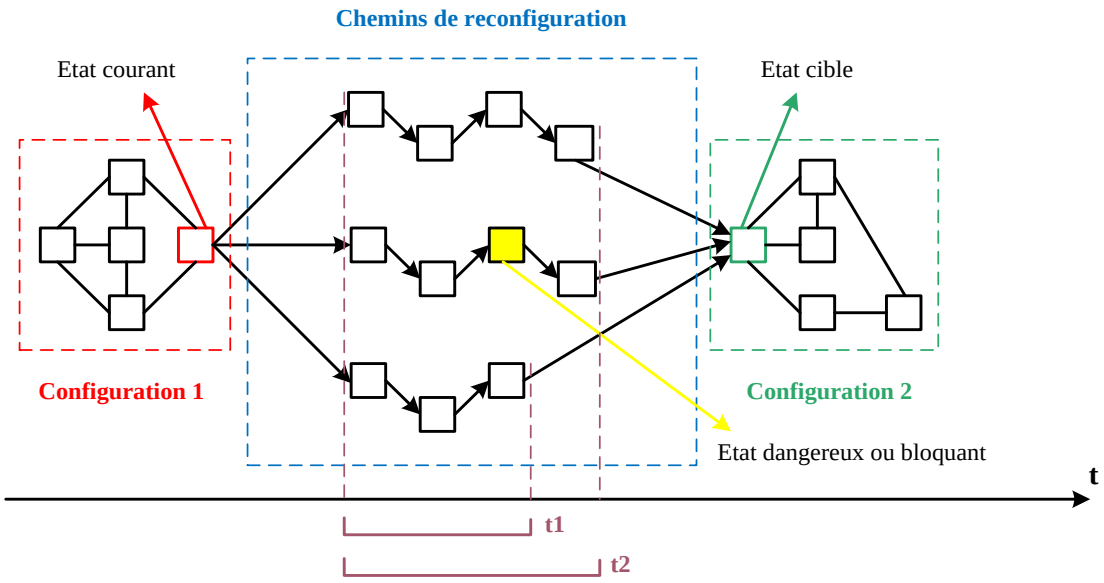


Figure 34 : Facteurs associés à l'optimalité des systèmes automatisés de production reconfigurables

L'optimalité peut être associée au facteur de temps ou / et au type de chemin parcouru, il est préférable d'achever une reconfiguration pendant une durée minimale (durée nécessaire pour switcher d'une configuration à l'autre). Pour passer d'un état courant d'une configuration 1 à un état reconfiguré d'une configuration 2, il est possible de passer par plusieurs chemins dont un peut être dangereux ou bloquant. C'est pourquoi il faut choisir le chemin le plus sûr (figure 34).

6 La reconfiguration et la tolérance aux fautes dans le cadre des SCR

Après avoir présenté le principe des systèmes de contrôle reconfigurable ses objectifs et ses architectures, nous avons détaillé les caractéristiques associées à ce type de systèmes et déterminé les moyens pour atteindre les objectifs. Dans cette section, nous présentons une étude bibliographique sur la reconfiguration et la tolérance aux fautes des systèmes de contrôle reconfigurable.

6.1 Principe de la commande tolérante aux fautes

L'arrivée massive de nouveaux produits sur le marché, l'amélioration continue des produits existants et les changements de législation concernant l'environnement et la sécurité font que les industries doivent s'adapter pour rester compétitives (Koren et al. 1999). Dans un contexte d'industrie 4.0, les SAP modernes sont confrontés à un marché international agressif composé de multiples changements imprédictibles. Les SAP sont ainsi devenus de plus en plus complexes en raison de l'arrivée de l'internet des objets, de la personnalisation des produits et de la part grandissante du logiciel dans les usines (ElMaraghy 2019), (ElMaraghy et al. 2012). La complexité accrue dans les systèmes induit une quantité importante d'informations qui peut conduire le système dans un comportement anormal.

La tolérance aux fautes peut être vue comme une classe de reconfiguration lorsqu'on parle d'une occurrence de faute. Les commandes tolérantes aux fautes ont pour objectif de maintenir le système disponible en annulant les comportements non désirés qui pourraient

apparaître lors de l'occurrence d'une défaillance. En cas de défaillance, le système doit identifier les ressources défectueuses pour les substituer par des ressources disponibles pour effectuer une reconfiguration.

Pour assurer une tolérance aux fautes, le système de contrôle reconfigurable doit être composé de trois éléments (figure 35) :

- Le contrôleur décrivant le comportement normal souhaité du système ainsi que son comportement dégradé,
- Le diagnostiqueur détectant les défauts,
- Le bloc de reconfiguration qui consiste à prendre la décision de passer du contrôleur de comportement normal au contrôleur tolérant aux fautes.

Une architecture générique de la reconfiguration assurant une commande tolérante aux fautes peut être alors déterminée par le schéma de la figure 35.

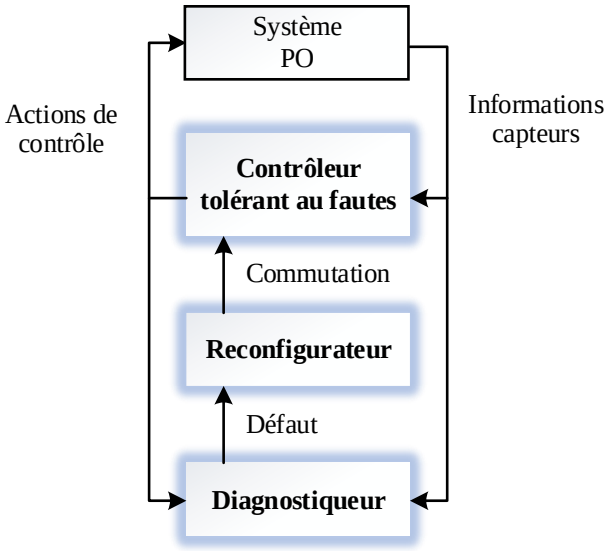


Figure 35 : Boucle de la reconfiguration de la commande (commande tolérante aux fautes)

6.2 Etude bibliographique sur la reconfiguration et la commande tolérante aux fautes

Dans cette section nous présentons des travaux de recherche présentés dans la littérature qui rassemble la reconfiguration et la tolérance aux fautes dans l'objectif d'assurer un contrôle sûr de fonctionnement.

Introduite dans les années 80, la commande tolérante aux fautes (CTF) a été l'objet de plusieurs travaux de recherche dans le domaine des SAP. L'objectif principal est d'éliminer les problèmes associés à l'apparition des fautes pouvant conduire à des comportements indésirables ou même catastrophiques. Ce genre de problématique a été souvent évité en s'appuyant sur une redondance matérielle au niveau des actionneurs et des capteurs. Cette stratégie est non seulement coûteuse, mais elle nécessite également un dispositif de maintenance majeur. Dans ce contexte, et pour répondre aux nouveaux défis posés par les coûts et la tolérance aux fautes, de nombreuses méthodes et techniques basées sur des modèles dynamiques ont été développées et analysées de manière analytique. Ces méthodes de synthèse de la CTF sont généralement

classées en deux grandes familles ; la CTF passive et la CTF active. La distinction entre ces deux catégories dépend de la méthode de synthèse, des défauts considérés, du type de redondance et du comportement du système en cas de dégradation.

Une CTF passive (Stefanovski 2018),(Badihi et Zhang 2017) permet l'utilisation d'un seul contrôleur à la fois pour le comportement normal et celui fautif, elle est étroitement liée à un contrôle robuste. Même si d'un point de vue conceptuel l'idée des approches passives est attirante, ce type de solution peut imposer des limitations au niveau du comportement normal en boucle fermée. Cependant, une CTF active (Lan et Patton 2016),(Gao et al. 2017), (Wang et al. 2016)) utilise un contrôleur qui adapte sa loi de contrôle suite à l'occurrence d'un événement fautif. Ce type de stratégie permet de concevoir un contrôleur pour le comportement normal et un second contrôleur pour le comportement fautif. L'objectif principal est de passer d'un contrôleur à l'autre en fonction de l'apparition des défauts. Une étude comparative entre les méthodes CTF actives et passives est présentée dans (Jiang et Yu 2012) et (Blanke et al. 2016).

Les travaux de recherche présentés jusqu'ici portent sur des systèmes continus représentés par des équations différentielles. Cependant, la majorité des systèmes automatisés de production sont généralement contrôlés par des règles opérationnelles pouvant être modélisées par des systèmes à événements discrets. Contrairement aux systèmes continus, un SED est défini comme un système dynamique pouvant être décrit dans un espace d'états discret et dont l'évolution est décrite par des transitions d'état déclenchées par des événements (Cassandras et Lafortune 2008).

Plusieurs approches ont été développées pour cette classe de systèmes. Dans (Paoli et al. 2011) (Paoli et al. 2008), les auteurs ont proposé une approche sur la CTF active basée sur trois notions (figure 36) : la *première* consiste à exploiter une architecture à plusieurs superviseurs pour réagir activement aux effets des défauts, la *seconde* se base sur l'évaluation de l'effet de l'algorithme de diagnostic sur les performances de cette architecture, et la *dernière* consiste à définir un nouveau diagnostiqueur appelé contrôleur de diagnostic, qui réalise la commutation entre les différents superviseurs.

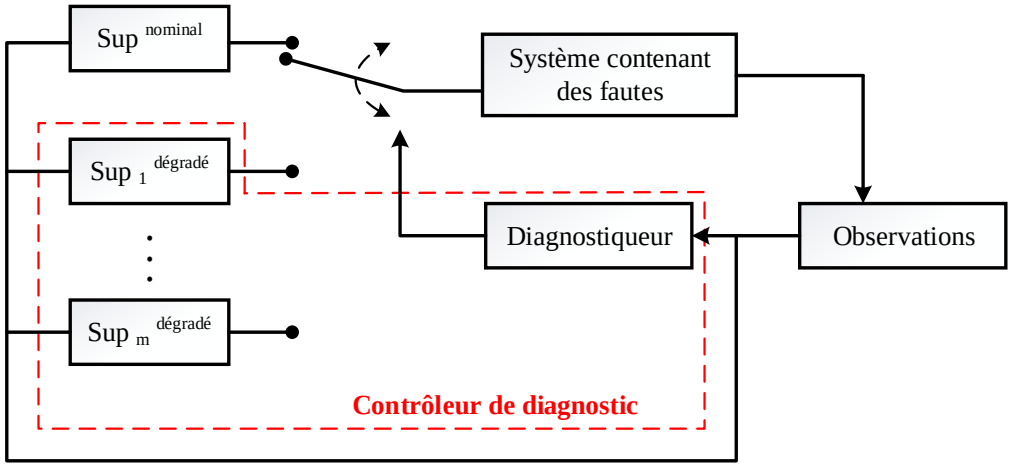


Figure 36 : Architecture de la CTF active pour les SED (inspirée de (Paoli et al. 2008))

Les auteurs traitent le problème de la CTF active dans le cadre des SED modélisés par des automates. En partant d'un modèle approprié du système, ils intègrent la notion de diagnosticabilité sûre en tant qu'étape nécessaire pour déterminer une supervision tolérante aux fautes des SED. Ensuite, ils introduisent deux nouvelles notions : la *contrôlabilité sûre*, qui représente la capacité, après l'apparition d'une faute, de détourner le système des zones

interdites et le *système de tolérance de fautes active*, qui présente la propriété permettant la continuité sûre du fonctionnement après l'apparition des défauts. En outre, les auteurs montrent la possibilité de définir une architecture de contrôle globale pour traiter le problème lié à la CTF en introduisant un type particulier d'automate dit « contrôleur de diagnostic ».

Dans (Shu et Lin 2014), les auteurs suggèrent une méthode de traitement de la CTF qui assure la sécurité d'un SED. Ils considèrent plusieurs modes fautifs. Chaque mode fautif est modélisé par un automate. Ces automates avec l'automate de mode normal décrivent un SED avec des défauts. Chaque mode fautif a des états illégaux qui doivent être évités par le contrôleur afin que le défaut puisse être toléré. Dans leur travail, ils supposent que la CTF entreprend des actions (désactivations) uniquement lorsque l'occurrence d'une faute est certaine. Ils considèrent également les cas d'observation à la fois complète et partielle et ils dérivent alors des conditions nécessaires et suffisantes pour l'existence d'une CTF.

Dans (Sánchez et Montoya 2006), les auteurs ont conçu une architecture de boucle de contrôle sûr basée sur le concept de la CTF et le contrôle robuste (figure 37).

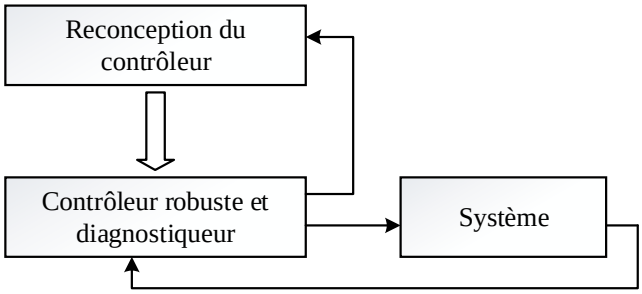


Figure 37 : Architecture de contrôle adaptée aux tolérances aux fautes

En effet, en mode de fonctionnement normal, le système fonctionne sans défaut. Dans ce mode, le superviseur doit limiter le comportement du système à un langage souhaité. En cas de perte d'observabilité de certains événements observables, le système passe en mode de fonctionnement fautif où les défauts sont non détectables. Dans ce cas, le superviseur peut autoriser le système à ne pas exécuter le comportement souhaité, mais il doit l'empêcher d'atteindre des situations (états) catastrophiques ou irrécupérables. L'architecture proposée est basée sur une modélisation modulaire du superviseur, chaque module est responsable à maintenir le système dans son comportement souhaité en mode normal. Après la perte d'observabilité sur ses événements correspondants, et avant que cette perte puisse être détectée, plusieurs modules sont aussi destinés à empêcher le système à atteindre un état indésirable. Le superviseur global est obtenu à travers la globalité des actions de contrôle de tous ces superviseurs.

Dans (Niguez et al. 2017), les auteurs intègrent la notion de temps pour répondre à la problématique de la CTF active pour des systèmes à événements discrets modélisés par des automates temporisés avec des gardes. Pour les auteurs, le temps est essentiel pour détecter certains défauts, il est utilisé comme critère pour sélectionner la loi de commande. L'approche consiste dans un premier pas à concevoir le modèle représentant le comportement de l'ensemble du système tout en respectant les contraintes de temps. Ensuite, selon le résultat du diagnostic, une loi de commande reconfigurée est extraite du modèle conçu sur la base du temps d'exécution le plus rapide des tâches souhaitées.

Cette notion de temps est évoquée aussi dans (Attia et al. 2010). Les auteurs étudient le problème de contrôle des SED soumis à des contraintes temporelles strictes en utilisant une méthode formelle basée sur l'algèbre (max, +) (J. Komenda et al. 2018). La contribution principale consiste à formuler les conditions nécessaires et suffisantes pour l'existence d'une loi de contrôle à rétroaction linéaire assurant le respect des contraintes temporelles strictes.

Dans cette thèse, nous proposons une approche d'une CTF active basée sur la reconfiguration de contrôle (RC). L'objectif principal est de concevoir une nouvelle loi de contrôle basée sur la théorie du contrôle par supervision (SCT) afin de reconfigurer l'ancienne loi correspondante au comportement normal en cas de détection d'un défaut. La SCT initiée par Ramadge et Wonham (Ramadge et Wonham 1989) vise à synthétiser un superviseur garantissant que le comportement d'un système reste acceptable par rapport aux spécifications. Plusieurs approches ont été proposées dans cet axe de recherche. Par exemple dans (Faraut et al. 2010), les auteurs discutent une procédure de reconfiguration du contrôleur basée sur la SCT tout en évitant une reconception complète du contrôleur (figure 38). L'approche de reconfiguration maintient le comportement qui ne change pas par identification (1), élimine celui qui devient inutile suite à une détection de défauts (2&3), et ajoute le nouveau comportement (4&5). La SCT garantit que les spécifications restent toujours respectées lorsqu'une synthèse est réalisée. Ainsi, la reconfiguration partielle à l'aide de la SCT continue de respecter les spécifications inchangées et garantit que les nouvelles sont également respectées dans la nouvelle loi de contrôle.

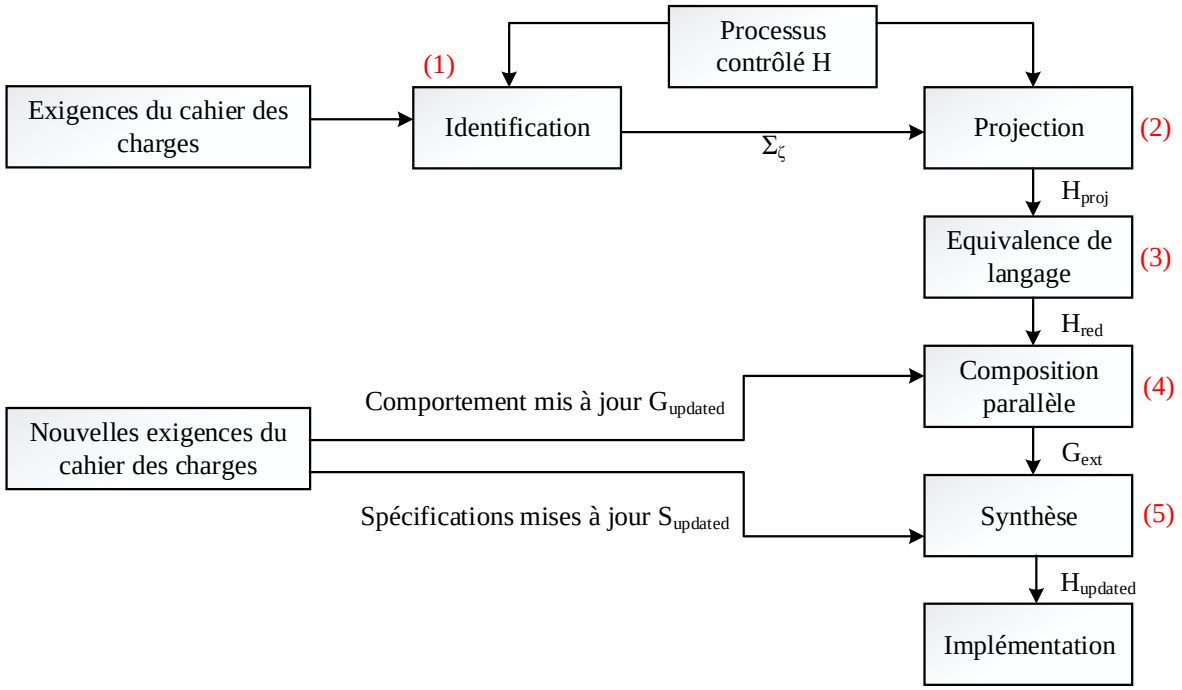


Figure 38 : Procédure proposée pour reconfigurer la loi de commande (inspirée de (Faraut et al, 2010))

Dans (Kumar et Takai 2012), un cadre de reconfiguration de contrôle des SED qui prend en compte la détection de défaut est présenté afin de reconfigurer le contrôle et garantir que certaines spécifications souhaitées sont satisfaites. En cas de détection d'un défaut, le contrôleur est reconfiguré et pendant toute la durée postérieure à la détection des défauts (ainsi qu'avant la reconfiguration), le système contrôlé doit satisfaire les nouvelles spécifications. Enfin, le

problème de contrôle est développé mathématiquement en définissant une condition de reconfigurabilité.

Dans (Sülek et Schmidt 2013), les auteurs étudient la tolérance aux fautes dans le contexte de contrôle par supervision. Ils considèrent un SED où certains événements ne sont plus possibles en cas d'apparition d'une faute. Tout d'abord, ils déterminent les conditions nécessaires et suffisantes pour l'existence d'un superviseur qui respecte un ensemble donné de spécifications en comportement normal ainsi que celui fautif. En outre, ils prouvent qu'il est possible d'identifier un sous-langage suprême tolérant aux fautes si la condition existante est violée. Enfin, ils suggèrent un algorithme de calcul de ce sous-langage et montrent qu'il est correct.

Un nouveau concept de reconfiguration du contrôle basé sur le masquage des défauts pour garantir la CTF est évoqué dans (Wittmann et al, 2013). Le principal objectif de ce concept est de masquer les défauts du contrôleur nominal alors que le système reconfiguré en boucle fermée garde un comportement acceptable. Les degrés de liberté essentiels sont établis par un bloc de reconfiguration placé entre le système fautif et le contrôleur nominal. Les auteurs tentent d'assurer un comportement complet, non conflictuel et contrôlable de la reconfiguration automatique du système en boucle fermée. Le contrôleur nominal et la modélisation du bloc de reconfiguration sont entièrement dissociés (Moor 2016).

7 Conclusion

Dans ce chapitre, nous avons défini le concept de la reconfiguration des systèmes automatisés de production, sa relation avec le temps réel et ses classifications (section 2). Nous avons ensuite défini le principe d'un système de contrôle reconfigurable ainsi que ses objectifs, ses architectures (section 3) et ses caractéristiques associés (section 4). La mise en œuvre des solutions de reconfiguration nécessite la mise au point d'une architecture de contrôle flexible afin de fournir les alternatives combinatoires au mécanisme de reconfiguration. Ce dernier doit respecter les principes basés sur l'optimalité afin de garantir l'adoption de la meilleure solution de contrôle possible. Ces deux notions de flexibilité et d'optimisation ont fait l'objet de la cinquième section comme étant les moyens assurant l'atteinte des objectifs fixés par les systèmes automatisés de production reconfigurables. Dans la section 6, nous avons présenté des travaux de recherche existants dans la littérature sur la reconfiguration et la tolérance aux fautes.

Le chapitre suivant est consacré à la proposition d'une approche méthodologique, utilisable dans l'industrie, pour la conception d'une commande reconfigurable pour les systèmes automatisés de production.

CHAPITRE III

Approche méthodologique pour la construction de la commande dans un objectif d'implantation dans un API

1 Introduction

Les deux premiers chapitres ont présenté différents aspects liés aux systèmes automatisés de production, à la sûreté de fonctionnement, aux architectures de conception de la commande et à la reconfiguration de la commande. Les études établies tout au long de ces deux chapitres, nous ont permis d'enrichir nos hypothèses de travail pour en finalité, élaborer une démarche de reconfiguration respectant les exigences suivantes :

- Type de système envisagé : Système automatisé de production (SAP)
- Type de reconfiguration retenue : Les SAP nécessitent des dispositifs, processus et/ou protocoles redondants à des niveaux spécifiques pour garantir la continuité du fonctionnement. Dans le cadre de la méthodologie de conception de la commande reconfigurable proposée, cette redondance est assurée au niveau du contrôleur (reconfiguration de la commande). Le mécanisme de reconfiguration sera inclus dans le contrôle pour garantir la continuité de l'exécution des ordres de contrôle.
- Elément déclencheur de la reconfiguration : La détection de défauts
- Type de commande retenue : L'architecture de contrôle reconfigurable est élaborée avec la capacité de changer la structure et le comportement en ligne afin d'adopter une solution de contrôle capable de répondre aux perturbations et de minimiser la dégradation de l'efficacité et de la réactivité. Il faut que la commande soit sûre de fonctionnement par construction.
- Architecture de commande retenue : Pour fournir différentes solutions au problème de contrôle global, une architecture de contrôle flexible est nécessaire. Le mécanisme de reconfiguration utilise alors cette flexibilité pour évaluer les différents objectifs de contrôle afin d'atteindre certains objectifs d'efficacité. Elle sera précisée avec le cadre de travail.
- Outil de conception et modélisation de la commande : Systèmes à événements discrets et machines à états finis.
- Réalisation du contrôle/commande : Implémentation dans un automate programmable industrielle
- Outil d'interprétation des machines à états finis pour implémentation : Grafcet
- Outil de vérification formelle des Grafcet à implémenter : Model-checking
- Outil de test et simulation de la commande réalisée : Jumeau numérique

Pour répondre à cet ensemble d'exigences, nous avons choisi, pour l'élaboration de la commande reconfigurable, de travailler dans le cadre de la théorie de contrôle par supervision (SCT). Une méthode à base de modèles permettant ainsi la prise en compte a priori des propriétés attendues du système pour synthétiser l'ensemble des comportements admissibles tout en se basant sur des modélisations de la partie opérative et des spécifications.

Pour détailler les étapes de notre méthodologie d'élaboration d'une commande reconfigurable et tolérante aux fautes, nous avons adopté l'outil SADT comme outil d'analyse fonctionnelle. Le premier niveau (A-0) regroupe toutes les hypothèses présentées ci-dessus pour répondre à notre objectif (figure 39).

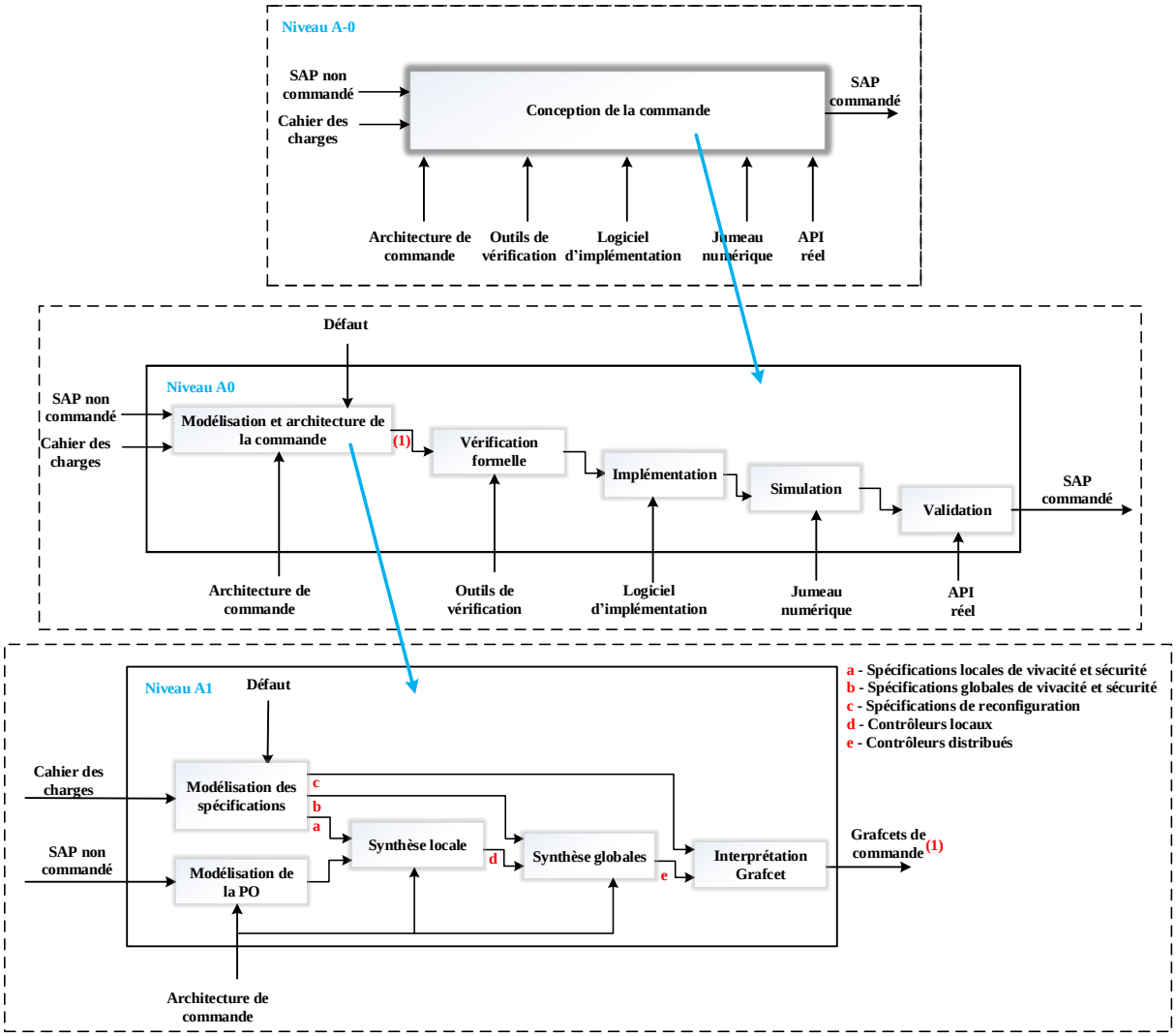


Figure 39 : Diagramme de SADT pour l'analyse fonctionnelle de la reconfiguration de la commande des SAP

Le deuxième niveau du diagramme SADT (niveau A0) présente les étapes majeures de l'élaboration de la commande reconfigurable et tolérante aux fautes pour les SAP. Ces étapes sont définies comme suit : Modélisation et conception de la commande (étape 1), vérification formelle de la commande à l'aide des outils de vérification « model-checker » (étape 2), implémentation de la commande élaborée à l'aide des logiciels d'implémentation (étape 3) tests de simulation sur un jumeau numérique (étape 4) et enfin validation de la commande sur un système réel (étape5).

Le troisième niveau du diagramme SADT (niveau A1) présente une vue détaillée de l'étape 1 du niveau (A-0). La conception de la commande reconfigurable et tolérante aux fautes est établie à partir de :

- Un modèle de comportement de la PO
- La prise en compte de défauts
- Un ensemble de spécifications de sécurité (ce que le système ne doit pas faire), de vivacité (ce que le système doit faire) et de reconfiguration (passage du mode normal au mode dégradé).
- Une synthèse de contrôleur locale
- Une synthèse de contrôleur globale
- Une interprétation du contrôleur en spécification Grafcet

Cette méthodologie peut être suivie quelle que soit l'architecture de contrôle pour assurer une reconfiguration suite à un défaut tolérable.

Dans la suite de ce chapitre, nous présentons dans un premier temps (section 2) le principe de la SCT et les problèmes associés aux approches de commande basées sur la SCT. Ensuite, nous développons notre méthodologie de construction d'une commande reconfigurable et tolérante aux fautes (section 3 et section 4). Les différentes étapes de l'approche sont illustrées autour d'un exemple didactique simple en section 5.

2 Vers une synthèse distribuée pour la reconfiguration de la commande

La SCT (figure 40), comme introduit au chapitre 2 vise à synthétiser un superviseur (SUP_G) qui assure que le comportement d'un système G donné reste dans le comportement désiré défini par un ensemble de spécifications K. Le modèle du comportement de la partie opérative du système non contrôlé (PO_G) représente l'abstraction formelle du système et indique la relation entre un signal d'entrée et le signal de sortie sans rétroaction, généralement déterminée par les propriétés physiques du système sous forme d'un automate à états finis. Afin de contraindre le comportement libre du système, il est nécessaire de définir des spécifications qui sont également modélisées par un automate à états finis.

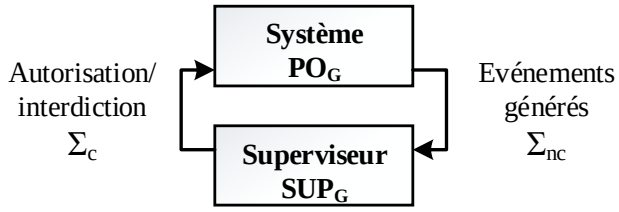


Figure 40 : Schéma de contrôle centralisé par supervision

Les algorithmes de synthèse génèrent automatiquement le superviseur le plus permissif permettant de restreindre le comportement du système tout en respectant 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 (SUP_G) 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. Un comportement est dit alors contrôlable ou commandable

(Kumar et al. 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 l'ensemble des spécifications K.

L'explosion de l'espace d'états est considérée comme l'un des inconvénients majeurs de l'utilisation de la SCT classique. En général, si le système n'est pas complexe, l'approche de contrôle centralisée de la SCT est suffisante pour synthétiser un superviseur unique. Cependant, pour modéliser un système de grande dimension constitué de plusieurs sous-systèmes, cette approche ne peut pas surmonter le problème lié à l'explosion de l'espace d'états. En effet, le nombre total d'états du modèle du système augmente en fonction du nombre d'éléments de partie opérative (EPO) le constituant (Zaytoon et Riera 2017) (Wonham et al. 2018a). L'espace d'états est défini par une dimension 2^N si le nombre des variables constituant le système est défini par N. Cependant, pour la modélisation d'un système automatisé de production, il faut associer deux événements à chaque variable (capteur/actionneur). Ces deux événements correspondent aux fronts montants et descendants de chaque variable. Par conséquent, la dimension de l'espace d'états s'élève à 2^{2N}, ce qui conduit à une explosion combinatoire importante et également à des problèmes d'interprétation des modèles lorsque les systèmes à étudier sont de grande dimension (constitué de plusieurs éléments de PO).

L'approche centralisée (figure 41-a) nécessite d'élaborer un unique automate décrivant le comportement du système ainsi qu'un unique automate pour les spécifications (Roussel et Giua 2005). Il s'avère que la tâche est difficile mais les résultats de la synthèse dépendent fortement de la qualité de ces spécifications proposées par le concepteur. Par conséquent, pour pallier les problèmes de modélisation, une démarche raffinée (Fraikin, Frappier, et St-Denis 2014) de la synthèse de superviseur a été proposée et elle est illustrée en figure (41-b). Elle consiste à construire la partie opérative du système à partir de plusieurs éléments de partie opérative appelés EPO_i. Pour obtenir le superviseur local (SUP_i) correspondant à chaque EPO_i, un ensemble de spécifications locales ($Spec^{Li}$) est défini. Le superviseur (SUPG) de fonctionnement global (centralisé) est alors obtenu par l'application de la synthèse entre les différents superviseurs locaux SUPi et les modèles des spécifications globales ($Spec^{NLi}$).

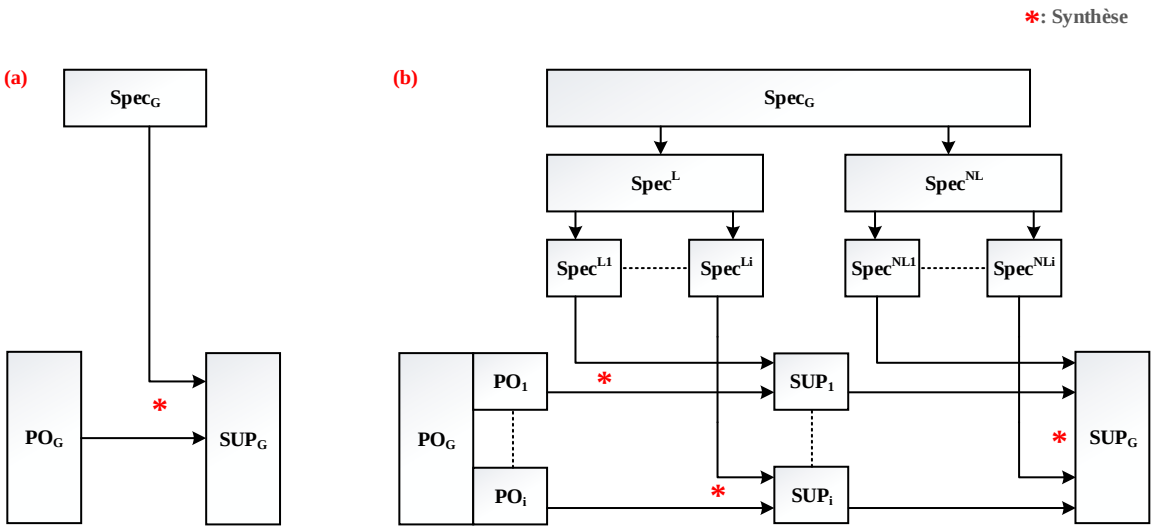


Figure 41 : Méthodes d'obtention d'un superviseur centralisé : (a) par synthèse de R&W, (b) par raffinement

Si cette démarche raffinée règle en partie les problèmes de modélisation du comportement de la partie opérative et des spécifications, elle ne règle pas les problèmes de complexité. En effet, la complexité (Ziller et Schneider 2003) du superviseur (SUP_G) est définie par $O(|Q|^2|\Sigma|) = O((|Q_P|.|Q_E|)^2|\Sigma|)$, avec Q_P est l'ensemble d'états du système, Q_E est l'ensemble d'états des spécifications et Σ l'ensemble des événements. Par conséquent, la synthèse centralisée pour un système complexe conduit à une explosion de l'espace d'états du superviseur (Wonham, Cai, et Rudie 2018).

Dans l'objectif d'effectuer de la reconfiguration, il faut tenir compte, dans la modélisation, des deux modes de comportement de la partie opérative : le mode de comportement normal PO_N et le mode de comportement prenant en compte l'apparition des défauts PO_F. La figure 42 illustre les différents chemins pour obtenir le modèle global de la PO du système (PO_{GNF}) à partir des modèles des éléments de partie opérative EPO_{Ni} et EPO_{Fi}.

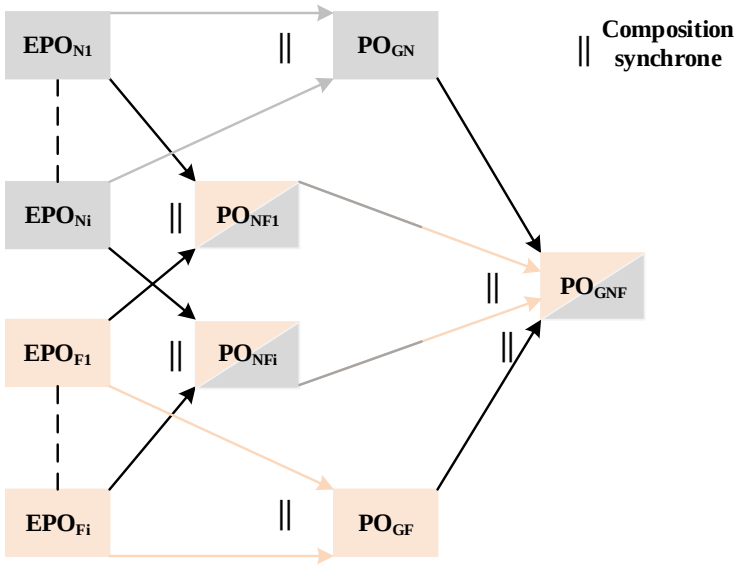


Figure 42 : Modélisation de la PO d'un SAP

Par conséquent, en ajoutant les événements associés aux défauts, la complexité du modèle global de la partie opérative PO_{GNF} se définit par $O(|Q_P|\ |Q_F|)$, avec Q_P le produit de l'ensemble des états constituant les EPO_{Ni} et Q_F le produit des états constituant les EPO_{Fi}. La complexité de calcul du superviseur centralisé s'exprime donc par $O((|Q_P|.|Q_E|.|Q_F|)^2\ |\Sigma|\ |\Sigma_F|)$, avec Σ_F l'ensemble des événements défectueux.

Pour pallier ces problèmes d'explosion d'espace d'états et de complexité de modélisation, des approches décentralisées et distribuées de la synthèse de superviseur ont été proposées. Ces approches partent du principe que le comportement de la PO est limité par plusieurs spécifications locales. L'approche synthétise ainsi un superviseur pour appliquer chacune de ces spécifications séparément. Les superviseurs décentralisés (Lin et Wonham 1987) (figure 43) fonctionnent en coopération selon des règles de fusion de décision conjonctive (coordinateur) pour former une décision globale qui décrit le même comportement commandé que celui du superviseur centralisé ou monolithique.

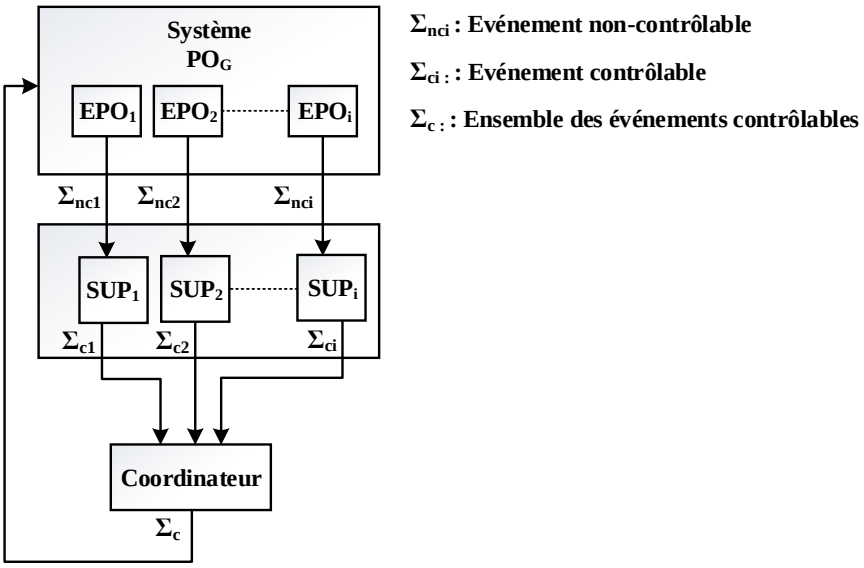


Figure 43 : Superviseurs décentralisés (inspirée de (Yoo et Lafortune 2002))

En général, les superviseurs décentralisés ignorent leurs interactions globales, le résultat peut présenter un conflit qui conduit à un blocage. Le problème est donc de coordonner efficacement les superviseurs décentralisés pour assurer un comportement non bloquant global, comme l'aurait garanti le superviseur centralisé. En d'autres termes, l'approche décentralisé nécessite une coordination globale qui peut également être impossible à calculer et à implémenter pour des systèmes de grande dimension.

Pour surmonter cette difficulté, des approches basées sur la synthèse des superviseurs distribués ont été proposées (Cai et Wonham 2010), (Jan Komenda et van Schuppen 2007). Ces superviseurs sont considérés comme étant des agents intelligents qui assurent une communication entre eux pour échanger des informations sur l'occurrence d'un événement qui est essentiel pour le contrôle (figure 44). Il est prouvé que le comportement des superviseurs distribués résultants est identique au comportement monolithique ou décentralisé présentés ci-dessus mais qui sont déterminés sous forme de modèles moins complexes (Wonham, Cai, et Rudie 2018a).

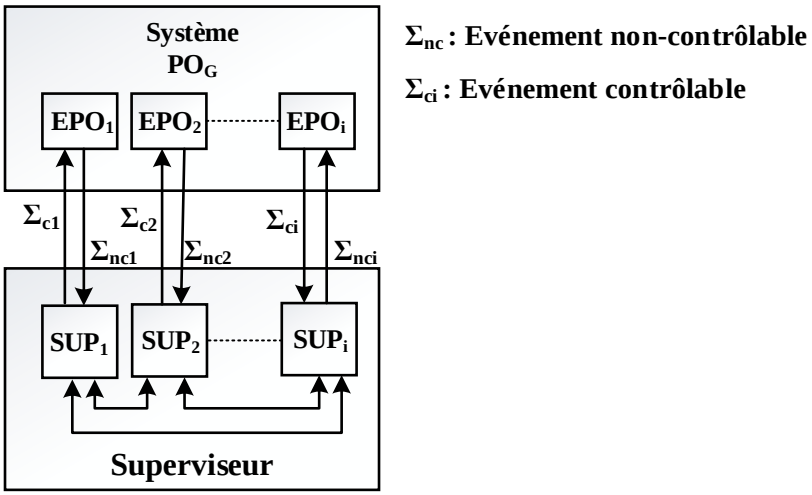


Figure 44 : Superviseurs distribués

En finalité, l'approche de reconfiguration présentée dans cette thèse va s'appuyer sur une démarche distribuée pour le contrôle des systèmes automatisés de production définis comme étant des SED. Le contrôle distribué va être assuré par plusieurs *contrôleurs* qui contrairement au superviseur, restreignent davantage le comportement du système en intégrant non seulement ce que le système ne doit pas faire (c'est la notion de superviseur) mais aussi ce que le système doit faire (c'est la notion de contrôleur). Par conséquent, les modèles des contrôleurs résultants seront moins complexes que ceux des superviseurs. Chaque contrôleur distribué est associé à un sous-système constituant le système global. En plus de son ensemble d'événements observés reçu directement de la partie opérative du sous-système, chaque contrôleur utilise également les événements observés reçus d'autres contrôleurs. La motivation derrière l'utilisation de la synthèse de contrôle distribué des SAP réside dans les avantages qu'elle présente. En effet, la commande distribuée d'un SAP se caractérise par sa flexibilité (Kapitanov 2017),(Lafou et al. 2016), (Lucas Silva, Ribeiro, et Teixeira 2017) et sa facilité à être modifiée et configurée (Nooruldeen et Schmidt 2015).

3 Principe de l'approche proposée pour la reconfiguration de la commande des SEDs

La démarche de reconfiguration peut être déclenchée par des événements liés aux produits ou par des événements liés 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 dans la production en cours. De plus, le changement d'état d'une ressource de production est caractérisé par deux événements que sont 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 étant d'utiliser des redondances matérielles pour remplacer l'élément défectueux. L'implémentation d'une démarche de reconfiguration dépend alors de l'événement de déclenchement et des contraintes de temps qui s'exercent sur le système lorsque cet événement se produit.

Deux situations complémentaires peuvent être envisagées : i) le lancement d'une nouvelle production lorsque le système est dans une situation d'arrêt et ii) l'occurrence d'une défaillance dans un système en cours de fonctionnement (Deschamps 2007). L'approche retenue dans ce chapitre se situe dans le deuxième cas.

Les nombreuses solutions proposées dans l'industrie et la recherche reposent sur la redondance matérielle pour remédier à la défaillance d'un composant du système, solution qui s'avère coûteuse en raison de la technologie et de la maintenance des composants des SAP. Pour éviter cette redondance, nous proposons dans ce chapitre, une approche de commande distribuée qui soit reconfigurable, basée sur une information temporisée des SED (figure 45). L'idée est de concevoir une commande reconfigurable capable en cas de détection d'un défaut de type capteur, d'adapter et d'exploiter les services encore disponibles offerts par la partie opérative (PO) du système. 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 dit dégradé 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. 2019a).

La boucle de reconfiguration (figure 45) de la commande repose sur trois éléments :

- Le contrôleur obtenu à partir du superviseur élaboré dans le cadre de la théorie de contrôle par supervision (SCT).
- Le diagnostiqueur qui consiste à détecter les défauts. Le diagnostiqueur n'est pas abordé dans cette thèse. Quelques travaux de recherche connexes sont présentés dans (A. Philippot et Carré-Ménétrier 2011) (Blanke et al. 2016), (Hélouët et al. 2014). Nous avons restreint notre approche au cas des défauts de capteur non observables qui sont définis par un capteur bloqué à 0 ou à 1.
- Le bloc de reconfiguration qui consiste à prendre la décision de passer du contrôleur de comportement normal au contrôleur tolérant aux fautes.

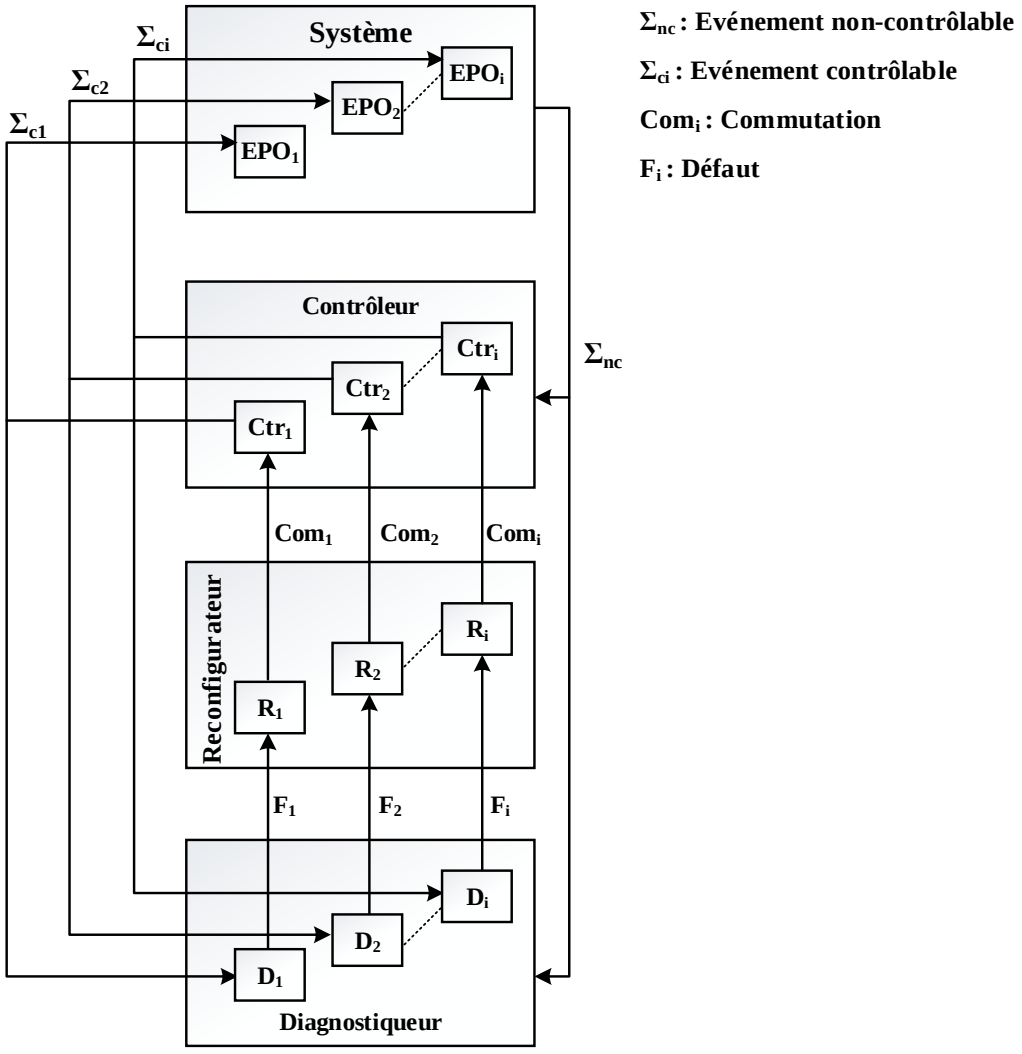


Figure 45 : Boucle de la reconfiguration distribuée de la commande

Le contrôle et la reconfiguration distribués des SAP apparaît comme une approche pour gérer la complexité de la synthèse des contrôleurs pour les systèmes à grande dimension. Nous proposons alors dans ce qui suit, d'expliciter notre méthodologie de reconfiguration de la commande. Le schéma proposé pour l'approche de la reconfiguration distribuée de la commande (figure 46) comporte sept étapes :

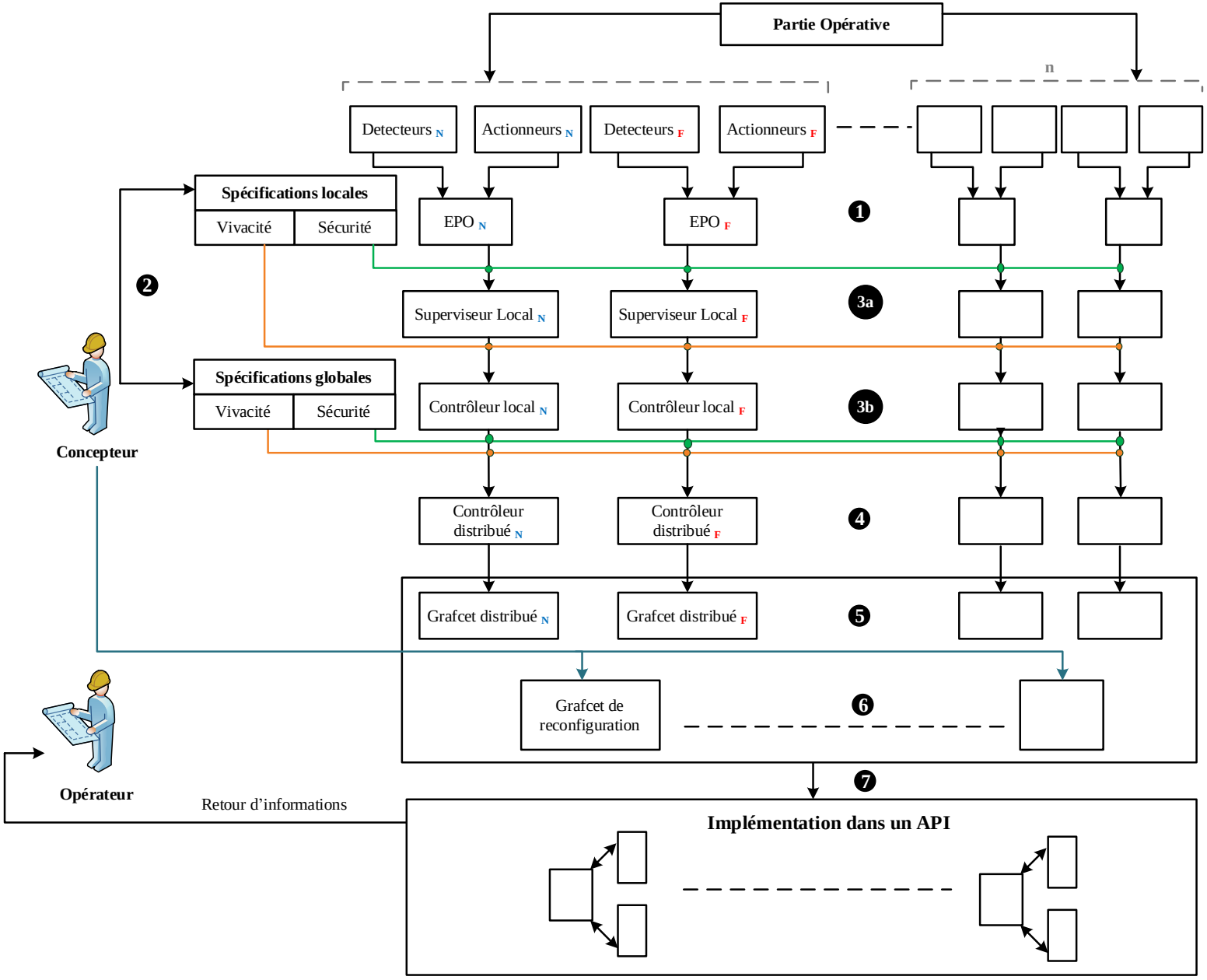


Figure 46 : Architecture de la reconfiguration de la commande distribuée

❶ Décomposition de la PO en plusieurs éléments de la PO (n×EPO) et modélisation pour chaque EPO du comportement normal et son comportement équivalent

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 dégradé de chaque EPO

❼ Implémentation dans un API dans les langages de programmation (Norme IEC IEC61131-3)

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 dégradé), ...

Avant d'adopter l'architecture distribuée, nous l'avons comparé avec deux approches centralisées présentées dans les figures 47 et 48. La première (figure 47) consiste à modéliser la PO d'une manière centralisée dès le départ en prenant en compte les deux modes de fonctionnement pour arriver à un contrôleur global tolérant aux fautes et reconfigurable. Tandis que la deuxième (figure 48) se base sur une application d'une synthèse raffinée qui consiste à séparer les deux modes de fonctionnement pour faciliter les tâches de modélisation, et les synchroniser à la fin pour obtenir le contrôleur global tolérant aux fautes et reconfigurable (les résultats de l'étude comparative peuvent être trouvés dans la section 5 de ce chapitre).

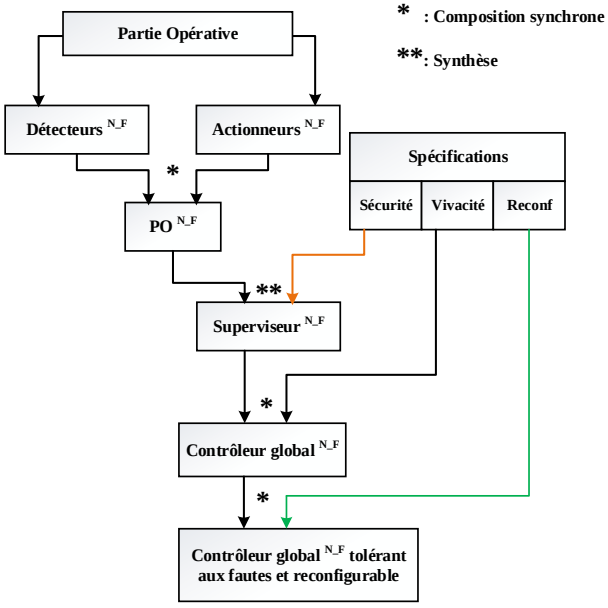


Figure 47 : Première approche centralisée développée basée sur l'approche classique de la SCT

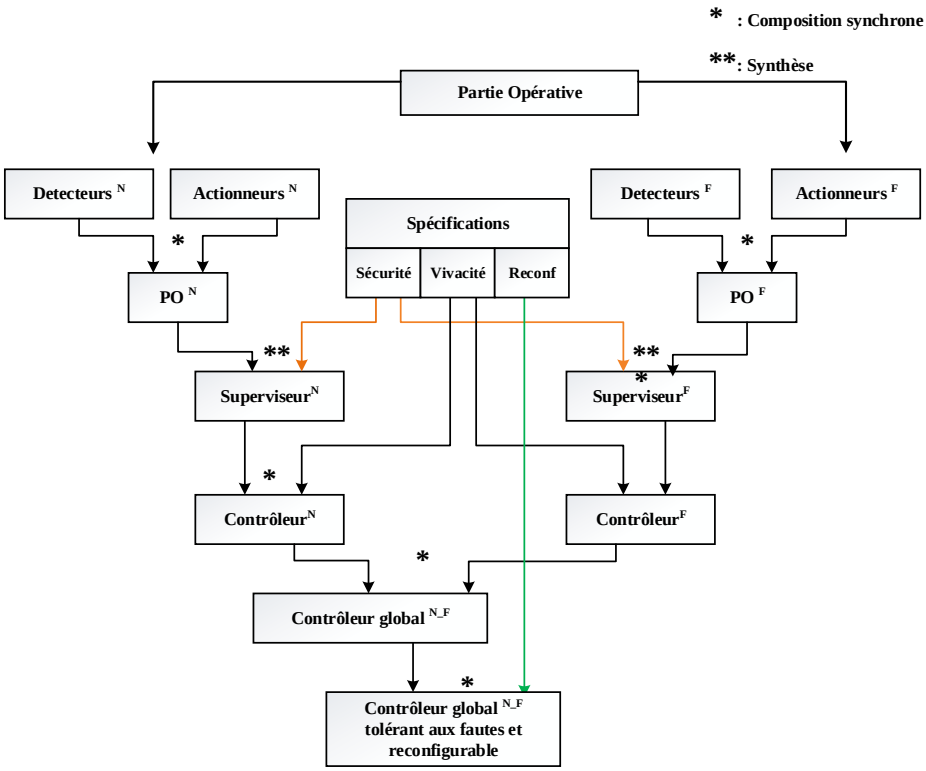


Figure 48 : Deuxième approche centralisée développée basée sur la synthèse raffinée de la SCT

L'application de ces deux approches sur un exemple didactique simple a engendré une explosion combinatoire importante et la formalisation des spécifications de la reconfiguration a été complexe nous confortant dans le choix d'une architecture distribuée détaillée dans la figure 46.

4 Formalisation de l'approche proposée pour la reconfiguration de la commande des SED

Dans cette section, nous proposons de détailler les différentes étapes de la contribution proposée dans la figure 46.

4.1 Modélisation

La modélisation d'un SAP repose sur sa description structurelle. Le modèle à concevoir est plus ou moins détaillé et dépend de l'outil de modélisation et du but pour lequel le système est étudié. Etant donné que nous étudions la reconfiguration de la commande suite à une détection de défauts au niveau des capteurs, il est nécessaire de modéliser tous les événements élémentaires associés aux éléments composant le système. Dans cette section, nous présentons la démarche préconisée pour modéliser les deux comportements normal et dégradé de la PO, ainsi que l'outil de modélisation utilisé.

4.1.1 Modélisation du comportement normal de la PO

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 (PC) 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 système. Ces modèles peuvent être obtenus à partir des relations entre les entrées et les sorties du système de commande.

Dans cette thèse, l'interprétation de Balemi (Balemi et al. 1993) est utilisée. L'ensemble des événements commandables Σ_c représente l'ensemble des sorties de la partie commande (les ordres à envoyer aux actionneurs), avec « $\uparrow z$ » 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 partie commande (les valeurs logiques des capteurs), avec « $\uparrow e$ » 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 est basée sur la modélisation proposée dans (Alexandre Philippot 2006). L'idée principale consiste à décomposer la partie opérative en plusieurs éléments de PO (EPO^N) 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 l'EPO est alors obtenu par la synchronisation de ces deux modèles. Les modèles pratiques de chaque EPO ne sont pas synchronisés pour réaliser une reconfiguration distribuée de la commande.

Formellement, le modèle du comportement normal d'un EPO (EPO^N) est défini par l'automate $A^N = \{Q^N, \Sigma^N, \delta^N, q_0^N, Q_m^N\}$ avec :

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 : $\delta^N (q_i^N, \sigma) = q_j^N$, σ est 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 $Q_m^N \subseteq Q^N$

L'ensemble des comportements normaux des EPO est défini par l'ensemble G^N, tel que :

$G^N = \bigcup_{i=1}^{n} A_{(i)}^N$ où *n* est le nombre des EPO constituant le SAP.

4.1.2 Vers une prise en compte de temps pour les SED

4.1.2.1 Intégration des événements temporisés dans les SED

Plusieurs extensions des SED ont été proposées pour disposer de fonctionnalités plus larges et de spécifications plus riches. Une des plus anciennes est basée sur la logique temporelle (Brandin et Wonham 1994). Un SAP est soumis à des contraintes de temps comme par exemple, l'instant de début de tâche ou la durée des tâches. Ces informations peuvent être utilisées pour calculer des lois de contrôle plus permissives. La prise en compte du temps introduit une nouvelle dimension dans la modélisation des SED et facilite le contrôle d'un nombre considérable d'applications mais peut engendrer également une complexité importante en termes d'états.

En se basant sur des travaux antérieurs (Ostroff et Wonham 1985) sur la logique temporelle, une version temporisée des SED a été introduite dans (Brandin et Wonham 1994). Les auteurs étendent la théorie de R&W en introduisant la notion d'événements forcés et de contraintes temporelles. L'écoulement du temps est modélisé par l'occurrence d'un événement particulier noté *tick*. Cette approche offre de bonnes solutions pour le contrôle des systèmes à événements discrets temporisés (SEDT). Cependant, la nature discrète du temps génère une explosion exponentielle du nombre d'états.

La démarche se base sur deux étapes pour aboutir à une modélisation du comportement d'un SEDT. Les auteurs soulignent que, dans l'interprétation, les activités ont une durée, tandis que les événements sont instantanés. La première étape consiste à décrire les états du système et les transitions qui les lient par un 5-uplet noté $A\_{act}= (Q\_{act}, \Sigma\_{act}, \delta\_{act}, q0\_{act}, Qm\_{act})$ avec :

Q_{act} : est l'ensemble fini des états de A_{act} associés à un ensemble d'activités A dont les éléments sont des activités (a_i).

Σ_{act} : est l'ensemble des événements de A_{act}

δ_{act} : est la fonction de transition partielle $\delta\_{act} : \Sigma\_{act} \times Q\_{act} \rightarrow Q\_{act}$.

$q0\_{act}$: est l'état initial de l'automate A_{act}, tel que $q0\_{act} \in Q\_{act}$

Qm_{act} : est l'ensemble des états marqués dans A_{act}, tel que $Qm\_{act} \subseteq Q\_{act}$

Soit N l'ensemble des nombres naturels {0, 1, 2, …}. Chaque événement $\sigma \in \Sigma\_{act}$ est associé à un intervalle $[l_a, u_a]$. $l_a \in N$ est la limite de temps inférieure et $u_a \in N$ est la limite de temps supérieure. Un triplet (σ, l_a, u_a) est noté événement temporel.

Un événement peut être classifié en deux catégories selon la borne limite supérieure de l'intervalle associé :

- Un événement σ est appelé prévu si $0 \leq u_a < \infty$, l'ensemble de ces événements est noté Σ_{spe}.
- Un événement σ est appelé lointain si $u_a = \infty$, l'ensemble de ces événements est noté Σ_{rem}.

Par conséquent l'ensemble des événements Σ_{act} est défini par l'union de ces deux ensembles : $\Sigma\_{act} = \Sigma\_{spe} \cup \Sigma\_{rem}$. Chaque événement σ est associé à une temporisation t_σ qui mesure le temps écoulé depuis sa dernière validation.

Pour but de modéliser des contraintes temporelles, la deuxième étape consiste à définir un nouvel automate noté $A= (Q, \Sigma, \delta, q_0, Q_m)$ tel que :

- Q : est l'ensemble des états. Chaque état $q \in Q$ mémorise un état $q\_act \in Q$ et la valeur de temporisation t_σ associée à chaque événement $\sigma \in \Sigma\_act$: $q = \{q\_act \, |\{t_\sigma| \; \sigma \in \Sigma\_act \}\}$.
- Σ : est l'ensemble des événements. L'occurrence d'un événement noté tick modélise le passage du temps, $\Sigma = \Sigma\_act \cup \{tick\}$.
- δ : est la fonction de transition. Un événement σ peut être exécuté depuis un état q, i.e. $\delta(q,\sigma)$!, si δ_act (q_act, σ) !, et la contrainte temporelle associée à l'occurrence de σ est vérifiée.
- q_0 : est l'état initial
- Q_m : est l'ensemble des états marqués.

Un événement σ est considéré valide depuis un état q si δ_{act} (q_{act}, σ) existe. Il devient éligible, donc il peut être exécuté, lorsque la contrainte temporelle associée à sa date d'occurrence est vérifiée.

La prise en compte du temps rend le modèle étudié plus intéressant, mais en contrepartie, augmente sa complexité en termes de nombre d'états. L'exemple suivant illustre comment l'intégration du temps influence fortement la complexité du modèle obtenu (Brandin et Wonham 1994).

Soit un SED modélisé par l'automate $A\_{act} = (Q\_{act}, \Sigma\_{act}, \delta\_{act}, q_{0}\_{act}, Q_{m}\_{act})$ présenté dans la figure (49. a), tel que :

- $Q\_{act} = \{q_0\}$.
- $\Sigma\_{act} = \{ \alpha, \beta\}$, les deux événements temporels sont (α, 1, 1) et (β, 2, 3)
- $\delta\_{act}(q_0, \alpha) = \delta\_act\ (q_0, \beta) = 0$
- $q_{0}\_{act} = \{q_0\}$
- $Q_{m}\_{act} = \{q_0\}$

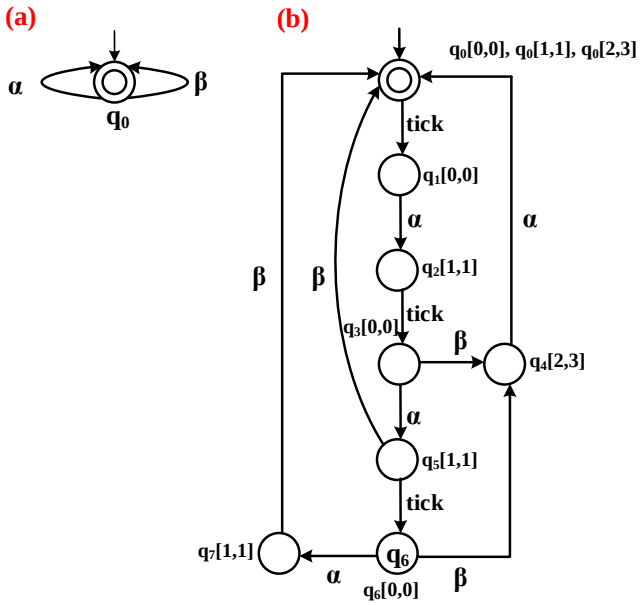


Figure 49 : Modélisation des automates : (a) A_{act} et (b) A

Le comportement temporel du système est modélisé par l'automate $A= (Q, \Sigma, \delta, q_0, Q_m)$ présenté dans la figure (49. b) avec :

- $Q = \{q_0\} \times t_\alpha \times t_\beta = \{q_0\} \times [0,1] \times [0,3]$
- $\Sigma = \{ \alpha, \beta, \text{tick}\}$
- $q_0 = \{q_0\}$
- $Q_m = \{q_0\}$

Chacun des états est caractérisé par une valeur particulière des temporisations associées aux événements. Il faut souligner que la prise en compte explicite du temps engendre l'augmentation du nombre d'états. En effet, en tenant compte du temps, le modèle du système passe d'un seul état et deux transitions à un modèle constitué de 7 états et 11 transitions comme le montre la figure (49. b) .

Afin de résoudre ce problème et contrairement à l'approche fondée sur des modèles où le temps est discret, certains auteurs proposent des approches basées sur des modèles à temps dense (Alur et Dill 1994), car il s'agit d'un modèle plus naturel pour les processus physiques fonctionnant en temps continu. Dans ce modèle, les temps des événements sont des nombres réels qui augmentent de façon monotone, sans limite. Le traitement du temps dense dans un cadre d'automates à états finis est difficile, car il n'est pas évident de transformer un ensemble

de traces à temps dense dans un langage formel ordinaire. C'est pour cette raison que la théorie des langages formels temporisés et des automates temporisés a été développée.

Néanmoins, nous pouvons nous poser des interogations sur la pertinance de l'utilisation de ces outils formels et mathématiques. La sémantique des modèles des automates temporisés est très précise et nécessite des transitions immédiates et des horloges infiniment précises. Par conséquent, aucune plateforme d'exécution ne permet d'implémenter aussi précisement un tel automate (Altisen et al. 2005). En outre, même si plusieurs outils ont été développés pour le model-checking temporisé (UPPAAL, …), rien ne permet d'assurer que les propriétés vérifiées sur les modèles théoriques soient préservées lors de l'implémentation. On parle ici des problèmes « *d'implémentabilité* » et de préservation des propriétés après implémentation des modèles d'automates temporisés. Soit A un automate temporisé développé dans l'hypothèse d'un environnement pour vérifier une propriété logique θ, l'automate A préserve θ si l'exécution du programme codant A après implémentation vérifie toujours θ. Ce qui n'est pas toujours vrai pour un automate temporisé, cela peut être dû à des conditions sur la plateforme d'exécution, des conditions sur le programme implémentant l'automate temporisé et/ou des conditions sur la propriété à préserver. Pour illustrer ces propos, (Altisen et al. 2005) étudient l'exemple d'un système d'exclusion mutuelle (Thiare 2007) de Fischer (Kristoffersen et al. 1997). Deux systèmes S_1 et S_2 peuvent accéder à une section critique, ils sont supervisés par deux contrôleurs assurant que ces deux systèmes n'accèdent pas simultanément à la section critique. Chaque système est composé de n processus P_i concurrents, chaque processus ayant un numéro unique entre 1 et n. L'algorithme utilise une variable partagée id (id ∈ [0,n]) dont la valeur indique quel processus a demandé l'accès à la section critique. Si la ressource est libre (id= 0), aucun processus demande l'accès. Quand un processus de numéro i, veut entrer dans la section critique, il teste la valeur de la variable id. Uniquement si id=0, il attend au plus 2 unités de temps avant d'affecter à i la variable id. Puis, il attend au moins 2 unités de temps avant de tester à nouveau la valeur de id. Si id est toujours égale à i, alors le processus entre dans la section critique. Sinon, il réessaye d'acquérir la section critique en testant à nouveau la valeur de id. Chaque processus peut être alors dans quatre cas différents :

- RP : état de repos où le processus ne cherche pas à obtenir la ressource
- Requête : état où le processus débute une session de demande de la ressource
- Attente : état où le processus prend une option sur la ressource et attend un délai de 2 unités de temps
- SC : état où le processus a obtenu la ressource (il est en section critique)

Le système S_i et son contrôleur C_i sont illustrés dans la figure 50.

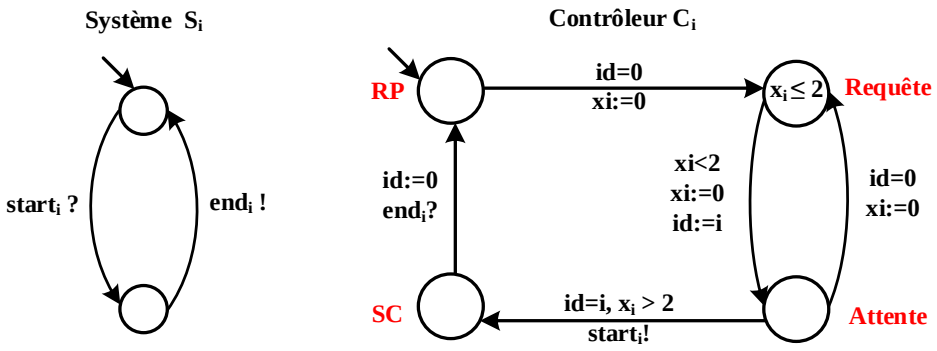


Figure 50 : Système d'exclusion mutuelle de Fischer

Les auteurs montrent que la propriété d'exclusion mutuelle de l'exemple décrit ci-dessus est étroitement liée au caractère très précis du modèle mathématique des automates temporisés qui le constituent. En effet, la propriété d'exclusion mutuelle repose sur la garde stricte $x_i > 2$. Combinée avec l'invariant $x_i \leq 2$ de l'état *Requête,* elle assure que lorsqu'un processus P_i peut atteindre sa section critique depuis l'état *Attente,* aucun autre processus P_j ne peut être dans l'état *Requête*. Ainsi, le relâchement de la contrainte $x_i > 2$ en $x_i \geq 2$ fait perdre la correction du système et démontre sa fragilité. Si la précision des horloges est finie ou, si elles ne sont pas parfaitement synchronisées ou, si des délais de communication existent entre les processus, ou si des délais de lecture/écriture des variables partagées ralentissent le système, il est alors possible que les transitions soient franchies à des instants interdits dans le modèle mathématique. Ce sont des contraintes ou des propriétés qui se rencontrent dans tout système réel et qui peuvent en finalité entraîner la violation de la propriété d'exclusion mutuelle. L'étude d'un tel système a conduit les auteurs à mettre en évidence plusieurs caractères liés aux modèles dit « idéals » des automates temporisés et qui ne sont pas reproductibles lors d'une implémentation dans un système réel tels que : (i) le temps d'exécution nul des actions (vitesse infinie du système lors de l'évaluation des gardes des transitions, et du choix de l'action à déclencher), et (ii) la synchronisation parfaite des différentes composantes du système (réception en temps nul des entrées, émission en temps nul des sorties).

Pour toutes ces raisons notre choix s'est orienté vers les automates à états finis tout en intégrant la notion du temps.

4.1.2.2 Modélisation du comportement temporisé de la PO

Dans le cadre de nos travaux, 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. Pour cela, nous proposons une méthode d'intégration du temps en se basant sur les automates à états finis comme étant un outil de modélisation tout en évitant la complexité des modèles récurrente dans (Brandin et Wonham 1994). Le modèle temporisé est une extension de l'automate du comportement normal déterminé dans la sous-section (4.1.1). 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.

Le modèle de comportement normal 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 équivalent au comportement normal du système.

Par ailleurs, nous avons proposé une méthode permettant d'inclure le temps pour les SED (Tahiri et al. 2019). 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 automates à états finis. La temporisation définit le temps nécessaire pour l'activation ou la désactivation d'un capteur après la vérification d'une condition spécifique.

Le modèle du comportement équivalent au comportement normal d'un EPO (EPO^F) est alors obtenu par la synchronisation des deux modèles temporisés des détecteurs (détecteurs_F) et des actionneurs (actionneurs_F).

Formellement, le modèle (EPO^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 $\Sigma^F = \Sigma_{nT}^F \cup \Sigma_T^F$, avec Σ_T^F est l'ensemble des événements temporisés tel que : $\Sigma_T^F = \uparrow ck_k \cup \downarrow ck_k \cup d_k$ avec $\uparrow\downarrow ck_k$ est l'événement associé à l'activation et la désactivation de l'horloge et d_k est l'événement associé à la durée de l'horloge (d_k==1 si la durée est écoulée)
- δ^F : est la fonction de transition. Une transition est définie par : $\delta^F (q_1^F, \sigma) = q_2^F$. 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 51.

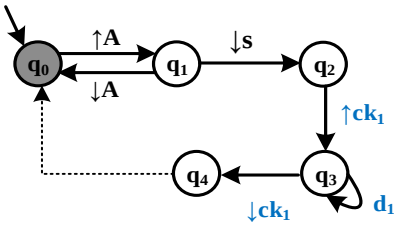


Figure 51 : Extrait d'un modèle d'un automate A^F

Le passage à l'état q_2 permet l'enclenchement de l'horloge 1 ($\uparrow ck_1$), l'apparition de cet événement permet le passage à l'état q_3. Le système reste à cet état pendant une durée prédéterminée, son écoulement active sa garde associée (d_1==1), par conséquent un passage à l'état q_4 est possible en désactivant l'horloge ($\downarrow ck_1$).

Le modèle équivalent de l'activation ou la désactivation d'un capteur est donné alors sous forme d'un modèle composé de trois transitions et un seul état comme le montre la figure 52.

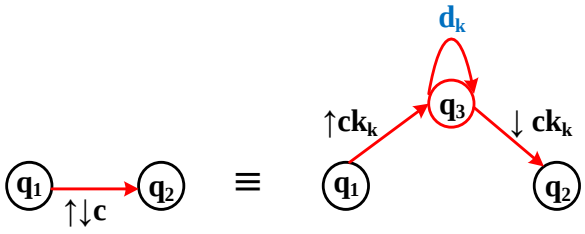


Figure 52 : Modèle équivalent de l'activation d'un capteur c1

Pour chaque capteur détecté fautif, un seul état est ajouté par rapport au modèle du comportement normal de la PO. Par conséquent le modèle résultant ne présente pas d'explosion combinatoire. Les durées associées aux gardes d_k sont définies suite à un apprentissage de l'ensemble des capteurs du système étudié. Cette phase permet de déterminer les durées nécessaires pour le fonctionnement des capteurs (activation et désactivation).

L’ensemble des comportements équivalents aux comportements normaux (ou dégradés) des EPO est défini par l’ensemble G^F, tel que :

$G^F = \bigcup_{i=1}^{n} A_{(i)}^F$ où *n* est le nombre des EPO constituant le SAP.

L'ensemble des modèles des EPO décrivant le système global est alors défini par : $G= G^N \cup G^F$.

4.2 Synthèse locale des contrôleurs

Le but d'une synthèse de contrôleur local est d'établir des contrôleurs qui peuvent décrire le comportement local souhaité de chaque EPO. Dans ce but, des spécifications locales doivent être définies pour restreindre chaque comportement des EPO aux comportements locaux souhaités.

4.2.1 Définition des spécifications

Dans l’approche classique de synthèse de la commande des SED (Ramadge et Wonham 1989), les contraintes s’expriment par des automates à états finis. Cependant, ceci peut généralement poser des problèmes de conception et d’interprétation des modèles. Pour remédier à ces problèmes, nous adoptons pour l’intégration de la commande, une démarche à base d’équations booléennes logiques consistant à contraindre chaque EPO jusqu’à son fonctionnement désiré. L’écriture des contraintes par équations logiques facilite la représentation des séquences d’événements et permet de réduire l’explosion combinatoire inhérente à l’utilisation des automates à états lors de la génération de la commande.

Il existe quelques travaux dans la littérature qui utilisent la modélisation des contraintes par équations logiques dans l’algèbre de Boole dans le contexte de la synthèse de la commande. Dans (Tajer, Philippot, et Carré-Ménétrier 2013), le cadre formel d’une modélisation des contraintes logiques dans l’algèbre de Boole a été développé. Cette modélisation est basée sur des propositions où les éléments manipulés sont les valeurs binaires 0 ou 1 représentant les valeurs des entrées et des sorties de la PC.

Dans (Riera et al. 2014), les auteurs ont proposé une approche de filtre de sécurité afin de garantir la sécurité quel que soit le programme de contrôle. Ce filtre de sécurité est basé sur un ensemble d’équations booléennes qui définissent les contraintes de sécurité. Pour un ensemble donné de variables d'entrée et pour la première affectation des variables de sortie, le filtre de sécurité doit trouver l'affectation la plus proche de la première qui ne viole aucune contrainte de sécurité. Une version améliorée de cette proposition est donnée dans (Pichard et al. 2019), les auteurs ne traitent pas les contraintes de vivacité/ fonctionnelles.

L’utilisation des équations logiques booléennes a pour avantage de décrire les contraintes de manière précise, modulaire et flexible, dans un langage familier de l’expert. Ceci permet de remédier au problème de la lourdeur de modélisation par un seul automate regroupant toutes les spécifications à respecter.

Les spécificités du système représentent le comportement des opérations à effectuer (ou non) par le système. Elles sont exprimées sous forme de contraintes de sécurité (ce que le système ne doit pas faire) et de vivacité (ce que le système doit faire). L'intégration des contraintes de spécifications a pour objectif d’inhiber les actions et/ou à organiser et à 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. Les contraintes peuvent être appliquées globalement à l'ensemble du système ou localement à chaque élément de partie opérative.

Les spécifications locales représentent un ensemble de contraintes de sécurité et de vivacité appliquées localement à chaque EPO. Chaque contrainte est définie par une implication logique donnée par la formule suivante :

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

« q » est un état de l'ensemble des états de G, et « y » est un événement contrôlable. L'implication ci-dessus signifie que si « q » est vrai, alors « y » est interdit.

4.2.2 Contrôleurs locaux

Un contrôleur local (CL) peut décrire le comportement local normal (CL^N) et dégradé (CL^F) de chaque EPO, sachant que le comportement temporisé (CL^F) représente ici le comportement dans le cas de détection de défaut capteur. Un contrôleur local peut fonctionner séparément des autres contrôleurs conçus. Pour obtenir les automates des contrôleurs locaux des comportements normaux et dégradés $A^{N\,(CL)}$ / $A^{F\,(CL)}$, une synthèse d'un modèle étendu de A^N et A^F avec l'ensemble de spécifications à respecter est appliquée. En effet, les automates étendus de A^N et A^F sont obtenus via une machine à états finis étendue (MEFE) (Akesson et al. 2006). Contrairement aux machines à états finis, une machine à états finis étendue utilise des gardes, des actions et des variables qui peuvent faciliter à l'opérateur une représentation compacte d'un SED de taille importante.

Les expressions des gardes sont utilisées pour limiter le comportement du système et les fonctions d'action sont utilisées pour mettre à jour les variables. Les automates résultants sont notés $A^{N}_{(ext)}$ et $A^{F}_{(ext)}$, ils contiennent les mêmes modèles que A^N et A^F avec l'ajout d'actions. Ces derniers décrivent l'état cible de chaque transition du système exprimé par les équations suivantes : ($A^N = q^{N}_{(i)}$) ou ($A^F = q^{F}_{(i)}$). Ensuite, l'ensemble des spécifications locales doit être appliqué. Contrairement aux travaux précédents (Tajer, Philippot, et Carré-Ménétrier 2013) et (Riera et al. 2015) où le cahier des charges consiste à interdire un événement contrôlable suite à un événement incontrôlable, notre méthode repose sur la vérification dans chaque état de $A^{N}_{(ext)}$ et $A^{F}_{(ext)}$ si l'événement contrôlable est autorisé ou non. Pour obtenir automatiquement les automates $A_i^{N(CL)}$ ($A_i^{F(CL)}$) (i est l'indice de l'automate correspondant à l'EPOi), nous proposons d'interpréter la spécification par un automate à états finis étendus composé d'un seul état et d'une transition rebouclante associée à l'événement contrôlable "z" et à la garde exprimée par {($A^{N\,(CL)}$ / $A^{F\,(CL)}$) ! = q}, ce qui signifie que si l'état actuel de l'automate $A^{N}_{(ext)}$ / $A^{F}_{(ext)}$ est différent de "q", alors "z" est autorisé sinon la transition associée à " z "est supprimée.

Les équations logiques des spécifications sont traduites en automates à états finis étendus pour appliquer l'algorithme de synthèse proposé par le Logiciel Supremica (Akesson et al. 2006).

En appliquant la synthèse du contrôle de supervision entre les automates présentant $A^{N}_{(ext)}$ / $A^{F}_{(ext)}$ et les spécifications, nous obtenons des automates $A^{N\,(CL)}$ / $A^{F\,(CL)}$ correspondant aux contrôleurs locaux. Ces derniers sont permissifs et respectent le comportement désiré du système, tout en garantissant le respect des spécifications.

Un exemple appliqué pour une spécification (q_2 . ↓ z = 0) est donné dans la figure 53. L'événement ↓ z n'est autorisé qu'après le front montant du capteur e_2 (état q3). En appliquant la spécification, la transition sortant de l'état q_2 est supprimée.

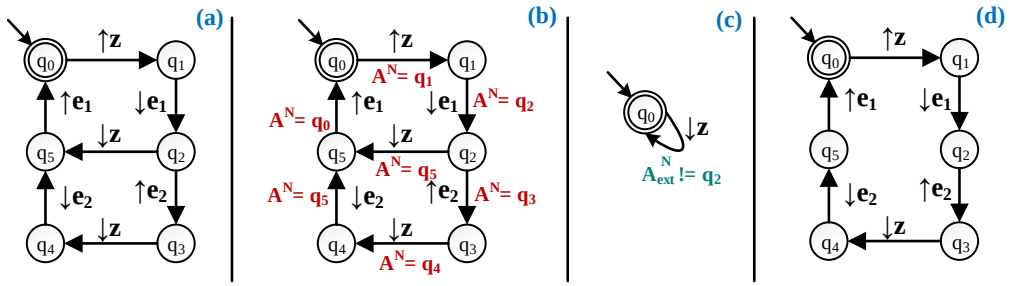


Figure 53 : Exemple d'obtention du CL : (a) automate de comportement normal A^N, (b) automate de comportement normal étendu $A^N_{(ext)}$, (c) spécification, (d) automate du contrôleur local résultant

Cette méthode de synthèse ne présente pas d'explosion combinatoire rencontrée dans les approches basées sur la modélisation monolithique des spécifications. En effet, si *n* est le nombre d'états de $A^N_{(ext)}$ et $A^F_{(ext)}$ et m_i est le nombre d'états de chaque spécification tel que i∈[1,...., j], le nombre d'états des contrôleurs locaux $A^{N\ (CL)}$ / $A^{F\ (CL)}$ est donné par $k=n\times(m_1\times\ldots\times m_j)$. Puisque les spécifications sont définies comme un automate avec un seul état, le résultat du produit de $(m_1\times\ldots\times m_j)$ est toujours égal à 1. Ce qui implique que k = n, contrairement à la SCT centralisée où $k = n \times m_i$ avec $m_i>=1$.

4.3 Synthèse globale

L'objectif de la synthèse de contrôleur globale est d'établir des contrôleurs distribués qui peuvent décrire le comportement global de la PO. Dans ce but, des spécifications globales doivent être définies pour assurer la liaison et la communication entre les différents contrôleurs.

4.3.1 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_g) alors {y = 0 sinon y = 1} **(eq2)**

Avec : cg = c+d.

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

Une condition « $c_g=c+d$ » 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_g ∈

$\uparrow\downarrow e_i$ (événements associés aux capteurs) et $/c_g \in \uparrow\downarrow d_i$ (événements associés aux durées)

4.3.2 Contrôleurs distribués

Un SAP 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 global souhaité. Par conséquent, une communication entre plusieurs EPO est nécessaire. Pour se faire, une synthèse de contrôle globale est nécessaire pour obtenir des contrôleurs distribués pour les comportements normaux et équivalents/dégradés pour chaque EPO.

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

- Première agrégation : les événements contrôlables non temporisés $\uparrow\downarrow Z$ sont fusionnés en macro-états. Dans (Qamsane et al. 2016), les auteurs associent les états atteints par les événements contrôlables dans des macro-états liés par des événements incontrôlables $\uparrow\downarrow e$. Dans notre étude, les macro-états sont également liés par des événements temporels $\{\uparrow ck_i, \downarrow ck_i, d_i\}$ en cas de comportement défectueux. Si l'état du contrôleur local CL est associé à un front montant d'un événement contrôlable, l'ordre de commande est autorisé et appartient à l'ensemble *Ord* (ensemble d'autorisations des ordres). S'il est associé à un front descendant de cet événement, alors l'ordre est inhibé et appartient à l'ensemble *Inh* (ensemble d'inhibitions des ordres). L'automate résultant est un contrôleur local agrégé (CLA) défini comme suit (figure 54) :
  $A^{N/F(CLA)} = (Q^{N/F(CLA)}, \Sigma^{N/F(CLA)}, \delta^{N/F(CLA)}, Set^{N/F(CLA)}, q_0^{N/F(CLA)}, Q_m^{N/F(CLA)})$. Avec $Set^{N/F(CLA)} = Ord^{N/F(CLA)} \cup Inh^{N/F(CLA)}$

- Deuxième agrégation : l'algorithme 1 est appliqué pour le comportement dégradé. Les événements temporisés $\uparrow\downarrow ck_i$ sont fusionnés dans des macro-états liés par des événements incontrôlables et des événements temporisés d_i (figure 54). Si l'état du contrôleur local agrégé suite à la première agrégation est atteint par un événement correspondant à l'activation de l'horloge, alors cet événement appartient à un ensemble noté A_{CK} (ensemble d'activations des horloges). S'il est atteint par un événement correspondant à la désactivation de l'horloge, alors cet événement appartient à un ensemble noté D_{CK} (ensemble des désactivations des horloges). La transition rebouclante est transformée en transition qui relie les deux macro-états qui contiennent les deux ensembles (A_{CK} et D_{CK}). L'automate résultant est le contrôleur local agrégé CLAT représentant le comportement dégradé (détection de défaut capteur) tel que :
  $A^{(CLA)T} = (Q^{(CLA)T}, \Sigma^{(CLA)T}, \delta^{(CLA)T}, Set^{(CLA)T}, q_0^{(CLA)T}, Q_m^{(CLA)T})$.
  Avec $Set^{(CLA)T} = Ord^{N/F(CLA)} \cup Inh^{N/F(CLA)} \cup A_{CK}. \cup D_{CK}$.

Algorithme 1: Agrégation du contrôleur local agrégé

Entrée :
$A^{N/F(CLA)}=(Q^{N/F(CLA)},\Sigma^{N/F(CLA)}, \delta^{N/F(CLA)}, Set^{N/F(CLA)}, q_0^{N/F(CLA)}, Q_m^{N/F(CLA)})$

Début

1. **Pour** chaque événement $\sigma \in \Sigma^{N/F(CLA)}$

---

2. $P(\sigma) = \begin{cases} \varepsilon \text{ if } \sigma \in CK \\ \sigma \text{ if } \sigma \notin CK \end{cases}$ tel que $CK = \uparrow ck_i \cup \downarrow ck_i$
3. Déterminer $A^{N/F(CLA)}$
4. Soit $A^{N/F(CLA)1} = (Q^{N/F(CLA)1}, \Sigma^{N/F(CLA)1}, \delta^{N/F(CLA)1}, Set^{N/F(CLA)1}, q_0^{N/F(CLA)1}, Q_m^{N/F(CLA)1})$ l'automate résultant après les deux opérations 2 et 3.
5. **Pour** chaque état $q^{N/F(CLA)1} \in Q^{N/F(CLA)1}$
6. **Si** $q^{N/F(CLA)1}$ fusionne plus qu'un seul état $q^{N/F(CLA)}$ **Alors**
7. **Pour** chaque transition $Tr = (q_1^{N/F(CLA)}, \sigma, q_2^{N/F(CLA)})$ entre ces deux états fusionnés
8. **Si** $\sigma \in \uparrow ck_i$ **Alors**
9. $A_{ck} \leftarrow A_{ck} \cup (ck_i)$
10. **Sinon** $\sigma \in \downarrow ck_i$ **Alors**
11. $D_{ck} \leftarrow D_{ck} \cup (ck_i)$
12. **Fin Si**
13. **Fin Pour**
14. **Fin Si**
15. **Si** $q^{N/F(CLA)1}$ contient l'activation et la désactivation de ck_i: $\{(A_{ck}: ck_i)), (D_{ck}: ck_i)\}$ **Alors**
16. $q^{N/F(CLA)1}$ est séparé en deux états liés par la transition rebouclante exprimée précédemment par $Tr = (q_1^{N/F(CLA)1}, d, q_2^{N/F(CLA)1})$ **Et**
17. $A_{ck} \in q_1^{N/F(CLA)1}$, $D_{ck} \in q_2^{N/F(CLA)1}$
18. **Fin Si**
19. **Fin Pour**
20. **Fin Pour**

Fin

Sortie : $A^{(CLA)T} = (Q^{(CLA)T}, \Sigma^{(CLA)T}, \delta^{(CLA)T}, Set^{(CLA)T}, q_0^{(CLA)T}, Q_m^{(CLA)T})$.

A titre d'illustration, la figure 54 présente un exemple d'obtention d'un contrôleur distribué par les deux agrégations définies précédemment.

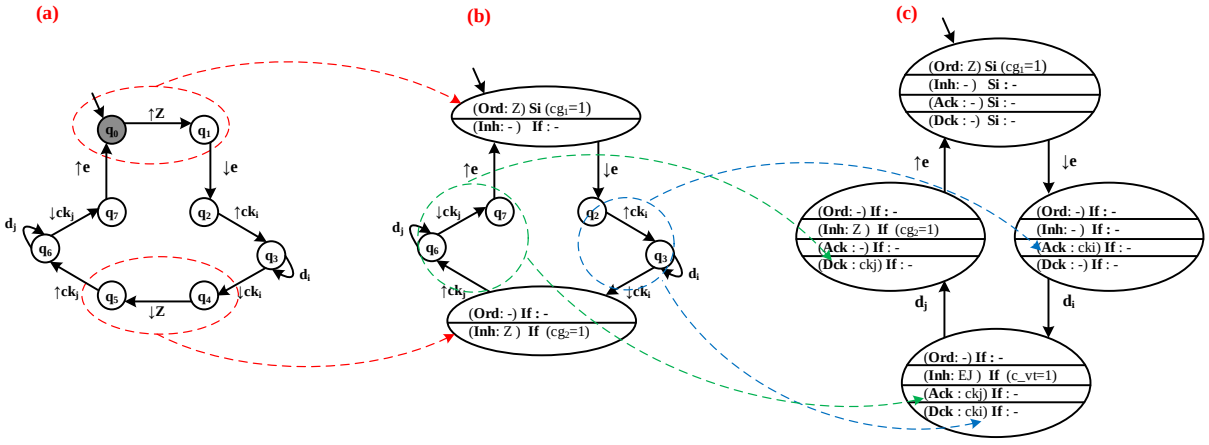


Figure 54 : Obtention du contrôleur distribué : (a) contrôleur local, (b) contrôleur local agrégé, et (c) contrôleur distribué

L'ordre A est autorisé dans l'état initial si la spécification globale cg est vraie, ce qui entraine la désactivation du capteur *s* pendant une durée d1.

4.4 Interprétation des contrôleurs distribués en grafcet

Bien que la théorie de contrôle par supervision SCT ait été largement acceptée dans le monde académique et que certaines applications de la SCT dans l'industrie aient été présentées dans la littérature (Balemi et al. 1993), (Brandin 1996), (Darabi et al. 2003), (Schafaschek et al. 2014), son application industrielle reste difficile. Cela est principalement dû aux problèmes rencontrés lors la mise en œuvre « physique » de la SCT quand le modèle à traiter est complexe.

Au début des années 70, la nécessité de décrire des contrôleurs de plus en plus complexes, dont la mise en œuvre serait facilitée par le développement des microprocesseurs, devenait une évidence. Les automates programmables industriels (API) apparaissaient alors et un outil de modélisation des contrôleurs ayant pour objectif d'être implémentés dans un API devient nécessaire. La formalisation de ces contrôleurs en adoptant le choix des automates temporisés (Bauer et al. 2004) comme outil de modélisation présente quelques inconvénients tels que : (i) la complexité de la représentation de la sélection de séquences et des séquences simultanées au niveau des transitions, (ii) la complexité de la modélisation des actions et (iii) la difficulté d'instanciation de plusieurs programmes API (esclave) dans le programme principal (maître). L'outil de modélisation doit alors répondre aux deux exigences suivantes (David 1993) :

- La première exigence est de pouvoir décrire la séquence des états d'un système qui peut contenir un très grand nombre d'états. Les Réseaux de Petri (RdP) étudiés au début des années 60 par C.A. Petri ont fourni une solution à ce problème (Petri 1996). Le RdP est un outil graphique qui permet de modéliser la concurrence. Une place dans un RdP correspond à une composante de l'état. Si un seul composant change (marquage d'état), l'état alors change. Ensuite, on peut observer quelle partie du système change, et de plus, un nombre raisonnable de places peut fournir la représentation d'un grand nombre d'états (le nombre de marquages peut augmenter de façon exponentielle avec le nombre de places).

- La deuxième exigence est d'avoir une compréhension claire du comportement des entrées/sorties d'un système. Il faut modéliser les différents types d'entrées/sorties et savoir quand un changement d'entrée particulier peut modifier l'état et/ou la sortie du système. Une solution à cette exigence a été apportée par le GRAFCET. Ce modèle a été défini par l'AFCET[5]. La norme française est devenue en 1987 une norme internationale (CEI60848 1988). Les comités nationaux des pays suivants ont déclaré leur explicite approbation de la publication : Australie, Belgique, Canada, Tchécoslovaquie, Danemark, Égypte, Finlande, France, Allemagne, Italie, Japon, Pays-

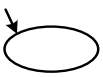
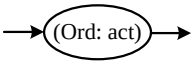
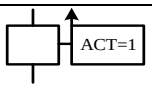
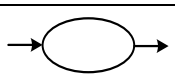
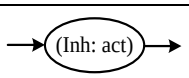
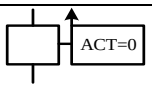
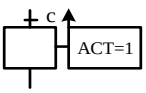
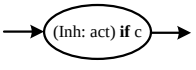
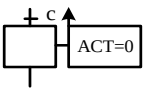
[5] En 1975, une commission de normalisation de la représentation du cahier des charges d'un automatisme logique (animée par Michel Blanchard) est créée au sein de l'AFCET (Association Française pour la Cybernétique Economique et Technique) ; après deux ans de travail, le rapport de la commission (présenté en Décembre 1977 à Montpellier) définit le Grafcet. Le groupe EPA (Equipements de Production Automatisée) de l'ADEPA (Agence pour le DÉveloppement de la Productique Appliquée à l'industrie) reprend les travaux de la commission de normalisation, son travail de mise en forme permet la parution de la norme NF C03-190 en 1982. Cette norme définit les symboles et les règles nécessaires à la représentation graphique, ainsi que l'interprétation qui en est faite. Cette norme a été établie pour les systèmes automatisés de production des applications industrielles, cependant aucun champ d'application n'est exclu.

Bas, Norvège, Portugal, Espagne, Suède, Suisse, Turquie, Royaume-Uni et Yougoslavie. Fondamentalement, un grafcet peut être comparé à une certaine classe de RdP (comme par exemple les RdP interprétés) mais le grafcet comporte des différences fondamentales puisqu'il est déterministe, il y a obligation de franchissement des transitions franchissables et le grafcet propose une interprétation unique des entrées/sorties. Il s'agit d'un modèle de spécification qui ne décrit que la fonction à exécuter, c'est-à-dire une machine séquentielle au sens mathématique, libre de toute technologie et de toute mise en œuvre.

Aujourd'hui, le grafcet est bien accepté dans l'industrie grâce à son interface graphique et surtout grâce à sa large diffusion dans le système éducatif et dans l'industrie dès la fin des années 70 plutôt que les RdP qui sont des modèles graphiques et mathématiques utilisés de préférence dans le monde académique pour par exemple des méthodes formelles d'analyse des SED. C'est pour toutes ces raisons que notre choix s'est orienté vers le grafcet comme outil d'interprétation des contrôleurs distribués de comportement normal (CD) et des contrôleurs de comportement dégradé (CD^T). C'est à l'utilisateur, ensuite, de choisir le langage de programmation d'API.

Plusieurs règles d'interprétation du grafcet sont données par les auteurs dans (Qamsane et al. 2016). Un état d'un CD/CD^T est interprété par une étape du grafcet, tandis qu'une transition d'un CD/CD^T est transformée en une réceptivité de transition du grafcet correspondant. Nous ajoutons à ces règles une interprétation des événements temporisés. Dans une traduction en grafcet, l'activation et la désactivation de l'horloge (A_{ck}, D_{ck}) ne sont pas interprétées, seule la durée est présentée. La transition marquée « *d* » sur le CD^T est interprétée par une réceptivité de transition sous la forme de (d/X_k), ce qui signifie que si la variable associée à l'étape « *k* » est activée, celle-ci doit être maintenue jusqu'à ce que la durée s'écoule ce qui provoque la transition à l'étape suivante « *k + 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'ordre, elle se traduit par une réceptivité à la transition qui précède l'étape associée à l'action (act = 1) ou (act = 0) (voir tableau 4).

Tableau 4 : Règles d'interprétation des éléments et des macro-états du CD/CD^T en grafcet

Elément du CD/CD^T	Interprétation en grafcet	Macro-état du CD/CD^T	Interprétation en grafcet
			
			
			
			

4.5 Modélisation du reconfigurateur

Après l'interprétation des différents contrôleurs distribués des deux modes de fonctionnement en grafcet et avant toute implémentation de ces derniers dans un API, il est nécessaire de définir les conditions de passage du mode normal au mode dégradé et vice versa lors de la détection d'un défaut ou lors de sa réparation.

Soit $G^{*}_{(N)}$ un extrait du grafcet de commande du fonctionnement normal d'un EPO quelconque, et $G_{(F)}$ un extrait de son grafcet de commande équivalent au fonctionnement normal (ou dégradé) présentés dans la figure 55.

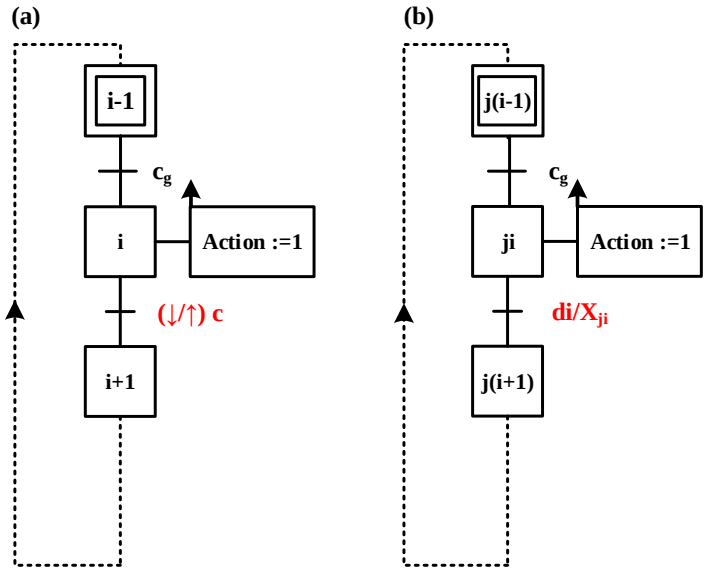


Figure 55 : Extraits des grafcets : (a) Grafcet^{N} et (b) Grafcet^{F}

La condition de reconfiguration notée (CR) est définie par l'équation logique suivante :

CR : **Si** X_i **et** f_c =1 **alors**

$$(F: G_{(F)}\{X_{ji}\} \textbf{ et } F: G_{(N)}\{\}) \qquad \textbf{(eq3)}$$

Sinon-Si X_{ji} **et** $f_c = 0$ **Alors**

$$(F: G_{(N)}\{X_i\} \textbf{ et } F: G_{(F)}\{\})$$

- 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 dégradé
- X_i est la variable booléenne associée à l'étape « i » du $G_{(N)}$
- X_{ji} est sa variable correspondante associée à l'étape « ji » du $G_{(F)}$
- f_c est le défaut détecté du capteur c

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

Cette équation est par la suite interprétée en grafcet pour gérer les deux grafcets des deux modes de fonctionnement de chaque EPO. L'interprétation est définie comme suit :

- Les deux contraintes conditionnées par *Si* sont interprétées par des réceptivités de transitions juste après l'étape initiale au niveau du grafcet de reconfiguration.
- L'expression *Sinon* correspond à un choix et interprétée alors par une divergence *OU* au niveau du grafcet.
- Les actions à effectuer après l'expression *Alors* sont traduites en actions correspondantes à des ordres de forçage des grafcets.

En suivant ces démarches, le grafcet de reconfiguration correspondant à l'équation (eq3) est donné dans la figure 56.

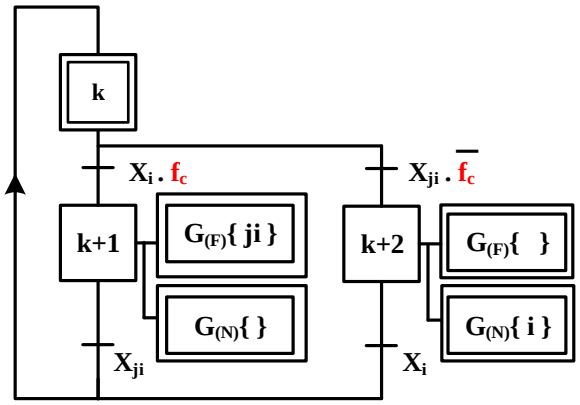


Figure 56 : Interprétation de la contrainte de reconfiguration en grafcet

Pour une raison de libération du fonctionnement des grafcets $G_{(N)}$ et $G_{(F)}$, une vérification des étapes dans lesquelles les deux grafcets sont forcés est ajoutée en réceptivité de transition juste après l'étape de forçage.

4.6 Vérification et implémentation

4.6.1 Méthodes d'implémentation de la commande

L'approche de synthèse et de reconfiguration de la commande que nous proposons repose sur différents modèles et formalismes. La figure 57 illustre une vue abstraite des différentes voies qui peuvent être suivies pour obtenir la réalisation du contrôle souhaité à partir de spécifications de contrôle informelles. Le processus global de conception implique trois types d'axes génériques : la formalisation, la synthèse et l'implémentation. Les deux premiers axes sont largement décrits dans les sous sections dessus. L'axe d'implémentation est discuté dans cette sous-section.

L'approche industrielle traditionnelle présentée par la partie de la figure 57 entourée de rouge se base sur une implémentation directe d'un programme de contrôle qui doit être raffiné pour obtenir la réalisation du contrôle cible. Ce raffinement est obtenu grâce à des procédures de tests et de validation pilotées par une description plus ou moins formalisée du système et des propriétés requises. Notre approche est basée sur une démarche alternative représentée par la partie entourée de bleu dans la figure 57. Elle consiste, à partir d'un modèle du système et d'un ensemble de spécifications locales et globales, à implémenter l'ensemble des contrôleurs

distribués avec un ensemble des contraintes de reconfiguration pour réaliser le contrôle global désiré.

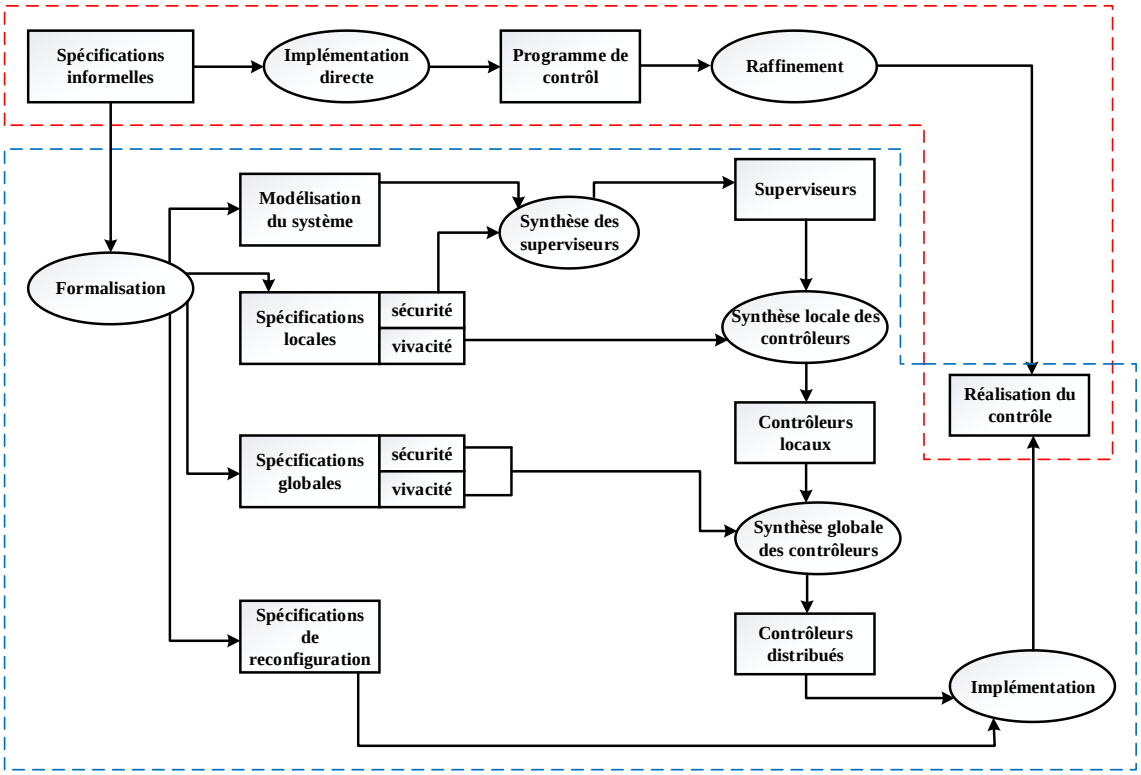


Figure 57 : Vue abstraite des activités et des modèles impliqués dans la synthèse et l'implémentation des contrôleurs (inspirée de (Zaytoon et Riera 2017))

Avant cette implémentation, des tests de vérifications sont effectués pour s'assurer que la commande est non bloquante. Pour cela, nous proposons une méthode de vérification par modèle checking en utilisant le logiciel UPPAAL[6] . C'est un environnement d'outils intégré pour la modélisation, la validation et la vérification des systèmes en temps réel modélisés par des réseaux d'automates étendus. Après une vérification satisfaisante, les contrôleurs peuvent être implémentés dans un automate programmable industriel (API)

Depuis les années 80, les API sont exploités dans de nombreux systèmes et plus spécialement dans les SAP. Les ingénieurs de contrôle/commande gèrent principalement le développement des applications d'automatisation industrielle par l'implémentation directe des contrôleurs dans des API (comme évoqué ci-dessus) tout en se basant sur l'interprétation des spécifications informelles des cahiers de charges. Cette démarche est soutenue par des outils d'ingénierie standardisés (diagramme d'activités UML (Yangui 2016) par exemple) pour la programmation des API. Cependant, les spécifications informelles du logiciel de contrôle doivent être transférées manuellement et intuitivement dans le programme de contrôle car elles ne sont pas formellement définies dans la pratique en raison d'un manque de temps et d'expertise (Zaytoon et Riera 2017) (Pichard 2018).

Cette voie d'implémentation du contrôleur conduit le plus souvent à une documentation inexacte des interdépendances séquentielles du programme de contrôle à implémenter et à des coûts supplémentaires liés à l'interprétation erronée des spécifications textuelles. Cela rend les processus de diagnostic des fautes, de reconfiguration et de maintenance du contrôleur

[6] www.uppaal.org

extrêmement difficiles. Par conséquent, de nombreuses études ont contribué à l'implémentation des contrôleurs basés sur la SCT en utilisant des programmes API. Ces derniers sont écrits dans des langages standardisés, tels que le langage Ladder (LD), le texte structuré (ST), le langage LIST (IL) ou le langage de programmation spécifique (SFC).

Un API reçoit des signaux d'entrées provenant des capteurs et envoie des signaux de sorties aux actionneurs (figure 58), conformément aux lois de commande implémentées dans le programme.

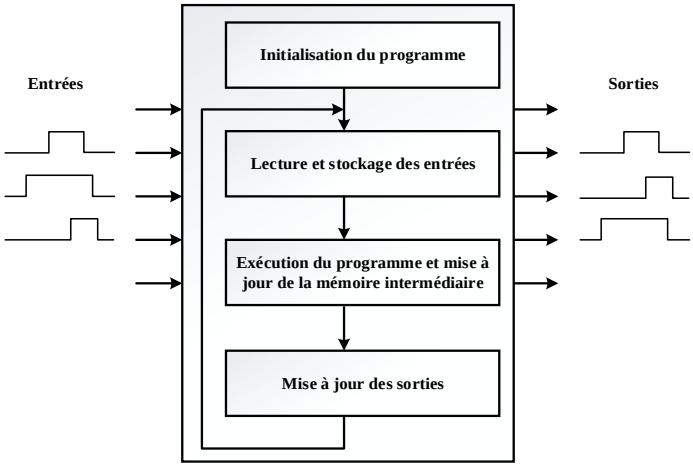


Figure 58 : Principe basic d'un API

En général, un API effectue cycliquement trois tâches : (1) la lecture et le stockage des entrées, (2) l'exécution du programme et (3) la mise à jour des sorties. La période affectée à ces tâches peut être constante (balayage périodique) ou peut varier (balayage cyclique). Pour un témoin extérieur, et en particulier pour le système, l'API peut être considéré comme un système réactif car les signaux de sortie changent simultanément d'états en réponse aux signaux d'entrées.

Le programme automate permet le calcul en temps réel des nouvelles valeurs de sorties, en fonction de l'état actuel de l'automate et l'observation des nouvelles valeurs d'entrées. Le programme se compose d'un ensemble d'expressions classiquement booléennes, qui sont évaluées pendant un cycle, avec leurs résultats stockés dans la mémoire intermédiaire. Lorsque toutes les expressions sont évaluées, généralement de manière séquentielle, la partie de la mémoire intermédiaire qui correspond aux signaux de sorties est copiée sur les sorties qui affectent le monde extérieur.

4.6.2 Méthode proposée pour la vérification formelle des grafcets de commande

L'approche de la reconfiguration distribuée proposée dans ce chapitre permet de s'affranchir d'étapes de composition entre modèles, source d'explosion de l'espace d'états. La proposition implique pour chaque élément de partie opérative EPO, un modèle de contrôleur du fonctionnement normal, un modèle de contrôleur du fonctionnement dégradé et un grafcet de reconfiguration. Il y a cependant 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, avant toute implémentation, une vérification de la synthèse des contrôleurs distribués établie est capitale. Cette phase consiste à vérifier la propriété de non-blocage de la commande conçue et la propriété d'atteignabilité des grafcets des comportements normaux et dégradés lors d'une reconfiguration.

Le codage des grafcets est supposé toujours juste si le concepteur suit correctement les règles de traduction, mais il peut être aussi vérifié de manière implicite. En effet, si la vérification d'une propriété n'est pas satisfaisante, cela implique qu'une erreur est commise lors de la synthèse des contrôleurs ou au niveau de la traduction des contrôleurs en grafcets si aucune erreur n'est prélevée au niveau de la synthèse.

Dans cet objectif, les contrôleurs distribués sont vérifiés grâce à un modèle de vérification avant l'implémentation dans un API. Les différents grafcets distribués des comportements normaux et dégradés, ainsi que les grafcets assurant la reconfiguration sont traduits en langage structuré (ST) et vérifiés à l'aide du logiciel UPPAAL.

Les programmes implantés dans les automates programmables industriels sont exécutés selon un cycle composé de quatre phases, comme le montre la figure 59 à l'exception de la phase d'initialisation lors du premier cycle automate.

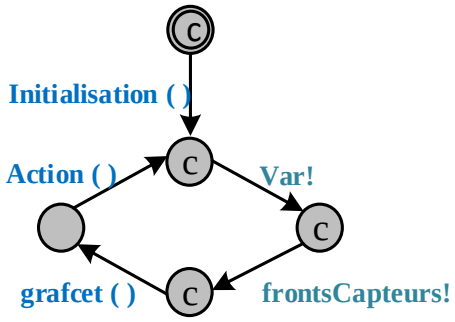


Figure 59 : Modélisation d'un cycle automate

Les quatre phases sont :

- Lecture des entrées (**var!**) : synchronisation avec les modèles d'entrées
- Lecture des fronts montants et descendants des capteurs (**frontsCapeurs!**) : synchronisation avec les modèles de gestions des fronts
- Exécution du programme principal (appel de la fonction **grafcet ( )**)
- Mise à jour des sorties (appel de la fonction **action ( )**)

Le modèle du cycle API permet une synchronisation de tous les modèles du système tout en respectant l'ordre d'exécution séquentielle des tâches cycliques du programme API. Lors du tout premier cycle automate, une phase d'initialisation est nécessaire pour effectuer certaines opérations telles que l'initialisation des programmes, les variables internes, les états des équipements du système. Les synchronisations sont de type broadcast (diffusion).

Le modèle de la figure 60, synchronisé avec le modèle de cycle automate de la figure 59 à travers le message "**var?**", permet d'assigner aléatoirement des valeurs vraies ou fausses à un capteur (représenté par l'argument "x" du modèle) au cours de chaque cycle automate. Ce modèle est instancié pour chaque capteur observé par le système.

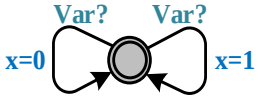


Figure 60 : Modèle générateur des capteurs

La modélisation des éléments du système est basée sur des fronts montants et descendants, pour cela un modèle générateur de ces fronts est modélisé (figure 61). Le passage de x à 1 (x==1) permet la lecture à 1 du front montant de x présenté par (RE_x=1), ensuite la lecture de la valeur du front montant passe à 0 (x est toujours à 1). Par la suite, x est désactivé ce qui est détecté par la lecture à 1 du front descendant de x (FE_x=1).

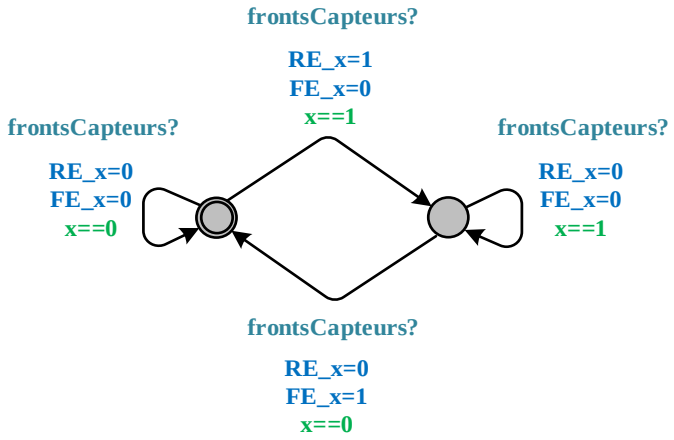


Figure 61 : Modèle générateur des fronts des capteurs

La modélisation en ST appliquée aux grafcets du système est donnée par la méthode suivante :

- Déclaration de toutes les variables constituant les grafcets (étapes, capteurs, action, durées, défauts...).
- Traduction des grafcets en équations de type auto-maintien (Machado et al. 2006). Pour les déterminer, il faut d'abord définir les fonctions de transition de tous les grafcets. Ensuite, les équations d'activation des étapes comme tout en déclaration les différentes fonctions de transition. La fonction d'une transition « ft » est donnée par une conjonction de l'état des étapes précédentes et de la condition logique associée à la transition. Une étape est activée si la fonction de l'une des transitions d'entrée est vraie. Elle reste active tant que la fonction de sortie est fausse. Les actions associées à une étape sont actives lorsque l'étape est active (Machado et al., 2006).
- Affectation des actions aux étapes correspondantes.

Pour vérifier d'éventuelles erreurs de transcription des grafcets en ST, une vérification formelle peut être appliquée dans UPPAAL. En effet, il faut vérifier que toutes les étapes X_i des grafcets conçus sont atteignables. Cependant, cette vérification reste non-suffisante, seule la relecture du programme peut assurer que cette transcription ne soit pas entachée d'erreurs.

Les modèles obtenus des contrôleurs ainsi que les reconfigurateurs couvrent l'espace d'états du système, c'est-à-dire que toutes les évolutions souhaitées du système sont déjà présentées dans les modèles traduits en ST. Par conséquent, une vérification exhaustive de la propriété de blocage (existe-il une évolution bloquante) suffit à démontrer que les contrôleurs distribués ne sont pas bloquants.

Suite à une vérification formelle satisfaisante des grafcets de commande, la commande est testée, simulée sur un jumeau numérique et ensuite implémentée dans un système réel. La figure 62 décrit l'architecture de vérification formelle et de simulation adoptée dans ce mémoire pour appliquer la commande conçue sur le système réel.

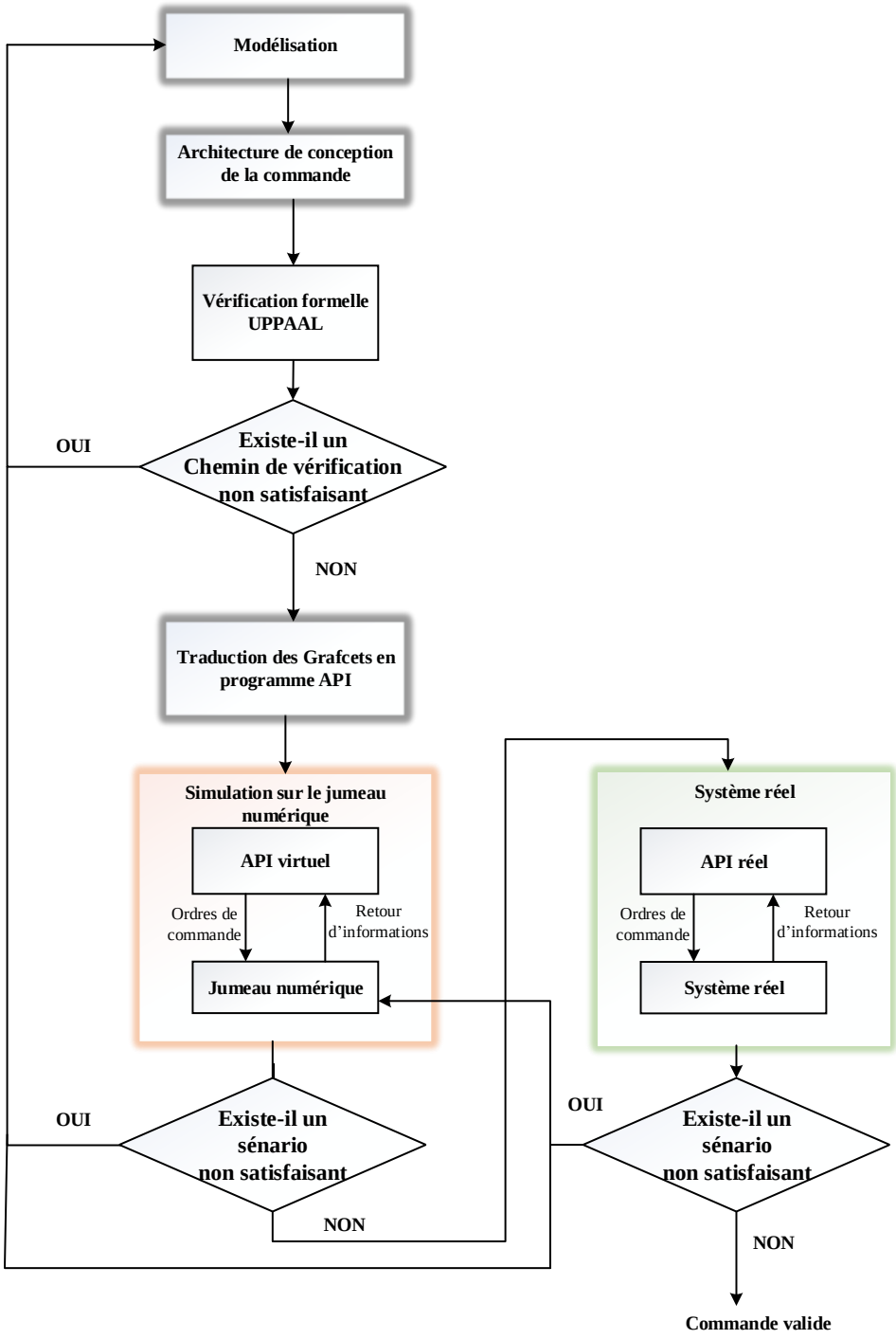


Figure 62 : Architecture de vérification formelle, simulation et application réelle

Après la phase de modélisation et conception de la commande, certaines exigences et propriétés doivent être testées (blocage de la commande, atteignabilité, …). Cette phase est assurée par la vérification formelle assurée par le model-cheker d'UPPAL. La commande ne peut être traduite en langage de programmation d'API que si toutes les exigences et propriétés sont exécutées avec succès. Dans le cas contraire, la commande présente des erreurs qui doivent être diagnostiquées et relevées pour les corriger. En supposant que la conception de la commande est correcte, l'erreur se manifeste généralement au niveau de la modélisation de la partie opérative et des spécifications. La deuxième partie de cette architecture consiste à implémenter la commande dans un API virtuel et la simuler sur le jumeau numérique du

système réel. Un ensemble de procédure d'essais (scénarios) est établi pour vérifier le bon fonctionnement du jumeau numérique.

Le jumeau numérique est devenu un outil important pour la réalisation d'une production manufacturière intelligente grâce à la numérisation des systèmes automatisés de production. Les technologies récentes, telles que les capteurs intelligents, l'Internet des objets (IoT), le cloud, l'apprentissage automatique et l'intelligence artificielle (IA), ont permis la numérisation et facilité le développement des jumeaux numériques dans le domaine de la production manufacturière. Un récent sondage effectué par Gartner révèle que 75% des organisations mettant en œuvre l'IoT, utilisent déjà des jumeaux numériques ou prévoient leur exploitation dans un délai d'un an. D'ici 2022, plus des deux tiers de ces sociétés devraient avoir déployé au moins un jumeau numérique en production (Gartner 2019).

(Grieves et Vickers 2017) proposent une des premières définitions du jumeau numérique : « Un jumeau numérique est une construction informationnelle numérique d'un système physique vu comme étant une seule entité ». Dans ce contexte, le mot « jumeau » implique que cette information numérique serait liée au système physique tout au long de son cycle de fonctionnement.

Le concept de jumeau numérique (Digital Twin) d'une usine se réfère à un modèle 3D d'une usine complète avec ses ressources, qu'elles soient humaines ou industrielles. Cette représentation numérique peut concerner un ensemble plus ou moins conséquent tel qu'une chaîne de production ou une usine entière. L'application de ce concept à la production manufacturière permet aux fabricants de créer des représentations numériques adaptées à leurs systèmes et aux processus de production en utilisant des données et des informations collectées permettant l'analyse, la prise de décision et le contrôle pour répondre à un objectif défini (figure 63).

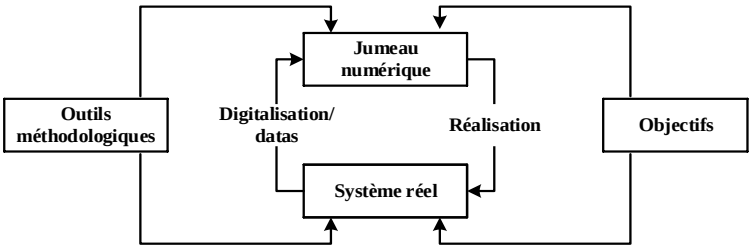


Figure 63 : Concept du jumeau numérique (inspirée de (Shao et Helu 2020))

La commande simulée sur le jumeau numérique (figure 62) n'est considérée valide pour implémentation dans le système réel que si tous les scénarios établis sont exécutés correctement. L'apparition d'un scénario non-satisfaisant implique qu'une erreur est commise lors de la phase de la modélisation. Après l'identification et la correction des erreurs, toutes les étapes précédentes sont rétablies jusqu'à que le jeu de tous les scénarios du jumeau numérique soit exécuté correctement. La commande est alors jugée valide, et implémentée dans un API réel. L'ensemble des scénarios est ensuite testé de nouveau mais cette fois sur le système réel, l'exécution de cet ensemble comme souhaité signifie que la commande est valide et aucune modification n'est nécessaire. Dans le cas contraire, deux cas sont à prévoir : soit la commande non-correcte est due à une erreur au niveau de la phase de la modélisation et doit être corrigée et revérifiée soit elle est liée à une erreur/écart lors de la réalisation du jumeau numérique, dans ce cas une correction du jumeau est nécessaire.

Après avoir explicité notre méthodologie de conception de la commande, nous l'appliquons dans la suite de ce chapitre sur un exemple « didactique simple ».

5 Application de l'approche de reconfiguration sur un exemple didactique simple

5.1 Présentation de l'exemple d'illustration

Le laboratoire CReSTIC collabore avec la société Real Games[7] depuis 2008, dans le cadre d'un partenariat de recherche et développement. Real Games propose des logiciels de simulations 3D de parties opératives (FACTORY I/O et HOME I/O).

L'illustration des propos de ce chapitre utilise le logiciel FACTORY I/O[8]. C'est un logiciel académique et de recherche qui présente la particularité de pouvoir être connecté à un environnement extérieur (automate programmable industriel, Python …). Il propose plus de 20 scènes inspirées des applications industrielles typiques pour pratiquer les démarches de contrôle du monde réel.

FACTOTY I/O offre aussi à l'utilisateur, la possibilité de la création d'une usine virtuelle à l'aide d'une bibliothèque d'éléments de PO industrielle, y compris des capteurs, des convoyeurs, des ascenseurs, des stations et bien d'autre. En effet, les outils d'édition proposés par ce logiciel font de la construction d'une scène 3D une expérience confortable et naturelle.

Pour communiquer avec l'automate, le simulateur d'automate ou le modbus Factory I/O utilise des pilotes pour chaque technologie spécifique (Allen-Bradley, Siemens, …).

Parmi les avantages de ce logiciel, il faut noter son aptitude à générer des défauts au niveau des capteurs et/ou actionneurs offrant une possibilité pour proposer des solutions de dépannage et de tolérance aux fautes.

Le système utilisé est un transfert de caisses (figure 64-a), il est de nature distribuée et instrumenté de 6 capteurs, un bouton poussoir, et 4 actionneurs. Il consiste à transférer les caisses générées par le poste de chargement vers un poste de déchargement en passant par deux convoyeurs. Le passage entre ces deux derniers est assuré par deux poussoirs. A noter ici que même si ce système choisi est de petite taille et apparait comme un exemple trivial, il permet d'aborder notre démarche rapidement.

Le système à commander est constitué de :

- Deux poussoirs P_1 et P_2 représentés par deux vérins monostables simple effet avec leurs capteurs de fin de course associés ($\{s_1, s_2\}$ pour P_1 et $\{s_3, s_4\}$ pour P_2) comme le montre la figure (634b).
- Deux convoyeurs transportant les caisses et les évacuant vers un poste de déchargement.
- Deux capteurs de position s_5 (resp. s_6) permettent de détecter les caisses présentes devant le poussoir P_1 et à la sortie du convoyeur 2.
- D'un bouton poussoir *dcy* pour le démarrage du cycle.

[7] https://realgames.co/
[8] https://factoryio.com/

Le cahier des charges à respecter est le suivant : le système ne doit pas envoyer les ordres de sortie des vérins P_1 et P_2 en même temps, l'ordre de sortie d'un poussoir ne peut être réalisé que si le vérin est en position rentrée (s_1/s_3) et l'ordre de sortie du vérin P_2 ne peut être exécuté qu'après la sortie du vérin P_1.

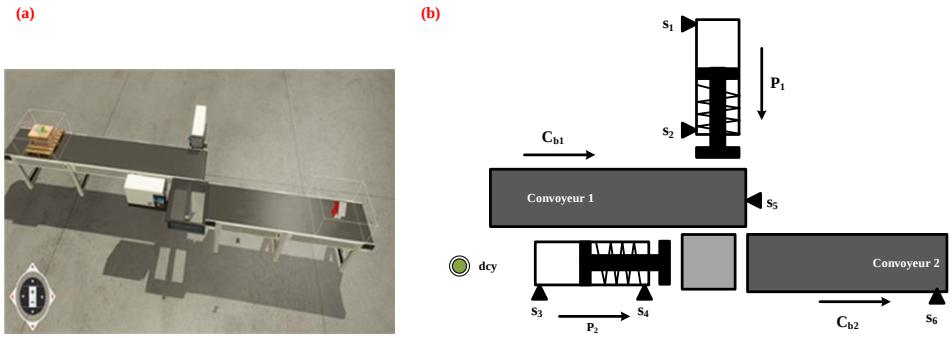


Figure 64 : (a) Système de transfert des caisses, (b) Actionneurs et capteurs constituant le système

5.2 Modélisation du comportement de la PO

La partie opérative est constituée de quatre éléments de partie opérative : le poussoir P_1, le poussoir P_2, le convoyeur C_{b1} et le convoyeur C_{b2}. La première étape consiste à modéliser ces EPO pour le comportement normal. Cette modélisation est assurée par le modèle pratique (Philippot 2006) en utilisant les événements des EPO associés à chaque actionneur et ses capteurs appropriés.

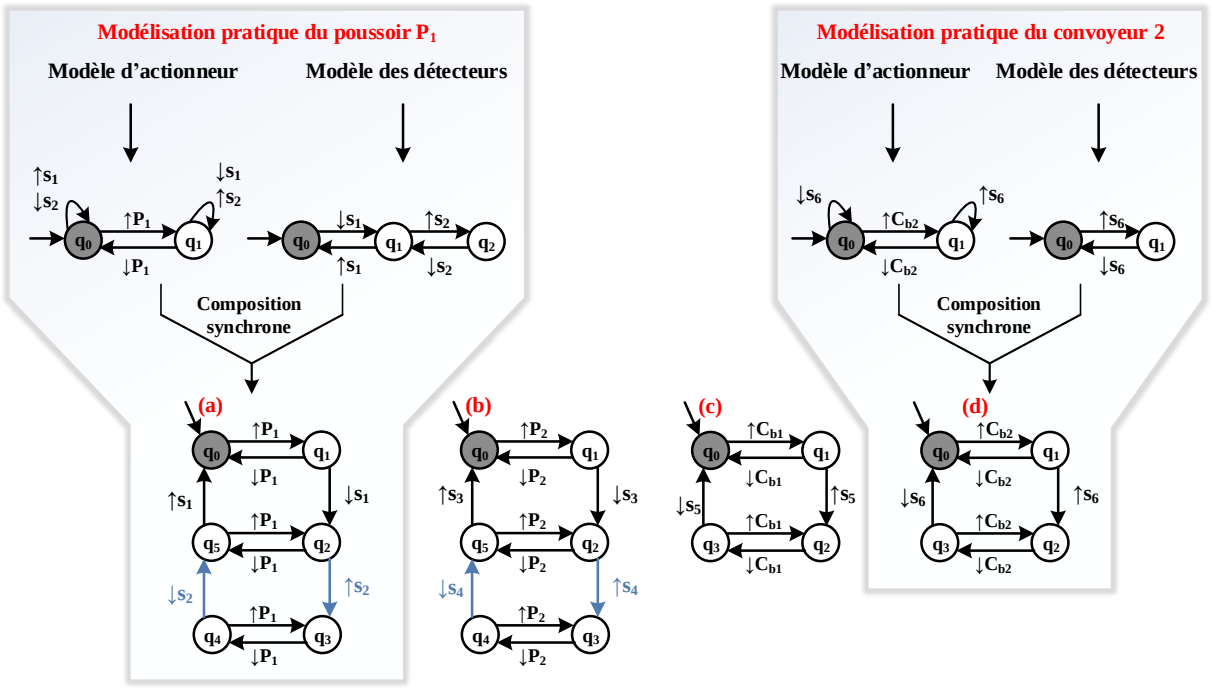


Figure 65 : Les modèles des EPO du mode normal. (a) P_1, (b) P_2, (c) Cb_1, (d) Cb_2

La figure 65 détaille la modélisation pratique du poussoir P_1 et du convoyeur C_{b2}. Le modèle pratique de l'EPO correspondant au poussoir P_1 est un automate obtenu par composition synchrone du modèle des détecteurs avec le modèle actionneur de cet élément. Le détecteur est modélisé par un automate dépendant du nombre de détecteurs associés au poussoir 1 (s_1 et s_2). L'actionneur est modélisé par un automate qui dépend de la technologie électromécanique

(monostable) du poussoir combiné avec les événements des détecteurs associés à cet actionneur. Cette démarche est appliquée aux quatre EPO constituant le système étudié. Les automates décrivant l'ensemble des comportements normaux des quatre EPO sont présentés dans la figure 65.

Les deux automates du poussoir (P_1) et du convoyeur 1 (Cb_1) sont définis comme suit :

(a): $A^{N(P1)} = \{Q^{N(P1)}, \Sigma^{N(P1)}, \delta^{N(P1)}, q_0^{N(P1)}, Q_m^{N(P1)}\}$ tel que :

- $Q^{N(P1)} = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
- $\Sigma^{N(P1)} = \Sigma_{nT}^{N(P1)}$, avec : $\Sigma_{nT}^{N(P1)} = \uparrow\downarrow Z \cup \uparrow\downarrow E$ avec : $\uparrow\downarrow Z = \{\uparrow P_1, \downarrow P_1\}$ et $\uparrow\downarrow E = \{\uparrow s_1, \downarrow s_1, \uparrow s_2, \downarrow s_2\}$
- $q_0^{N(P1)} = q_0$
- $Q_m^{N(P1)} = q_0$

(b): $A^{N(cb1)} = \{Q^{N(cb1)}, \Sigma^{N(cb1)}, \delta^{N(cb1)}, q_0^{N(cb1)}, Q_m^{N(cb1)}\}$ tel que :

- $Q^{N(cb1)} = \{q_0, q_1, q_2, q_3\}$
- $\Sigma^{N(cb1)} = \Sigma_{nT}^{N(cb1)}$, avec: $\Sigma_{nT}^{N(cb1)} = \uparrow\downarrow Z \cup \uparrow\downarrow E$ avec : $\uparrow\downarrow Z = \{\uparrow C_{b1}, \downarrow C_{b1}\}$ et $\uparrow\downarrow E = \{\uparrow s_5, \downarrow s_5\}$
- $q_0^{N(cb1)} = q_0$
- $Q_m^{N(cb1)} = q_0$

De la même manière, on obtient les deux automates décrivant P_2 et Cb_2. Les différents modèles obtenus décrivent toutes les évolutions possibles de l'actionneur vis-à-vis des capteurs.

La deuxième étape consiste à modéliser les EPO du système pour un mode dégradé où les capteurs s_2 et s_4 sont considérés comme potentiellement *défectueux*. Soit f_{s2} (resp. f_{s4}) le défaut du capteur s_2 (resp. s_4) associé à la sortie du poussoir P_1 (resp. P_2). La modélisation des EPO, dans ce cas, est une extension des modèles donnés figure 65 en ajoutant des événements temporisés qui compensent les informations sur s_2 et s_4 comme indiqué sur la figure 66.

L'activation du capteur s_2 (resp. s_4) est compensée dans le mode dégradé par l'horloge ck_1 (resp. ck_3). Tandis que la désactivation du capteur s_2 (resp. s_4) est compensée par l'horloge ck_2 (resp. ck_4).

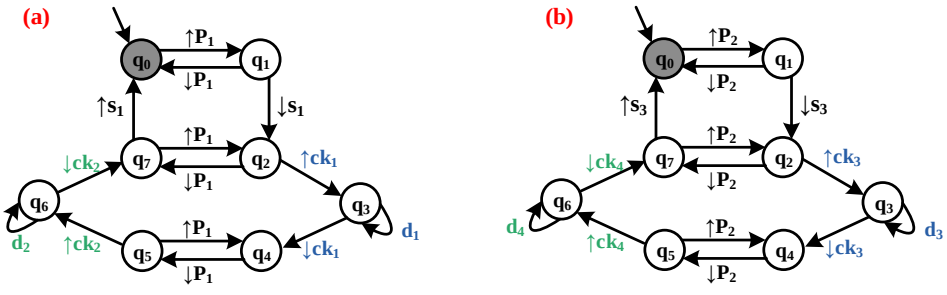


Figure 66 : Les modèles des EPO du mode dégradé. (a) P_1, (b) P_2

Le modèle du comportement équivalent au comportement normal du poussoir P_1 est présenté par l'automate suivant :

(a): $A^{F(P1)} = \{Q^{F(P1)}, \Sigma^{F(P1)}, \delta^{N(P1)}, q_0^{F(P1)}, Q_m^{F(P1)}\}$ tel que:

- $Q^{F(P1)} = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
- $\Sigma^{F(P1)} = \Sigma_{nT}^{F(P1)} \cup \Sigma_{T}^{F(P1)}$, avec : $\Sigma_{nT}^{F(P1)} = \uparrow\downarrow Z \cup \uparrow\downarrow E$ avec : $\uparrow\downarrow Z = \{\uparrow P_1, \downarrow P_1\}$ et $\uparrow\downarrow E = \{\uparrow s_1, \downarrow s_1\}$ et $\Sigma_{T}^{F(P1)} = \{\uparrow ck_1, \uparrow ck_2, \downarrow ck_1, \downarrow ck_2, d_1, d_2\}$
- $q_0^{F(P1)} = q_0$
- $Q_m^{F(P1)} = q_0$

Les deux modèles obtenus décrivent toutes les évolutions possibles des poussoirs P_1 et P_2 en fonction de leurs capteurs et de leurs horloges appropriées.

Parmi ces évolutions, après la désactivation du capteur s_1 du modèle (a) de la figure 66, un relâchement de l'ordre P_1 à partir de l'état q_2 est possible avant l'activation de l'horloge ck_1. Cette évolution n'est pas souhaitée dans le fonctionnement de cet EPO et sera supprimée par la suite en intégrant les spécifications locales.

5.3 Synthèse des contrôleurs locaux

Pour synthétiser les contrôleurs locaux correspondant à chaque EPO, une détermination de l'ensemble des spécifications locales est nécessaire. Cet ensemble est défini pour obtenir les comportements des quatre EPO dans le cas du fonctionnement normal et les deux comportements dégradés des poussoirs P_1 et P_2. Le tableau 5 présente les spécifications locales à appliquer aux différents EPO pour chaque mode de fonctionnement.

Tableau 5 : Ensemble des Spécifications locales

EPO	Spécifications locales pour le mode normal	Spécifications locales pour le mode dégradé
P_1	(1) : $(q_1 + q_2) \cdot \downarrow P_1 = 0$ (2) : $(\mathbf{q_4} + \mathbf{q_5}) \cdot \uparrow P_1 = 0$	(9) : $(q_1 + q_2) \cdot \downarrow P_1 = 0$ (10) : $(\mathbf{q_5} + \mathbf{q_7}) \cdot \uparrow P_1 = 0$
P_2	(3) : $(q_1 + q_2) \cdot \downarrow P_2 = 0$ (4) : $(\mathbf{q_4} + \mathbf{q_5}) \cdot \uparrow P_2 = 0$	(11) : $(q_1 + q_2) \cdot \downarrow P_2 = 0$ (12) : $(\mathbf{q_5} + \mathbf{q_7}) \cdot \uparrow P_2 = 0$
C_{b1}	(5) : $q_1 \cdot \downarrow C_{b1} = 0$ (6) : $q_3 \cdot \uparrow C_{b1} = 0$	(5) : $q_1 \cdot \downarrow C_{b1} = 0$ (6) : $q_3 \cdot \uparrow C_{b1} = 0$
C_{b2}	(7) : $q_1 \cdot \downarrow C_{b2} = 0$ (8) : $q_3 \cdot \uparrow C_{b2} = 0$	(7) : $q_1 \cdot \downarrow C_{b2} = 0$ (8) : $q_3 \cdot \uparrow C_{b2} = 0$

Par exemple, la spécification (10) exprimée par $(q_5 + q_7) . \uparrow P_1 = 0$ reflète le fait que l'événement $\uparrow P_1$ correspondant à l'autorisation de l'ordre de sortie du poussoir P_1 ne peut pas apparaître à l'état q_5 et à l'état q_7. En effet, $\uparrow P_1$ est autorisé uniquement à l'état initial, toute autre transition associée à cet événement est supprimée. Le modèle de l'élément de partie opérative du poussoir P_1 est parcouru état par état en suivant l'évolution du modèle, et à chaque état les spécifications locales sont vérifiées et la transition associée à l'événement non-autorisé est supprimée.

Même si les deux modes de fonctionnement sont équivalents, un décalage d'états est observé entre la spécification 2 (resp. 4) et celle exprimée par 10 (resp. 12). Cela est dû à l'intégration de l'information temporisée qui conduit à l'ajout d'un état pour chaque horloge introduite dans le modèle temporisé.

Pour appliquer la synthèse de contrôle, les spécifications définies dans le tableau 5 sont interprétées sous forme de machine à états finis étendus comme expliqué précédemment. En utilisant la synthèse de contrôle basée sur SCT, nous obtenons les contrôleurs locaux $CL^{N/F}$. Chacun d'eux décrit le comportement local souhaité de chaque EPO selon des comportements normaux (figure 67) et dégradés (figure 68).

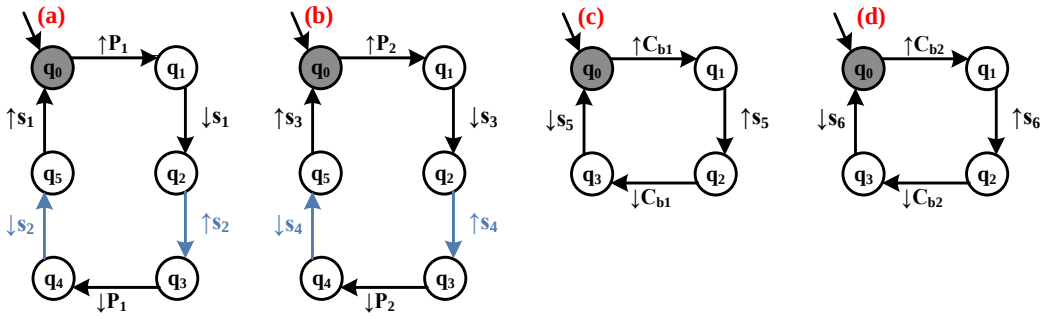


Figure 67 : Contrôleurs locaux CL^N. (a) P_1, (b) P_2, (c) Cb_1, (d) Cb_2

En fonctionnement normal, la désactivation du poussoir «↓ P_1» (resp. «↓ P_2») est assurée lorsque le capteur s_2 (resp. s_4) est activé (poussoir en position sortie).

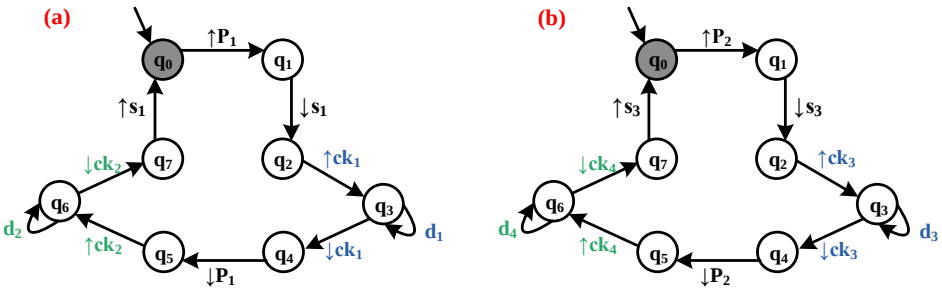


Figure 68 : Contrôleurs locaux CL^F. (a) P_1, (b) P_2

Cependant, lorsque les défauts des capteurs {fs_2, fs_4} apparaissent, les informations perdues sur ces deux capteurs sont compensées par des informations temporisées. En effet, lorsque P_1 (resp. P_2) commence à sortir, son capteur d'entrée associé s_1 (resp. s_3) est désactivé et l'horloge ck_1 (resp ck_3) est déclenchée pour une durée d_1 (resp. d_3) qui est le temps nécessaire pour que le poussoir atteigne le capteur s_2 (resp. s_4). Une fois la durée écoulée, l'horloge ck_1 (resp. ck_3) est désactivée ce qui provoque la désactivation de l'ordre P_1 (resp. P_2), puis l'activation de l'horloge ck_2 (resp. ck_4) pour une durée d_2 (resp. d_4) qui est le temps estimé pour désactiver s_2 (resp. s_4).

5.4 Synthèse globale

Les contrôleurs présentés ci-dessus n'agissent que localement. Pour assurer un fonctionnement et un contrôle global du système où ces contrôleurs peuvent communiquer entre eux, des spécifications globales sont intégrées.

Les spécifications globales du système de transfert sont présentées par le tableau 6 :

Tableau 6 : Ensemble des contraintes globales

EPO	Condition **Si**	**Alors**
C_{b1}	$(dcy + s_6) \cdot s_1=1$	Ord C_{b1}
C_{b2}	$s_4 + d_3=1$	Ord C_{b2}
P_1	$s_5 \cdot s_3 =1$	Ord P_1
P_2	$s_2 + d_1=1$	Ord P_2

La troisième contrainte par exemple, fait référence au fait que l'ordre P_1 est autorisé si le poussoir 2 est dans sa position rentrée ce qui s'exprime par $s_3 = 1$ et si une caisse est détectée devant le poussoir 1 exprimée par $s_5 = 1$. La quatrième contrainte fait référence au fait que l'ordre P_2 est autorisé si le poussoir 1 est sortie complètement soit par $s_2 = 1$. Si ce capteur est défectueux, la durée associée à son activation remplace l'information $s_2 = 1$.

Deux types de contrôleurs distribués sont distingués : CD^N pour le comportement normal et CD^F pour le comportement dégradé. Pour synthétiser les contrôleurs $\{CD^N, CD^F\}$ correspondant au système étudié, deux étapes sont à effectuer : (i) l'agrégation des contrôleurs locaux LC^N et LC^F résultant dans la sous-section 5.3 et (ii) la prise en compte des spécifications globales exprimées dans le tableau 6.

La première agrégation consiste à fusionner les événements contrôlables non temporisés des CL^N et CL^F soit $\{\uparrow P_1, \downarrow P_1, \uparrow P_2, \downarrow P_2, \uparrow Cb_1, \downarrow Cb_1, \uparrow Cb_2, \downarrow Cb_2\}$ en macro-états.

La deuxième agrégation n'est appliquée qu'aux événements temporisés des CL^F $\{\uparrow ck_1, \downarrow ck_1, \uparrow ck_2, \downarrow ck_2, \uparrow ck_3, \downarrow ck_3, \uparrow ck_4, \downarrow ck_4\}$, qui sont fusionnés en macro-états liés par les événements incontrôlables $\{\uparrow s_1, \downarrow s_1, \uparrow s_3, \downarrow s_3\}$ et les événements temporisés $\{d_1, d_2, d_3, d_4\}$.

Une fois les modèles des contrôleurs locaux agrégés et donc CLA^N et CLA^F obtenus, les événements contrôlables non temporisés $\{\uparrow P_1, \uparrow P_2, \uparrow Cb_1, \uparrow Cb_2\}$ appartiennent à l'ensemble *Ord* (ensemble d'autorisation des ordres), tandis que $\{\downarrow P_1, \downarrow P_2, \downarrow Cb_1, \downarrow Cb_2\}$ appartiennent à l'ensemble *Inh* (ensemble d'inhibition des ordres).

Les événements temporisés $\{\uparrow ck_1, \uparrow ck_2, \uparrow ck_3, \uparrow ck_4\}$ appartiennent à l'ensemble A_{ck}, tandis que les événements $\{\downarrow ck_1, \downarrow ck_2, \downarrow ck_3, \downarrow ck_4\}$ appartiennent à l'ensemble D_{ck}. Pour obtenir les contrôleurs distribués DC^N et DC^F, les contraintes globales doivent être prises en compte et ajoutées aux CLA^N et CLA^F.

Les contrôleurs distribués CD^N du système de transfert sont présentés dans la figure 69. L'événement contrôlable $\uparrow P_2$ est fusionné dans l'état initial du modèle $CD^{N/F(P2)}$ et appartient à l'ensemble Ord. Selon la vérification de la contrainte globale $\{s_2+d_1=1\}$, l'ordre P_2 est autorisé (Ord : P_2). En effet, si $s_2 = 1$, alors P_2 est autorisé, sinon (s_2 est défectueux), alors la durée d_1 nécessaire à l'activation de s_2 compense ce capteur défectueux.

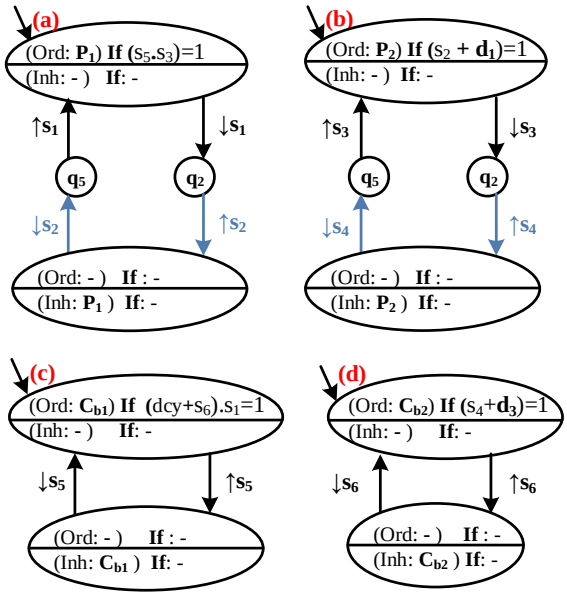


Figure 69 : Contrôleurs distribués CD du fonctionnement normal (a) P_1, (b) P_2, (c) C_{b1}, (d) C_{b2}

Dans la figure 70, une fois le capteur s_3 désactivé, l'horloge ck_3 est déclenchée, l'événement présentant son activation est fusionné dans un macro-état juste après la transition étiquetée $\downarrow s_3$ et appartient à l'ensemble A_{ck}. L'événement de désactivation de cette horloge est fusionné dans le même macro-état contenant l'inhibition de l'ordre P_2 (inh: P_2) et appartient à l'ensemble D_{ck}. ck_3 est activée suite à la désactivation de s_3 et désactivée après l'écoulement de la durée d_3. Par conséquent, aucune contrainte globale n'est nécessaire pour (A_{ck}: ck_3) et (D_{ck}:ck_3). Nous faisons de même pour $\uparrow ck_4$, $\downarrow ck_4$ et d_4.

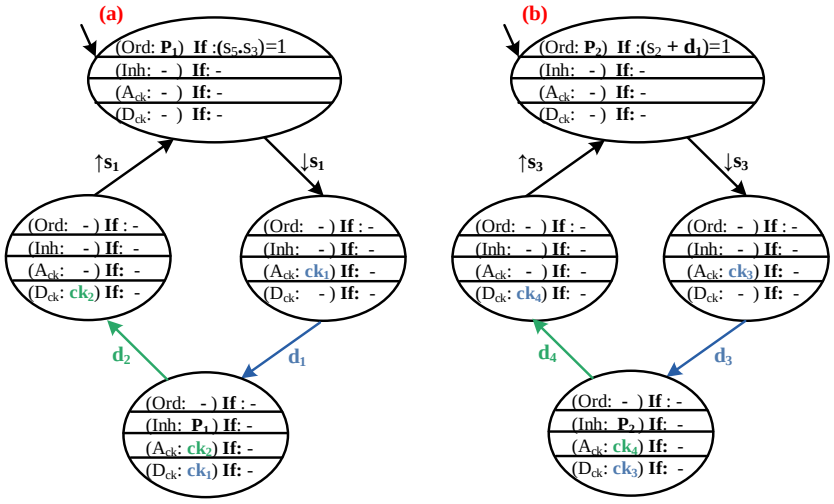


Figure 70 : Contrôleurs distribués CD^T du comportement dégradé (a) P_1, (b) P_2

5.5 Interprétation en grafcet

En appliquant l'interprétation du tableau 4 présenté précédemment aux contrôleurs distribués des comportements normaux et dégradés de l'exemple de transfert étudié précédemment, on obtient les grafcets de commande suivants :

- Quatre grafcets de commande (figure 71) pour le fonctionnement normal des vérins P_1 et P_2 et les 2 convoyeurs

- Deux grafcets de commande (figure 72) pour le fonctionnement dégradé des deux vérins dont nous avons supposé qu'un capteur est défectueux.

Les grafcets de commande résultants peuvent être considérés comme des modèles génériques pour ce type d'actionneur quel que soit le système auquel ils appartiennent. Par contre, la contrainte globale doit être modifiée selon l'environnement de travail de ces actionneurs. Par conséquent, ces modèles de grafcet peuvent être des éléments d'une bibliothèque d'éléments de commande.

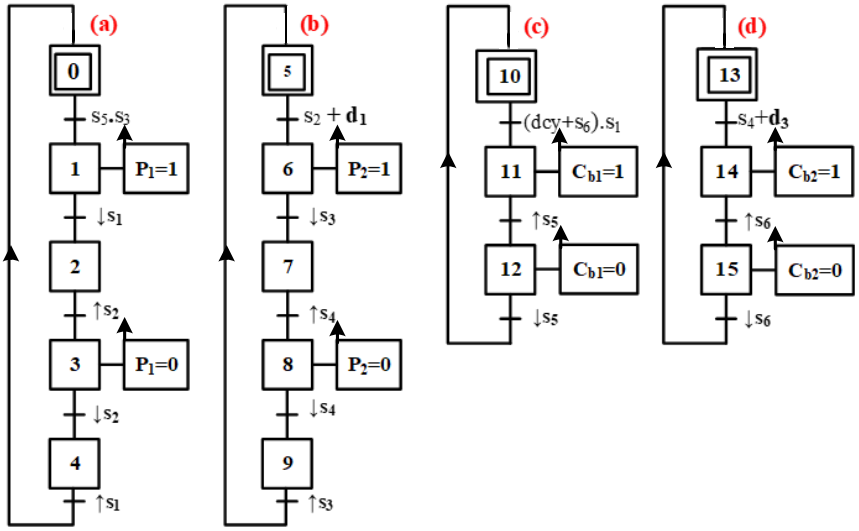


Figure 71 : Grafcet de commande de comportement normal (a) P_1, (b) P_2, (c) C_{b1}, (d) C_{b2}

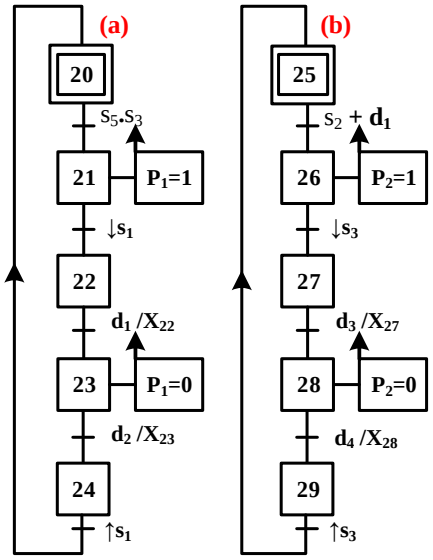


Figure 72 : Grafcets de commande de comportement dégradé (a) P_1, (b) P_2

5.6 Modélisation du reconfigurateur

Pour passer du comportement normal des deux poussoirs P_1 et P_2 à leurs comportements dégradés où fs_2 et fs_4 sont détectés, certaines règles de reconfiguration doivent être déterminées.

Dans le cas de la détection fs_2, un passage de $G^{N\,(P1)}$ à $G^{F\,(P1)}$ est nécessaire. Basé sur (eq (3)), nous obtenons les contraintes de reconfiguration ci-dessous pour le poussoir P_1 :

$CR_{1(P1)}$: **Si** X_2 **Et** $fs_2 = 1$ **Alors**
(F: $G^{F(P1)}$ {X_{22}}) Et (F: $G^{N(P1)}$ { })
Sinon X_{22} **ET** $fs_2 = 0$ **Alors**
(F: $G^{N(P1)}$ {X_2}) and (F: $G^{F(P1)}$ { })

$CR_{2(P1)}$: **Si** X_3 **Et** $fs_2 = 1$ **Alors**
(F: $G^{F(P1)}$ {X_{23}}) Et (F: $G^{N(P1)}$ { })
Sinon X_{23} **ET** $fs_2 = 0$ **Alors**
(F: $G^{N(P1)}$ {X_3}) and (F: $G^{F(P1)}$ { })

Le même principe est adopté en cas de détection fs_4. Pour gérer le passage du modèle grafcet à comportement normal au modèle grafcet à comportement dégradé, les contraintes de reconfiguration doivent être interprétées dans un formalisme grafcet pour une implémentation dans un API. Les règles de reconfiguration déterminées par les équations $CR_{1(P1)}$ et $CR_{2(P1)}$ sont données dans la figure 73.

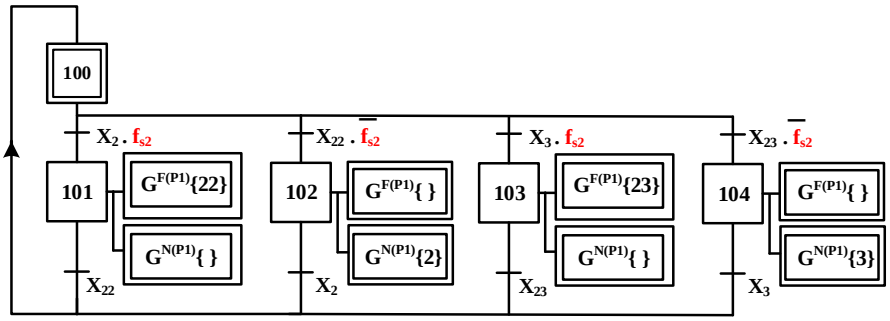


Figure 73 : Interprétation en grafcet $G^{R(P1)}$ des contraintes de reconfiguration $CR_{1(P1)}$ et $CR_{2(P1)}$

La mise en œuvre des modèles grafcet du comportement normal, dégradé et des contraintes de reconfiguration assurent une tolérance aux fautes lorsque les défauts capteurs fs2 et fs4 sont détectés. Par conséquent, une continuité du comportement souhaité du système est maintenue malgré les défauts qui entravent son comportement normal.

5.7 Modèles et programmes de vérification

Dans cette section, nous appliquons la méthode de vérification adoptée (4.6) uniquement sur le poussoir P_1 du système didactique présenté précédemment pour illustrer notre démarche.

Les grafcets à programmer sont $G^{N(P1)}$ et $G^{F(P1)}$: grafcets de commande du comportement normal et dégradé du poussoir P_1 et $G^{R(P1)}$: grafcet de reconfiguration assurant la commutation entre ces deux grafcets de commande.

La programmation en ST appliquée à ces trois grafcets est donnée par la méthode suivante :

- Premièrement, toutes les variables sont déclarées sur UPPAAL comme le montre la figure 74.

```
// Déclarations //

broadcast chan frontsCapteurs, var;

bool X0,X1,X2,X3,X4,            // Etapes du Grafcet normal //
     X20,X21,X22,X23,X24,       // Etapes du Grafcet temporisé //
     X100,X101,X102,X103,X104;  // Etapes du Grafcet de reonfiguration //
bool s1,s2,s5,s3                // Capteurs //
bool re_s1, re_s2;              // Fronts montants des capteurs //
bool fe_s1, fe_s2;              // Fronts descendants des capteurs //
bool d1, d2;                    // Durées //
bool P1;                        // Actions //
bool fs2;                       // Défaut//
int x;
```

Figure 74 : Déclaration des variables

- Deuxièmement, les grafcets sont traduits en équations d'auto-maintien. Pour les déterminer, il faut d'abord définir les fonctions de transition (figure 75) des trois grafcets. Ensuite, les équations d'activation des étapes comme le montre la figure 76. Une déclaration des différentes fonctions de transition est nécessaire.

```
void grafcet ()
{
bool TR_X0_X1,TR_X1_X2,TR_X2_X3,TR_X3_X4,TR_X4_X0;
bool TR_X20_X21,TR_X21_X22,TR_X22_X23,TR_X23_X24,TR_X24_X20;
bool TR_X100_X101, TR_X100_X102, TR_X100_X103, TR_X100_X104,TR_X101_X100, TR_X102_X100, TR_X103_X100, TR_X104_X100;

//(** Fonctions de Transitions **)

   //(* Grafcet GN(P1) *)
TR_X0_X1 :=  X0 and (s3 and s5);
TR_X1_X2 :=  X1 and (fe_s1);
TR_X2_X3 :=  X2 and (re_s2);
TR_X3_X4 :=  X3 and (fe_s2);
TR_X4_X0 :=  X4 and (re_s1);

   //(* Grafcet GF(P1)  *)
TR_X20_X21 :=  X20 and (s3 and s5);
TR_X21_X22 :=  X21 and (fe_s1);
TR_X22_X23 :=  X22 and (d1);
TR_X23_X24 :=  X23 and (d2);
TR_X24_X20 :=  X24 and (re_s1);

   //(* Grafcet GR(P1) *)
TR_X100_X101 := X100 and X2 and fs2;
TR_X100_X102 := X100 and X22 and not(fs2);
TR_X100_X103 := X100 and X3 and fs2;
TR_X100_X104 := X100 and X23 and not(fs2);

TR_X101_X100 := X101 and X22;
TR_X102_X100 := X102 and X2;
TR_X103_X100 := X103 and X23;
TR_X104_X100 := X104 and X3;
```

Figure 75 : Déclaration des fonctions de transitions

```
   //(* Activation des étapes*)

X0 := (TR_X4_X0) or X0 and not(TR_X0_X1);
X1 := (TR_X0_X1) or X1 and not(TR_X1_X2) ;
X2 := (TR_X1_X2) or X2 and not(TR_X2_X3);
X3 := (TR_X2_X3) or X3 and not(TR_X3_X4);
X4 := (TR_X3_X4) or X4 and not(TR_X4_X0) ;

X20 := (TR_X24_X20) or X20 and not(TR_X20_X21);
X21 := (TR_X20_X21) or X21 and not(TR_X21_X22);
X22 := (TR_X21_X22) or X22 and not(TR_X22_X23);
X23 := (TR_X22_X23) or X23 and not(TR_X23_X24);
X24 := (TR_X23_X24) or X24 and not(TR_X24_X20) ;

X100 := ((TR_X101_X100) or (TR_X102_X100) or (TR_X103_X100) or (TR_X104_X100))  or X100 and not((TR_X100_X101) or (TR_X100_X102) or (TR_X100_X103) or (TR_X100_X104));
X101 := (TR_X100_X101) or X101 and not(TR_X101_X100);
X102 := (TR_X100_X102) or X102 and not(TR_X102_X100);
X103 := (TR_X100_X103) or X103 and not(TR_X103_X100);
X104 := (TR_X100_X104) or X104 and not(TR_X104_X100);

   //(*Forçage des Grafcets*)
if (X101){X0:=0; X1:=0; X2:=0;X3:=0;X4:=0; X22:=1;}
if (X102){X20:=0; X21:=0; X22:=0;X23:=0;X24:=0; X2:=1;}
if (X103){X0:=0; X1:=0; X2:=0;X3:=0;X4:=0; X23:=1;}
if (X104){X20:=0; X21:=0; X22:=0;X23:=0;X24:=0; X3:=1;}

}
```

Figure 76 : Fonctions d'activations des étapes

- Enfin, l'affectation des actions est déterminée comme l'illustre la figure 77.

```
void action()
{
P1 := X1 or X2 or X21 or X22;
}
```

Figure 77 : Affectation des actions

La vérification formelle des grafcets consiste à vérifier certaines spécifications φ. Ces dernières peuvent être formulées en logique temporelle (LTL : Logique Temporelle Linéaire, CTL : Computation Tree Logic) (Clarke et al. 1986). LTL et CTL sont des logiques modales où la notion de vérité dépend de l'évolution du monde, c'est-à-dire qu'une proposition peut-être, à un moment, fausse puis, plus tard, devenir vraie. Elles sont définies sur un ensemble de propositions atomiques et combinées par un certain nombre de connecteurs logiques : non, ou, et, implications, vrai et faux ($\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$, true, false), ainsi que d'autres opérateurs temporels appelés « modalités » ou « quantificateurs ». On trouve par exemple :

- **A** ϕ – **A**ll: ϕ doit conserver tous les chemins à partir de l'état actuel.
- **E** ϕ – **E**xists: il existe au moins un chemin partant de l'état actuel et atteignant ϕ.
- **X** ϕ – Ne**x**t: ϕ doit appartenir à l'état suivant.
- **G** ϕ – **G**lobally: ϕ doit appartenir au chemin complet.
- **F** ϕ – **F**inally: ϕ doit appartenir à un élément du chemin.
- ϕ **U** ψ – **U**ntil: ϕ doit appartenir à un élément du chemin jusqu'à ce que ψ aussi.

La vérification du blocage des grafcets distribués de la commande ainsi que le grafcet de la reconfiguration est assurée par la vérification de la condition : Existe-t-il un chemin amenant à une situation de blocage ? Exprimé dans le vérificateur d'UPPAAL par : **E < > deadlock**. Si le résultat de la vérification est satisfait, un blocage est trouvé et une modification des spécifications définies dans des étapes précédentes est nécessaire. Par contre, si le résultat n'est pas satisfait, cela signifie qu'aucun blocage n'est trouvé et que les grafcets sont prêts à être implémentés dans un automate programmable industriel.

La vérification de l'atteignabilité du grafcet demandé suite à une reconfiguration est assurée en vérifiant l'étape active du grafcet de la reconfiguration et son étape forcée correspondante exprimée en UPPAAL par les conditions suivantes :

- [**E < > X101 et non X22]**
- [**E < > X102 et non X2]**
- [**E < > X103 et non X23]**
- [**E < > X104 et non X3]**.

Après une demande de reconfiguration, la vérification qu'un seul grafcet est fonctionnel est nécessaire. Pour cela, il faut vérifier l'existence d'un cas où deux étapes sont actives simultanément dans deux grafcets différents. Cette condition est exprimée par : **[E < > (X0 ou X1 ou X2 ou X3 ou X4) et (X20 ou X21 ou X22 ou X23 ou X24)**].

La même démarche est ensuite appliquée à la totalité des grafcets correspondant à chaque élément du système de transfert.

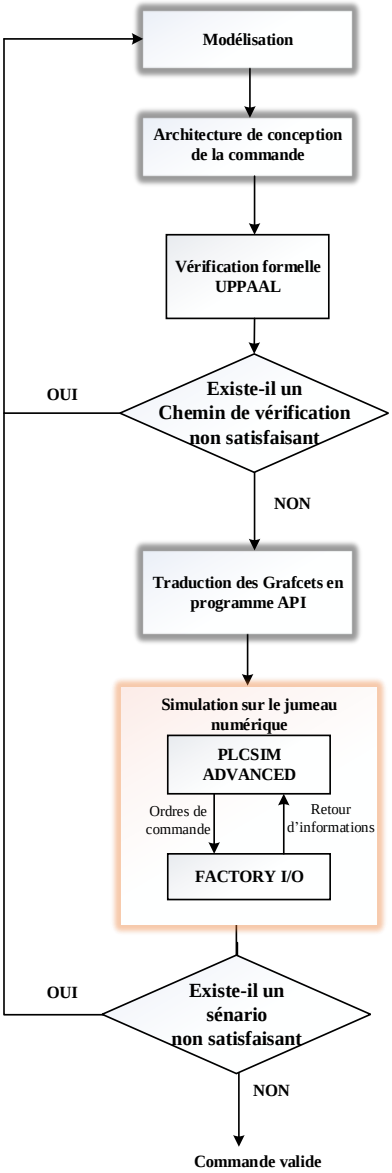


Figure 78 : Architecture pour la simulation de la commande conçue sur factory I/O

L'ensemble des grafcets est traduit en langage de programmation d'API (Ladder Diagram) et implémenté dans un automate virtuel (s7-300) à l'aide du logiciel Step7. Le driver

correspondant à ce type d'automate est ensuite configuré sur Factory I/O (Siemens S7-PLCSIM) (figure 78).

Plusieurs scenarios ont été prédéfinis et simulés afin de répondre au cahier des charges. Parmi ceux-ci, l'un eux nous a permis d'identifier une erreur dans la modélisation d'une spécification globale (La spécification [dcy · s_1=1] a été modifié par [(dcy + s_6) · s_1=1]) afin de prendre en compte le redémarrage du convoyeur après l'évacuation de caisses. Le benchmark réel traité dans le chapitre 4 permettra d'expliquer plus précisément cette étape d'analyses d'erreurs éventuelles.

5.8 Discussion

Pour montrer la pertinence, notre approche est comparée à deux approches de reconfiguration centralisées et conduit aux résultats des tableaux 7 et 8. Le premier tableau présente les résultats de la comparaison entre l'approche de la reconfiguration centralisée (1) basée sur la théorie de contrôle classique et l'approche distribuée. Le second tableau présente les résultats de comparaison entre l'approche centralisée (2) de synthèse de contrôle « raffinée » et l'approche distribuée.

Pour cet exemple didactique simple, les tableaux 7 et 8 montrent que, pour la phase de modélisation de la partie opérative dans le cadre de l'approche centralisée qu'elle soit classique ou raffinée, les modèles obtenus contiennent un nombre élevé d'états et de transitions ce qui n'est pas le cas dans l'approche distribuée où les modèles sont de plus petite taille.

En ajoutant les spécifications de sécurité aux modèles de PO (PO^{N_F} ou PO^{N} et PO^{F}) tout en appliquant le principe de la SCT, le nombre d'états et de transitions diminue. Cela est dû aux états et aux évolutions interdits et supprimés pour garantir la sécurité du système. En revanche, pour l'approche distribuée, aucun changement n'est observé car aucune spécification de sécurité n'est appliquée pour le type d'actionneurs constituant le système étudié.

En appliquant les spécifications de vivacité pour les deux approches centralisées, le nombre d'états pour les modèles des contrôleurs (contrôleur $^{N\_F}$ ou contrôleur^{N} et contrôleur^{F}) est le même nombre d'états de ses modèles de superviseurs correspondants, tandis que le nombre de transitions diminue. Pour l'approche distribuée, l'application de la synthèse locale conduit à une diminution du nombre de transitions pour les contrôleurs locaux, et l'application de la synthèse globale conduit à une diminution à la fois des états et des nombres de transitions. Les contrôleurs distribués sont de petite taille tandis que les contrôleurs globaux calculés pour les deux approches centralisées explosent en nombre d'états et de transitions.

Tableau 7 : Résultats des étapes de la modélisation de l'approche centralisée classique et l'approche distribuée

	Approche centralisée (1)		**Approche distribuée**		
	Etats	**Transitions**		**Etats**	**Transitions**
PO^{N_F}	100	440	**$EPO^{N(P1)}$**	6	10
			$EPO^{N(P2)}$	6	10
			$EPO^{F(P1)}$	10	18
			$EPO^{F(P2)}$	10	18
SUP^{N_F}	75	300	**$SUP^{N(P1)}$**	6	10
			$SUP^{N(P2)}$	6	10

			SUP$^{F(P1)}$	10	18
			SUP$^{F(P2)}$	10	18
Contrôleur^{N_F}	75	251	**Contrôleur distribué$^{N(P1)}$**	4	4
			Contrôleur distribué$^{N(P2)}$	4	4
			Contrôleur distribué$^{F(P1)}$	4	4
			Contrôleur distribué$^{F(P2)}$	4	4
Contrôleur reconfiguré^{N_F}	19200 3648 interdits 4 bloquants	122624	N'existe pas car contrôle distribué		
Grafcet	Difficilement applicable à cause du nombre élevé du contrôleur reconfiguré		**Grafcet$^{N(P1)}$**	5	5
			Grafcet$^{N(P2)}$	5	5
			Grafcet$^{F(P1)}$	5	5
			Grafcet$^{F(P1)}$	5	5
			Grafcet_reconf$^{(P1)}$	5	8
			Grafcet_reconf$^{(P2)}$	5	8

Tableau 8 : Résultats des étapes de la modélisation de l'approche centralisée raffinée et l'approche distribuée

	Approche centralisée (2)		**Approche distribuée**		
	Etats	**Transitions**		**Etats**	**Transitions**
PO^{N}	36	120	**EPO$^{N(P1)}$**	6	10
			EPO$^{N(P2)}$	6	10
PO^{F}	100	360	**EPO$^{F(P1)}$**	10	18
			EPO$^{F(P2)}$	10	18
SUP^{N}	27	72	**SUP$^{N(P1)}$**	6	10
			SUP$^{N(P2)}$	6	10
SUP^{F}	75	240	**SUP$^{F(P1)}$**	10	18
			SUP$^{F(P2)}$	10	18
Contrôleur^{N}	27	26	**Contrôleur distribué$^{N(P1)}$**	4	4
			Contrôleur distribué$^{N(P2)}$	4	4
Contrôleur^{F}	75	191	**Contrôleur distribué$^{F(P1)}$**	4	4
Contrôleur^{N_F}	675	2521			
			Contrôleur distribué$^{F(P2)}$	4	4
Contrôleur reconfiguré^{N_F}	172800	52800	N'existe pas car contrôle distribué		
Grafcet	Difficilement applicable à cause du nombre élevé d'états		**Grafcet$^{N(P1)}$**	5	5
			Grafcet$^{N(P2)}$	5	5
			Grafcet$^{F(P1)}$	5	5

	du contrôleur reconfiguré	**Grafcet$^{F(P1)}$**	5	5
		Grafcet_reconf$^{(P1)}$	5	8
		Grafcet_reconf$^{(P2)}$	5	8

En ajoutant les spécifications de reconfiguration aux contrôleurs globaux, le nombre des états et de transitions constituant les contrôleurs reconfigurés^{N_F} devient très important pour un cet exemple constitué uniquement de deux actionneurs. L'obtention d'un Grafcet du contrôleur global reconfiguré^{N_F} pour les deux approches centralisées s'avère une tâche difficile à cause de leurs modèles complexes par rapport aux modèles obtenus par l'approche distribuée.

C'est parce que l'approche distribuée s'affranchit des étapes de composition entre modèles que l'explosion de l'espace d'états ne se produit pas. Les tableaux 7 et 8 le montrent clairement, aucun des modèles de l'approche préconisée ne dépassant 10 états pour cet exemple simple.

6 Conclusion

Ce chapitre a présenté différentes contributions à la synthèse et la reconfiguration de la commande des SAP.

Tout d'abord, nous avons présenté les problèmes liés aux approches de commande basées sur la SCT et comment cette dernière peut être adaptée pour pallier ses limitations. Pour ce faire, un nouveau cadre pour un contrôle tolérant et reconfigurable des SAP a été proposé. L'idée principale de l'approche est d'assurer la continuité des services du système en cas de défaillance d'un capteur qui peut entraver son comportement normal. Pour atteindre cet objectif, deux modes de fonctionnement sont nécessaires (le mode normal et celui prenant en compte les défauts (dégradé)). Le premier comportement est contrôlé à l'aide d'une synthèse de contrôle distribuée. Alors que le deuxième comportement est contrôlé par une approche étendue où les informations temporelles sont prises en compte. Tout d'abord, les contrôleurs locaux $CL^{N/F}$ sont conçus par une synthèse entre les modèles des $EPO^{N/F}$ et les spécifications correspondant à chaque comportement souhaité. Nous avons proposé de définir une spécification comme une équation logique. Ensuite, des contraintes globales sont définies pour garantir un comportement global où tous les contrôleurs distribués $CD^{N/F}$ peuvent communiquer entre eux. Une fois les $CD^{N/F}$ obtenus, ils sont interprétés dans un langage grafcet pour but d'implémentation dans un API. Au lieu de définir un seul grafcet contenant les deux modes de fonctionnement en commutant entre les deux avec un simple « OU » dans la réceptivité, nous avons opté pour une séparation des modes de fonctionnement et du reconfigurateur (commutateur). Pour passer des modèles Grafcet du comportement normal aux modèles grafcet du comportement dégradé, des contraintes de reconfiguration sont définies et interprétées aussi en grafcet. Cette démarche offre à l'opérateur un visuel clair sur le mode de fonctionnement actif en permettant de visualiser l'aspect switch lié à la reconfiguration.

La commande obtenue est sûre de fonctionnement par construction et assure le contrôle des deux comportements du système, mais elle peut être bloquante. Pour pallier ce problème, nous avons opté pour une vérification des grafcets de commande correspondants au différents $CD^{N/F}$ avant l'implémentation par model-checking. Notre approche implique une sollicitation forte des grafcets de reconfiguration en cas de détection de défauts mais aussi en cas de remise en marche normale 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 mais aussi de l'atteignabilité de chaque étape des grafcets lors de la demande d'une reconfiguration.

Pour cela, avant l'implémentation de l'ensemble de ces grafcets, une méthodologie de vérification de ces propriétés grâce au logiciel UPPAAL a été proposée. Tout d'abord, le non-blocage de l'ensemble des contrôleurs et des reconfigurateurs est vérifié. Ensuite, une vérification de l'atteignabilité des grafcets de commande après une reconfiguration est nécessaire. Si le grafcet est bien atteint à l'étape correspondante, une vérification supplémentaire est testée. Elle consiste à vérifier si deux grafcets de fonctionnement normal et dégradé d'un même EPO sont activés en même temps, ce qui est interdit, car la reconfiguration permet la commutation d'un grafcet normal à celui dégradé tout en désactivant le premier et vis-versa.

L'avantage principal de notre approche est l'utilisation du contrôle distribué qui, d'une part, évite l'explosion combinatoire récurrente dans les approches monolithiques et, d'autre part, permet la reconfiguration du seul EPO défectueux sans reconfigurer la totalité du contrôle du système. De plus, remplacer les événements associés aux capteurs défectueux par des événements temporisés évite l'utilisation d'éléments redondants. En outre, le comportement global est maintenu même si un défaut de capteur apparaît. En effet, le capteur défaillant dans une contrainte globale est remplacé par ses informations temporisées correspondantes qui assurent la continuité de communication entre les différents contrôleurs distribués du mode dégradé. Aussi, une méthode de mise en œuvre de notre contribution est proposée pour un exemple de système didactique. Pour montrer sa pertinence en termes d'utilisabilité, notre approche a été comparée à deux approches de reconfiguration centralisées.

Le chapitre suivant propose l'application de notre méthodologie de conception de la commande reconfigurable sur un système automatisé de production réel.

CHAPITRE 4

Application : Plateforme Cellflex 4.0

1 Introduction

Le chapitre précédent a permis de décrire pas à pas les différentes étapes de la démarche proposée de conception d'une commande reconfigurable et tolérante aux fautes pour les Systèmes Automatisés de Production. Celle-ci a été illustrée pédagogiquement autour d'un exemple simple d'un système de tri simulé. Nous proposons dans ce chapitre d'exploiter toutes les phases de l'approche proposée (figure 79) sur un benchmark avec un système réel disponible au sein du laboratoire CReSTIC de l'URCA, l'intérêt étant que ce benchmark dispose d'un jumeau numérique utilisé pour la phase de vérification.

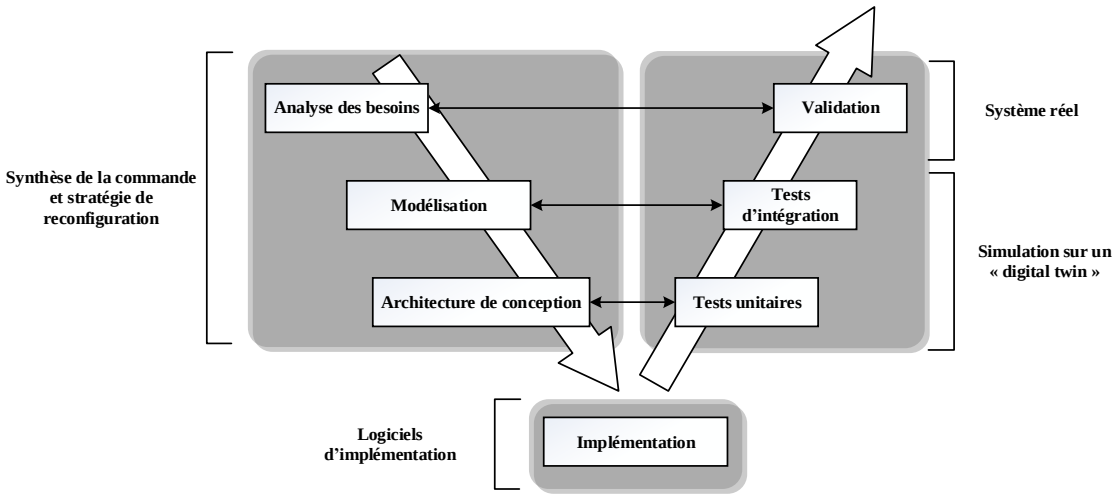


Figure 79 : Cycle de conception d'une commande reconfigurable et tolérante aux fautes

L'URCA dispose de cinq plateformes qui ont vocation à optimiser, valoriser et mettre au service de la communauté une expertise technologique de haut niveau. Ces plateformes sont ouvertes à l'ensemble des équipes de l'URCA, aux équipes académiques extérieures à l'URCA et à la communauté industrielle. Parmi celles-ci, « CellFlex4.0 »[9] est une plateforme de formation et de recherche en lien direct avec les concepts d'Industrie 4.0. Elle regroupe notamment (Figure 80) : une plate-forme multi-énergies renouvelables alimentant un atelier flexible d'embouteillage et de packaging (CellFlex) disposant de son jumeau numérique (NXMCD[10]), l'ensemble est complété par des outils logiciels pour la simulation, l'émulation et la virtualisation de systèmes (Factory I/O et Home I/O[11], Emulate3D[12], CIROS[13]).

[9] https://www.univ-reims.fr/meserp/accueil/plateau-de-formation-et-de-transfert-technologique,9485,27019.html
[10] https://www.plm.automation.siemens.com/global/fr/products/mechanical-design/mechatronic-concept-design.html
[11] https://realgames.co
[12] https://www.demo3d.com/
[13] https://www.festo-didactic.com/be-fr/produits/logiciels/ciros/?fbid=YmUuZnIuNTM1LjE2LjIwLjExMTA

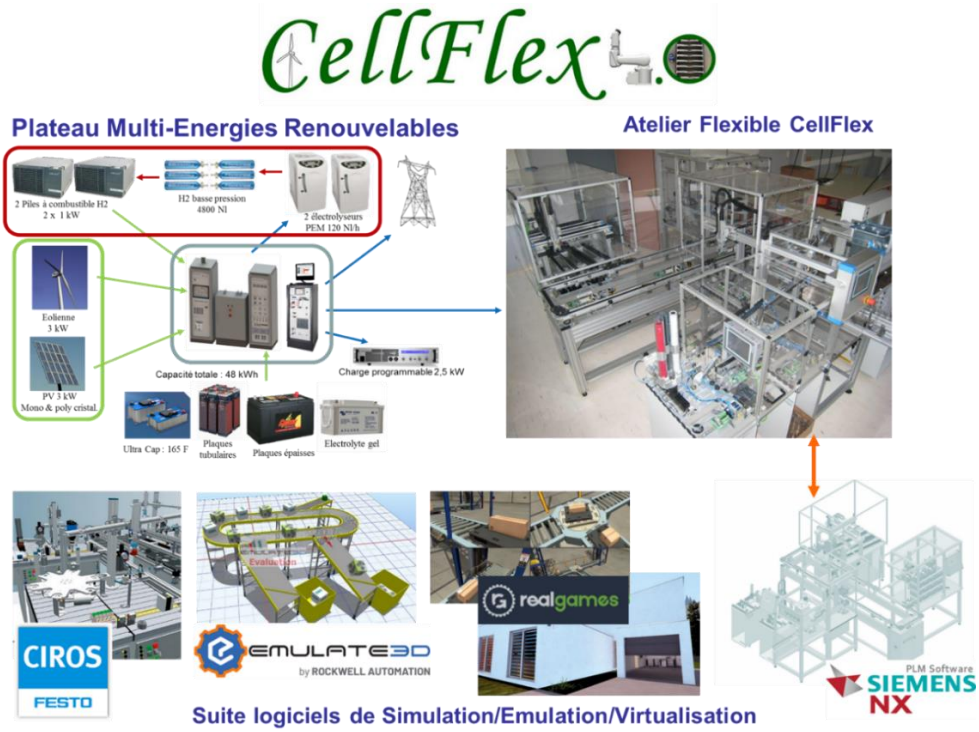


Figure 80 : Plateforme CellFlex4.0

L'atelier flexible de la plateforme permet de mettre en œuvre et d'illustrer les différents flux d'informations et de produits circulant au sein d'une entreprise industrielle moderne. Cette cellule de production est orientée « emballage et conditionnement » et dispose de matériel et de logiciels de contrôle-commande et de supervision industrielle. Elle est issue de l'axe ICOS du CPER 2007-2013 et s'inscrit actuellement dans le cadre du projet FFCA (Factories of Future Champagne-Ardenne) du CPER 2017-2022.

Dans le cadre de cette thèse, l'atelier est exploité en partie pour la validation des propositions et son jumeau numérique sous NX MCD de Siemens est utilisé pour la phase de tests. Ce passage à l'échelle de notre contribution permet de discuter des avantages et des inconvénients de l'approche et de mesurer ses performances et ses limites.

Dans ce chapitre, l'atelier flexible est tout d'abord décrit dans sa globalité puis détaillé au niveau de la station de bouchonnage. La section 3 reprend les différents modèles de la station. Les étapes de synthèse sont reprises en section 4 et 5 avant l'interprétation des contrôleurs distribués en grafcet (section 6). La section 7 concerne la modélisation du reconfigurateur qui permettra ainsi de passer à la phase de vérification formelle des modèles avant implémentation (section 8). La section 9 consiste à tester le code à implémenter par « virtual commissionning » à l'aide d'une émulation sur le jumeau numérique du système. Enfin, le code est validé sur le système réel en section 10.

2 Description de l'atelier flexible

L'atelier flexible consiste à remplir des bouteilles pour les conditionner en lots de 6 (appelé sixpack), les stocker ou les exporter. Il est distribué en 6 stations et 1 convoyeur central illustrées en figure 81. Fonctionnellement, la station d'import-export fournit des sixpacks vides sur le convoyeur central qui relie chaque poste de travail. Pendant ce temps, des bouchons sont acheminés vers la station d'embouteillage qui a été alimentée d'un mélange de matière première

effectué par la station de mixage. La station d'embouteillage permet de disposer de bouteilles remplies, bouchonnées et tracées par puces RFID. Ces dernières sont transportées vers la station de transfert qui, lorsque six bouteilles sont disponibles, les dépose dans un sixpack vide. Ce sixpack plein est alors soit stocké (station de stockage), soit envoyé à la station d'import/export pour récupération par l'opérateur. La station de transfert dispose également d'un système de vision permettant d'identifier des problèmes de conditionnement. Les bouteilles sont alors réacheminées vers une cellule robotisée afin de (i) retirer le bouchon pour le remettre dans le stock de la station de bouchonnage, (ii) vider la bouteille pour retraiter le produit au niveau de la station de mixage, (iii) réinjecter la bouteille vide dans la station d'embouteillage.

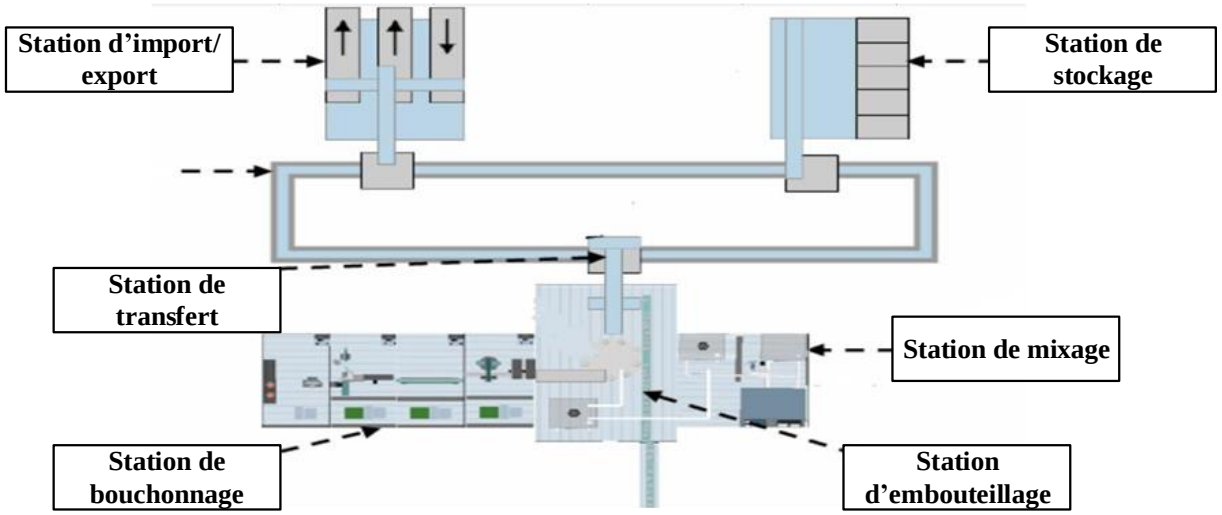


Figure 81 : Les stations constituant la plate-forme CellFlex 4.0

L'ensemble est piloté par 6 automates programmables industriels de type Siemens S7-1500 (avec ou sans E/S déportés) et dispose d'interfaces homme machine permettant un contrôle global ou local de l'atelier.

Dans ce chapitre, l'approche de conception de la commande reconfigurable est appliquée uniquement à la station de bouchonnage. L'atelier étant un système distribué communiquant entre station par Profinet. La station de bouchonnage communique par exemple avec la station d'embouteillage par l'intermédiaire d'une variable d'échange sur la présence ou non de bouteilles pour la dépose d'un bouchon. La station de bouchonnage (figure 82) est considérée comme étant un SAP distribué constituée de 8 actionneurs contrôlés par différentes technologies et 15 détecteurs constituants :

- Un magasin de bouchons de 3 couleurs (blanc, noir, rouge).
- Un dispositif pour les évacuer.
- Un vacuostat avec aspiration et expiration pour la préhension des bouchons.
- Un convoyeur tampon qui permet d'acheminer les bouchons vers le bras manipulateur.
- Un bras manipulateur qui, par l'intermédiaire d'une pince, transfère les bouchons sur les bouteilles de la station d'embouteillage.

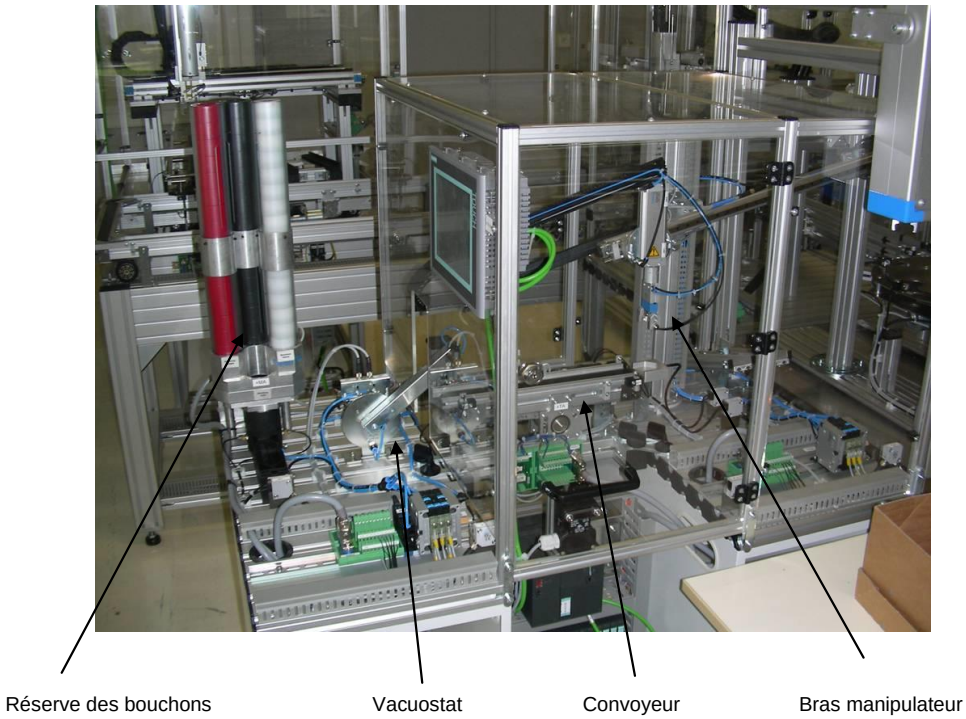


Figure 82 : Station de bouchonnage

Remarque : Le jumeau numérique de cette station ne représente actuellement que le magasin de bouchons blancs, le traitement des bouchons noirs et rouges ne sera donc pas pris en compte ici.

Les entrées et les sorties de l'API commandant la station d'acheminement des bouchons sont représentées respectivement dans les tableaux 9 et 10.

Tableau 9 : Les sorties de l'API de la station de bouchonnage

Evénement	Actionneur	Description
VBB	Vérin bouchon blanc	Rentrée du vérin des bouchons blancs (pilotage monostable)
EJ	Ejecteur	Sortie du vérin d'éjection (pilotage monostable)
VTAS	Ventouse	Aspiration d'air de la ventouse
VTEX		Expiration d'air de la ventouse
VRM	Vérin rotatif	Déplacement du vérin rotatif vers le magasin
VRC		Déplacement du vérin rotatif vers le convoyeur
CONV	Convoyeur	Rotation du convoyeur
BMC	Bras de manipulation	Déplacement du bras de manipulation vers le convoyeur

BME		Déplacement du bras de manipulation vers l'embouteillage
PINCES	Pince	Ouverture des pinces (pilotage monostable)
DVL	Vérin de levage	Descente du vérin de levage (pilotage monostable)

Tableau 10 : Les entrées de l'API de la station de bouchonnage

Capteur	Description
c_vbb	Vérin des bouchons blancs rentré
pm	Magasin vide
cer	Vérin d'éjection rentré
ces	Vérin d'éjection sorti
c_vrc	Vérin rotatif sur le convoyeur
c_vrm	Vérin rotatif sur le magasin
c_vt	Ventouse sous pression
dconv	Présence d'un bouchon au début du convoyeur (en logique inverse)
fconv	Présence d'un bouchon à la fin du convoyeur (en logique inverse)
c_bmc	Bras de manipulation côté convoyeur
c_bme	Bras de manipulation côté embouteillage
vlb	Vérin de levage en position basse
vlh	Vérin de levage en position haute
recept	Présence d'un bouchon sur le réceptacle à la fin du convoyeur
bp	Information de la station d'embouteillage sur la présence d'une bouteille

3 Modélisation du système

La partie opérative de la station de bouchonnage est constituée de 8 actionneurs :

- Un vérin simple effet VSE (*vérin bouchon blanc*) possédant un capteur de fin de course actif lorsque le vérin est rentré. Pour faire tomber un bouchon, il faut rentrer le vérin complètement, le relâcher, puis le désactiver lorsque le capteur du vérin indique que celui-ci est rentré.
- Un vérin simple effet VSE (*éjecteur*) entouré de trois capteurs : deux capteurs de fin de course (position rentrée et position sortie) et un capteur à infrarouge qui détecte la présence d'un bouchon devant ce vérin.
- Un vérin double effet VDE (*bras rotatif*) possédant deux détecteurs de présence du bras (côté magasin et côté convoyeur).

- Un actionneur pour l'aspiration et l'expiration d'air (*ventouse*) possédant un capteur de pression pour détecter les bouchons aspirés.
- Un *convoyeur* à bande, actionné par un moteur à un sens de rotation associé à deux capteurs de présence bouchon en début et en fin de convoyeur et un capteur de présence bouchon sur le réceptacle.
- Un vérin double effet VDE (*bras de manipulation*) possédant deux capteurs de position en zone convoyeur et en zone embouteillage.
- Un vérin simple effet VSE (*bras de levage*) possédant deux capteurs de positions haute et basse.
- Une *pince* qui n'est associé à aucun capteur.

Les modèles des différents éléments de PO sont obtenus par la conception de leurs modèles pratiques résultant d'une composition synchrone des modèles des détecteurs et leurs modèles d'actionneurs correspondants.

- Les modèle pratiques du vérin de bouchons blancs (figure 83.a) et du convoyeur (figure 83.f) sont présentés sous forme d'un automate à 4 états.
- Les modèles correspondant à l'éjecteur (figure 83.b) et au bras de levage (figure 83.c) sont présentés sous forme d'un automate à 6 états.
- Le modèle pratique de la ventouse (figure 83.e) est présenté sous forme d'un automate à 10 états.
- Les deux modèles correspondant au vérin rotatif (figure 83.g) et au bras de manipulation (figure 83.h) sont présentés par un automate à 15 états.
- Le modèle de la pince (figure 83.d) présenté par un automate à deux états n'a que la structure « actionneur » car il n'a pas de détecteur associé.

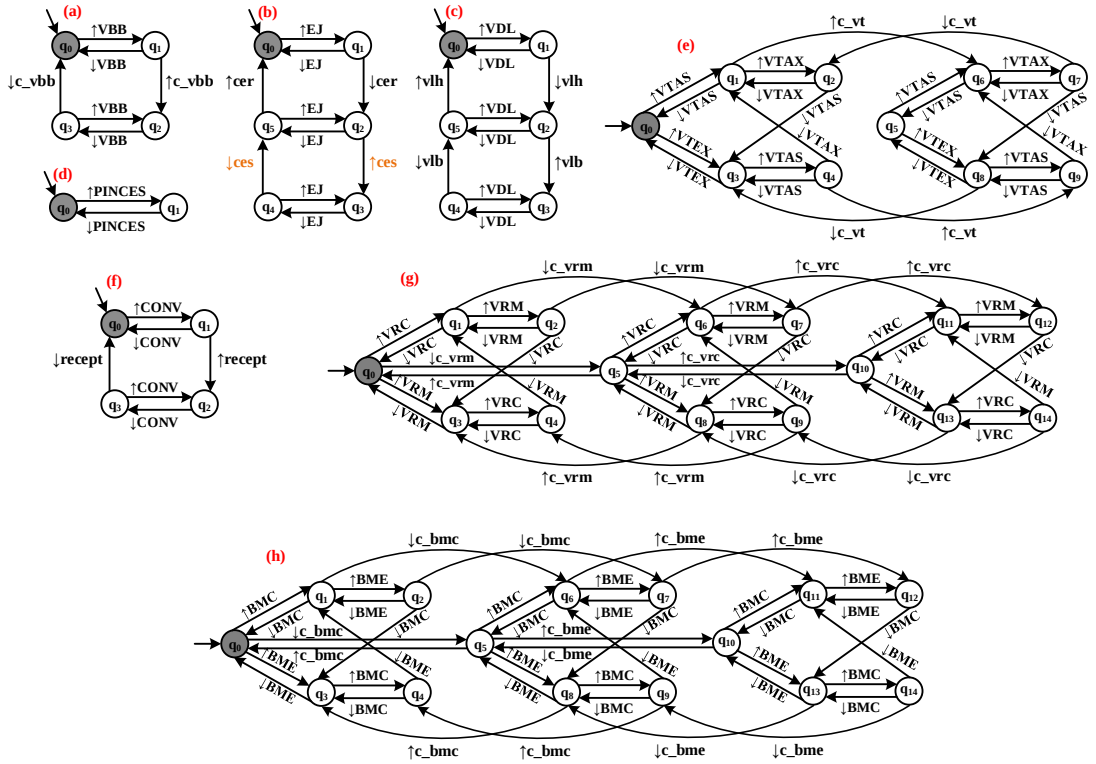


Figure 83 : Modèles pratiques des comportements normaux des EPO de la station de bouchonnage : (a) vérin des bouchons blancs, (b) éjecteur, (c) bras de levage, (d) pince, (e) ventouse, (f) convoyeur, (g) vérin rotatif, (h) bras de manipulation

Dans cette application, nous intégrons un défaut au niveau du capteur de fin de course en position sortie (*ces*). Nous associons à l'activation de ce capteur (↑ces) une horloge ck_1, et à sa désactivation (↓ces) une horloge ck_2. Le modèle pratique de cet élément de PO devient alors un modèle temporisé présenté par la figure 84 constitué de 8 états.

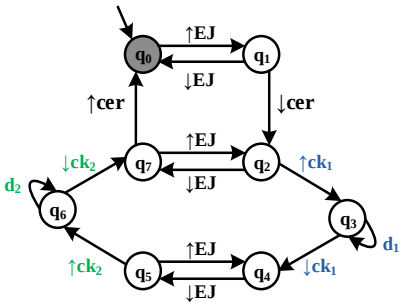


Figure 84 : Modèle pratique de comportement dégradé de l'éjecteur

4 Synthèse de la commande locale

Le tableau 11 présente l'ensemble des contraintes locales de sécurité et de vivacité (établies par l'expert) à appliquer aux différents EPO constituant la station d'acheminement des bouchons.

Par exemple, le comportement normal de l'éjecteur correspond à la sortie de la tige ↑EJ jusqu'à l'activation du capteur *ces*. Ensuite, cette action de sortie doit être inhibée ↓EJ pour rétracter le vérin jusqu'à ce que le capteur *cer* soit activé. Pour éviter des transitions dans le modèle qui s'écartent de ce comportement, les contraintes de vivacité ont été définies en mode normal par les deux équation (3) et (4). La spécification exprimée par l'équation (3) traduit le fait que l'ordre de sortie EJ ne peut être inhibé qu'à l'état q_3 c'est-à-dire après l'activation du capteur *ces* (vérin en position sortie). Tandis que la spécification exprimée par l'équation (4) reflète le fait que l'ordre de sortie EJ n'est autorisé qu'à l'état initial q_0. La même spécification (3) est maintenue pour le comportement dégradé (19) de cet élément. En revanche, la spécification (4) est modifiée puisque le modèle de comportement dégradé a changé de structure par rapport au comportement normal. En effet, le nombre d'états du modèle a augmenté de deux états suite à l'introduction des événements temporisés. La spécification se définit alors par l'équation (20).

Tableau 11 : Spécifications locales de vivacité et de sécurité

Elément de partie opérative	Spécifications locales pour le mode normal	Spécifications locales pour le mode dégradé
Vérin bouchon blanc	$(1): q_1.\downarrow VBB = 0$ $(2): q_3.\uparrow VBB = 0$	-
Ejecteur	$(3): (q_1 + q_2).\downarrow EJ = 0$ $(4): (q_4 + q_5).\uparrow EJ = 0$	$(19): (q_1 + q_2).\downarrow EJ = 0$ $(20): (q_5 + q_7).\uparrow EJ = 0$
Ventouse	$(5): q_4.\downarrow VTEX = 0$ $(6): q_0.\uparrow VTEX = 0$ $(7): q_1.\downarrow VTAS = 0$	-

	(8): $q_3.\uparrow VTAS = 0$	
Vérin rotatif	(9): $(q_5 + q_6).\downarrow VRM = 0$ (10): $(q_0 + q_8).\uparrow VRM = 0$ (11): $(q_1 + q_2).\downarrow VRC = 0$ (12): $(q_4 + q_8).\uparrow VRC = 0$	-
Convoyeur	(13): $q_1.\downarrow CONV = 0$ (14): $q_3.\uparrow CONV = 0$	-
Vérin de levage	(13): $(q_1 + q_2).\downarrow DVL = 0$ (14): $(q_4 + q_5).\uparrow DVL = 0$	-
Bras de manipulation	(15): $(q_5 + q_6).\downarrow BMC = 0$ (16): $(q_0 + q_8).\uparrow BMC = 0$ (17): $(q_1 + q_2).\downarrow BME = 0$ (18): $(q_4 + q_8).\uparrow BME = 0$	-
Pince	-	-

L'ensemble de ces spécifications reflète la sûreté fonctionnelle de la station d'acheminement des bouchons. La synthèse des contrôleurs par la SCT des modèles des EPO avec les équations des spécifications locales correspondantes permet l'obtention des contrôleurs locaux des deux modes : normal et dégradé.

Pour synthétiser les contrôleurs locaux, les spécifications définies dans le tableau ci-dessus (tableau 11) sont interprétées sous forme de machine à états finis étendus (figure 85) comme expliqué dans le chapitre 3.

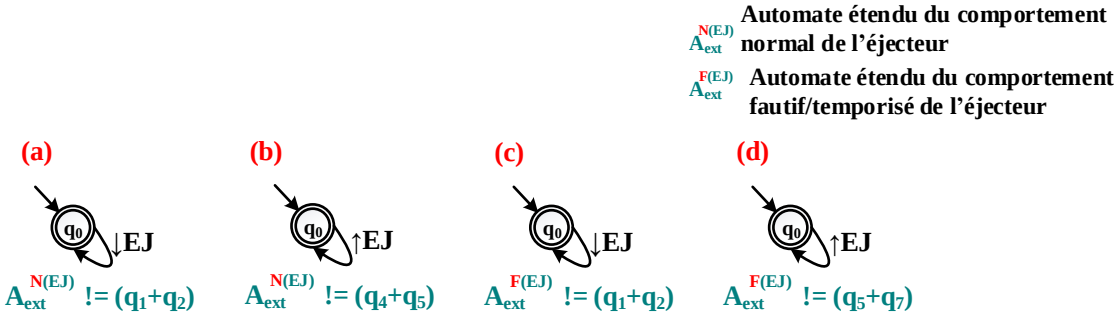


Figure 85 : Extrait de l'ensemble des spécifications de fonctionnement local de la station : (a) spécification « 3 », (b) spécification « 4 », (c) spécification « 19 », (d) spécification « 20 »

En utilisant la synthèse de contrôle basée sur la SCT, nous obtenons les contrôleurs locaux $CL^{N/F}$. Chacun d'eux décrit le comportement local souhaité de chaque EPO selon des comportements normaux pour tous les EPO (figure 86) et dégradé pour l'éjecteur (figure 87).

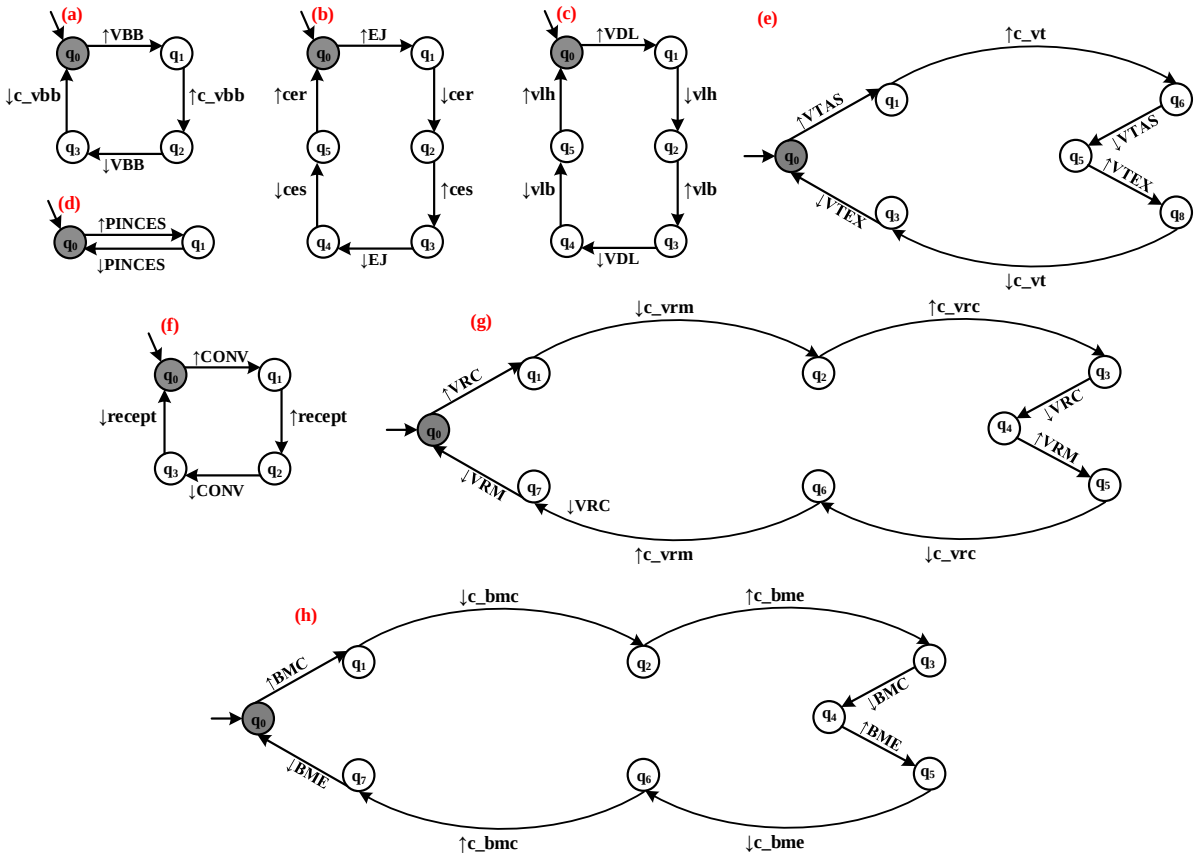


Figure 86 : Les contrôleurs locaux de comportement normal des EPO : (a) vérin des bouchons blancs, (b) éjecteur, (c) bras de levage, (d) pince, (e) ventouse, (f) convoyeur, (g) vérin rotatif, (h) bras de manipulation

Par exemple, la prise en compte de la spécification (3) pour le comportement normal et (19) pour le comportement dégradé permet la suppression des transitions associées à l'événement ↓EJ sortantes des états q_1 et q_2. Tandis que la prise en compte de la spécification (4) permet la suppression des transitions associées à l'événement ↑EJ sortantes des états q_4 et q_5 pour le comportement normal et celles sortantes des états q_5 et q_7 pour le comportement dégradé.

De la même façon, en appliquant les spécifications de vivacité et de sécurité aux modèles des EPO correspondant, nous obtenons les différents contrôleurs locaux $CL^{N/F}$ (figure 87).

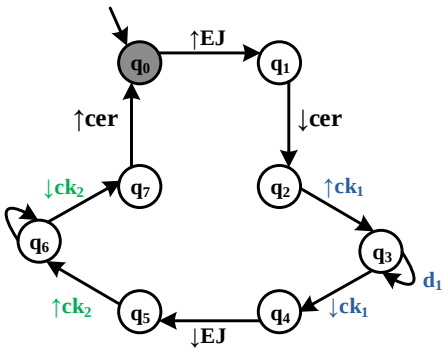


Figure 87 : Contrôleur local de comportement dégradé de l'éjecteur

5 Synthèse de la commande globale

Les contrôleurs présentés dans les figures 87 et 86 agissent d'une manière locale. Pour assurer le fonctionnement et le contrôle global du système où ces contrôleurs peuvent communiquer entre eux, les spécifications globales doivent être définies et intégrées.

Nous distinguons 15 spécifications informelles à prendre en compte pour les différents contrôleurs locaux $CL^{N/F}$:

1) Le vérin distribuant les bouchons blancs ne peut être actionné que si le magasin est vide.
2) Après la rentrée du vérin des bouchons blancs, ce dernier ne doit avancer qu'après une durée T permettant au bouchon de tomber. En pratique, T est la valeur logique récupérée à la sortie du bloc de temporisation « TON » sur Siemens. T=1 si la durée (T#1s) prédéfinie à l'entrée de « TON » est écoulée.
3) L'éjecteur ne doit être actionné que si un bouchon est présent dans le magasin et le vérin rotatif est sur le côté convoyeur.
4) L'éjecteur ne doit reculer que si un bouchon est aspiré.
5) La ventouse ne doit aspirer le bouchon que si le vérin rotatif est du côté magasin et que l'éjecteur est en position sortie.
6) La ventouse ne doit expirer l'air que si le vérin rotatif est sur le convoyeur.
7) Le vérin rotatif ne doit se déplacer vers le convoyeur que si un bouchon est aspiré ou aucun bouchon n'est aspiré et n'est présent au début du convoyeur.
8) Le vérin rotatif ne doit se déplacer vers le magasin que si un bouchon est présent au début du convoyeur ou que le vérin éjecteur atteint la position sortie.
9) Le convoyeur est actionné si un bouchon est présent au début du convoyeur et que le vérin rotatif est sur le côté magasin.
10) Le vérin de levage est actionné si un bouchon est présent sur le réceptacle à la fin du convoyeur et que le bras de manipulation est sur le côté convoyeur, ou que le bras de manipulation est sur le côté embouteillage et qu'une information fournie par la station d'embouteillage est délivrée.
11) Le vérin de levage est désactivé après 1s après que celui-ci atteint sa position basse.
12) Le bras de manipulation peut être déplacé vers l'embouteillage si le vérin de levage atteint sa position haute
13) Le bras de manipulation peut être déplacé vers le convoyeur si le vérin de levage atteint sa position haute
14) La pince peut être actionnée si le bras de manipulation est sur le côté convoyeur et le vérin de levage quitte sa position haute, ou si le bras de manipulation est sur le côté embouteillage et le vérin de levage atteint sa position basse.
15) La pince peut être désactivée si le bras de manipulation est sur le côté convoyeur et le vérin de levage est en position basse ou si le bras de manipulation est sur le côté embouteillage et le vérin de levage quitte sa position passe.

Le tableau 12 présente une détermination formelle des spécifications globales sous forme d'équations logiques booléennes. Par exemple, l'autorisation de l'éjection (Ord EJ) est assurée par la vérification de la spécification (*pm=0*), c'est-à-dire un bouchon est présent dans le magasin pour être éjecté.

Tableau 12 : Spécifications de sécurité et vivacité globales

Elément de PO	Condition Si	Alors
Vérin Bouchon Blanc	$pm = 1$	Ord VBB
	T=1	Inh VBB
Ejecteur	$c\_vrc.\overline{pm} = 1$	Ord EJ
	$c\_vt = 1$	Inh EJ
Ventouse	$c\_vrm.ces = 1$	Ord VTAS
	$c\_vrc = 1$	Ord VTEX
Vérin rotatif	$c\_vt + dconv.\overline{c\_vt} = 1$	Ord VRC
	$\uparrow ces + \overline{dconv} = 1$	Ord VRM
Convoyeur	$c\_vrm.\overline{dconv} = 1$	Ord CONV
Vérin de levage	$recept.c\_bmc + pb.\uparrow c\_bme = 1$	Ord DVL
	T=1	Inh DVL
Bras de manipulation	$\uparrow vlh = 1$	Ord BME
	$\uparrow vlh = 1$	Ord BMC
Pince	$c\_bmc.\downarrow vlh + c\_bme.\uparrow vlb = 1$	Ord PINCES
	$c\_bmc.vlb + c\_bme.vlh = 1$	Inh PINCES

Pour synthétiser les contrôleurs distribués $CD^{N/F}$, les spécifications globales présentées ci-dessus sont appliquées aux contrôleurs locaux $CL^{N/F}$. Comme nous l'avons détaillé dans le chapitre 3, l'obtention des $CD^{N/F}$ est assurée par l'algorithme de synthèse qui consiste à agréger les contrôleurs locaux et ensuite à intégrer les spécifications globales aux contrôleurs agrégés. La figure 88 illustre les modèles de contrôleurs distribués des comportements normaux des différents EPO constituant la station de bouchonnage, tandis que la figure 89 présente le modèle du contrôleur distribué du comportement dégradé de l'éjecteur.

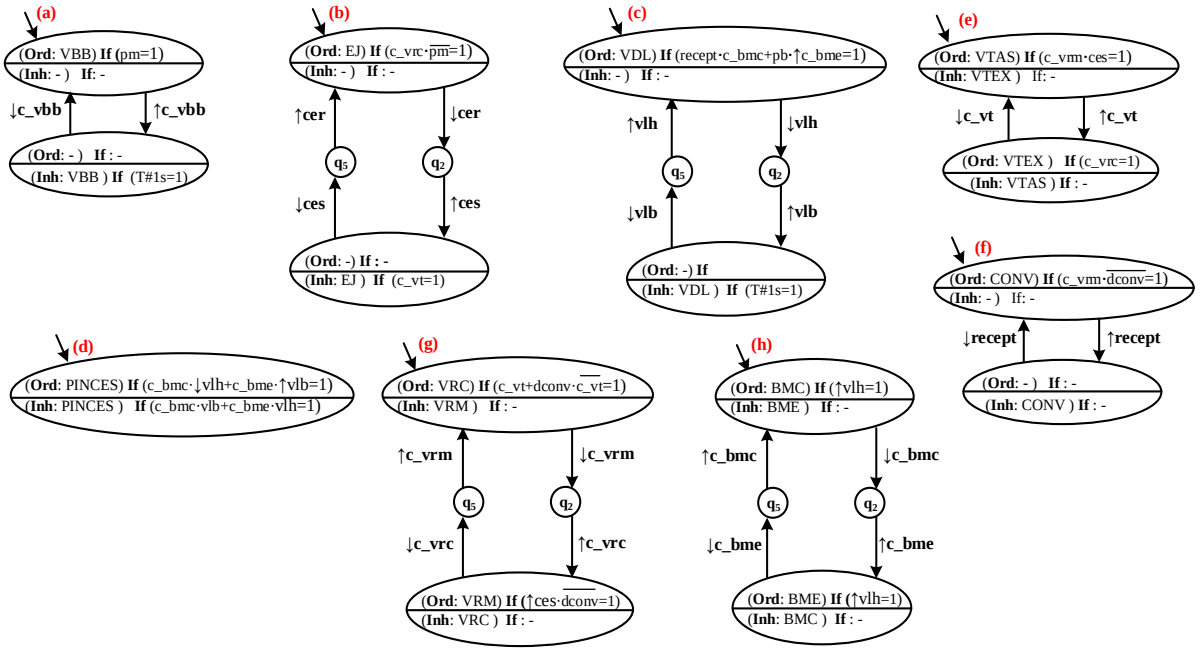


Figure 88 : Contrôleurs distribués des comportements normaux des différents EPO : (a) vérin des bouchons blancs, (b) éjecteur, (c) bras de levage, (d) pince, (e) ventouse, (f) convoyeur, (g) vérin rotatif, (h) bras de manipulation

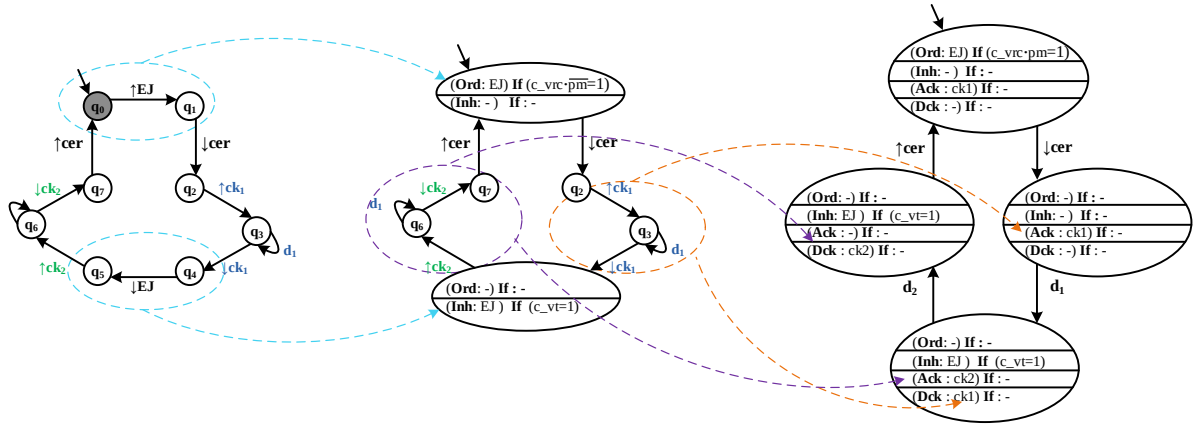


Figure 89 : Etapes d'obtention du contrôleur distribué du comportement dégradé de l'éjecteur

Pour obtenir le contrôleur distribué du comportement dégradé de l'éjecteur (figure 89), la première agrégation consiste à agréger l'ordre d'autorisation de l'éjecteur EJ dans l'ensemble d'autorisations *Ord* et l'inhibition de l'ordre de sortie de l'éjecteur dans l'ensemble des inhibition *Inh* en associant la spécification globale pour chaque ordre agrégé. La deuxième agrégation consiste à agréger les événements associés aux activations des horloges ck_1 et ck_2 dans l'ensemble A_{ck}, et les événements associés aux désactivations de ces horloges dans l'ensemble D_{ck}.

6 Interprétation des contrôleurs distribués en grafcet

En appliquant les règles d'interprétation d'un automate au grafcet comme nous l'avons présenté dans la section 3 du chapitre 3 aux contrôleurs distribués $CD^{N/F}$ de la station d'acheminement des bouchons étudiée précédemment, nous obtenons les grafcets de

commande des comportements normaux présentés dans la figure 90 et de comportement dégradé de l'éjecteur présenté dans la figure 91.

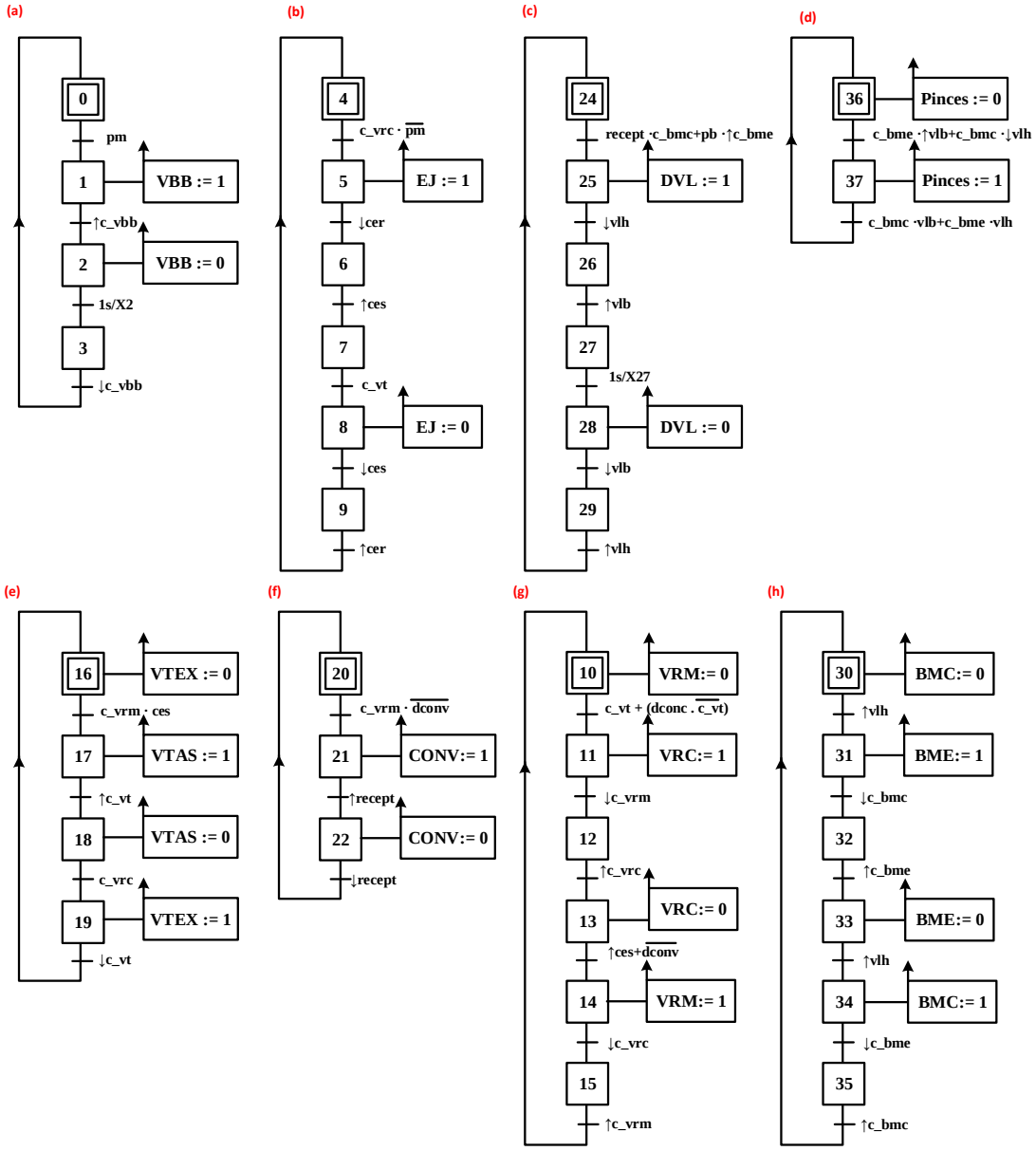


Figure 90 : Grafcets de commande de comportements normaux des différents EPO : (a) vérin des bouchons blancs, (b) éjecteur, (c) bras de levage, (d) pince, (e) ventouse, (f) convoyeur, (g) vérin rotatif, (h) bras de manipulation

L'obtention du grafcet de commande de l'éjecteur du mode dégradé est assurée par l'interprétation de l'automate du contrôleur du mode de comportement dégradé présenté dans la figure 89. L'état initial de l'automate est traduit par une étape initiale au niveau du grafcet, la spécification globale sur l'autorisation de l'ordre de l'éjection est traduite par une réceptivité de transition juste après l'étape initiale. L'autorisation de l'ordre de l'éjection est traduite par une action associée à l'étape qui suit la transition contenant la spécification globale. De la même manière, l'inhibition de l'ordre de l'éjection et sa spécification globale correspondante sont traduites au niveau du grafcet.

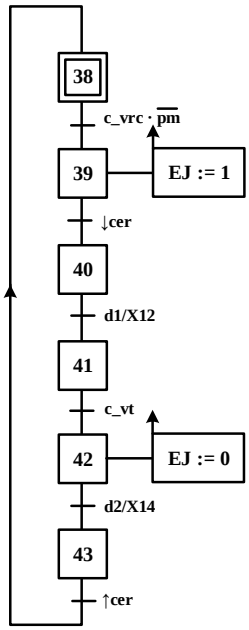


Figure 91 : Grafcet de commande du comportement dégradé de l'éjecteur

7 Modélisation du reconfigurateur

L'élément défectueux de cette station est le capteur *ces* associé à l'éjecteur. Pour assurer la reconfiguration de cet actionneur lors de la détection du défaut *fs*, une règle de reconfiguration doit être déterminée pour garantir une commutation entre les grafcets de comportement normal et dégradé.

Le passage de $G^{N(EJ)}$ à $G^{F(EJ)}$ et vice versa est basé sur les deux équations suivantes :

Equation 1 : RC_1 : **Si** X_6 **ET** $f_s = 1$ **Alors**

(F: $G^{F(EJ)}\{X_{40}\}$) **ET** (F: $G^{N(EJ)}\{\}$)

Sinon Si X_{40} **ET** $f_s = 0$ **Alors**

(F: $G^{N(EJ)}\{X_6\}$) **ET** (F: $G^{F(EJ)}\{\}$)

Equation 2 : RC_2 : **Si** X_8 **ET** $f_s = 1$ **Alors**

(F: $G^{F(EJ)}\{X_{42}\}$) **ET** (F: $G^{N(EJ)}\{\}$)

Sinon Si X_{42} **ET** $f_s = 0$ **Alors**

(F: $G^{N(EJ)}\{X_8\}$) **ET** (F: $G^{F(EJ)}\{\}$)

La traduction des deux équations en grafcet permet l'obtention du modèle reconfiguration de l'éjecteur présenté par le grafcet de la figure 92.

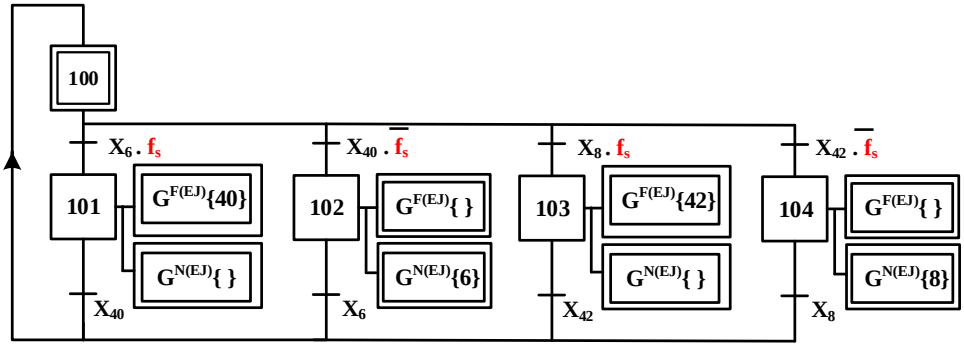


Figure 92 : Interprétation en grafcet des contraintes de reconfiguration CR1(EJ) et CR2(EJ) (Modèle du reconfigurateur)

8 Modèles et programmes de vérification

Dans cette section nous appliquons la méthode de vérification proposée dans la section (4.6.2) du chapitre 3 sur tous les grafcets de commande des EPO constituants la station d'acheminement des bouchons.

Les grafcets à programmer sont les grafcets des comportements normaux de tous les EPO, le grafcet de comportement dégradé de l'éjecteur, et le grafcet de reconfiguration assurant la commutation entre les deux modes de comportement de l'éjecteur.

Pour rappel, le cycle automate est exécuté selon un modèle composé de 4 phases, comme le montre la figure 93.

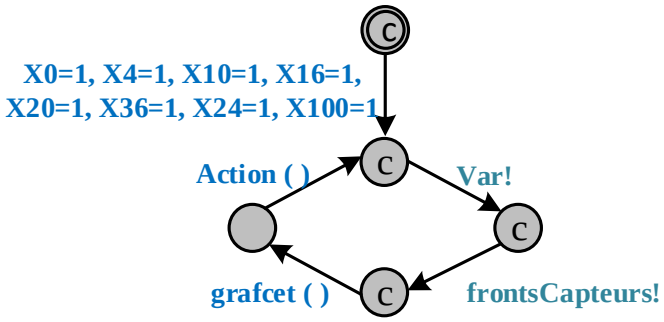


Figure 93 : Modélisation du cycle automate

- Lecture des entrées (**var!**)
- Lecture des fronts montants et descendants des différents capteurs constituant la station de bouchonnage (**frontsCapteurs!**)
- Exécution du programme principal (**Grafcet ( )**)
- Mise à jour des sorties de la station (**action ( )**)

Le modèle de cycle API permet une synchronisation de tous les modèles établis de la station de bouchonnage tout en respectant l'ordre d'exécution séquentielle des tâches cycliques du programme API. Lors du tout premier cycle automate, une phase d'initialisation est nécessaire pour initialiser les programmes.

La modélisation en ST appliquée à l'ensemble de ces grafcets est donnée par la méthode suivante :

- Premièrement, toutes les variables sont déclarées sur UPPAAL (figure 94).

```
bool X0,X1,X2,X3,X4,X5,X6,X7,X8,X9,              // Etapes du Grafcet normal //
     X10,X11,X12,X13,X14,X15,X16,X17,X18,X19,
     X20,X21,X22,X24,X25,X26,X27,X28,X29,
     X30,X31,X32,X33,X34,X35,X36,X37,
     X38,X39,X40,X41,X42,X43,                    // Etapes du Grafcet temporisé //
     X100,X101,X102,X103,X104;                   // Etapes du Grafcet de reonfiguration //

bool c_vbb,pm,pb,cer,ces,c_vrc,c_vrm,c_vt,
     dconv,fconv,c_bmc,c_bme,vlb,vlh,recept;     // Capteurs //


bool Re_c_vbb,Re_pm,Re_cer,Re_ces,Re_c_vrc,Re_c_vrm,Re_c_vt,
     Re_dconv,Re_fconv,Re_c_bmc,Re_c_bme,Re_vlb,Re_vlh,Re_recept;  // Fronts montants des capteurs //

bool Fe_c_vbb,Fe_pm,Fe_cer,Fe_ces,Fe_c_vrc,Fe_c_vrm,Fe_c_vt,
     Fe_dconv,Fe_fconv,Fe_c_bmc,Fe_c_bme,Fe_vlb,Fe_vlh,Fe_recept;  // Fronts descendants des capteurs //
```

Figure 94 : Extrait de la déclaration des variables des grafcets sous UPPAAL

- Deuxièmement, les grafcets sont traduits en équations d'auto-maintien. Pour les déterminer, il faut d'abord définir les fonctions de transition (figure 96) de l'ensemble des grafcets. Ensuite, les équations d'activation des étapes comme le montre la figure 97. Une déclaration des différentes fonctions de transition est nécessaire (figure 95).

```
bool TR_X38_X39,TR_X39_X40,TR_X40_X41,TR_X41_X42,TR_X42_X43,TR_X43_X38;

bool TR_X100_X101, TR_X100_X102, TR_X100_X103, TR_X100_X104,TR_X101_X100,
     TR_X102_X100, TR_X103_X100, TR_X104_X100;
```

Figure 95 : Extrait de la déclaration des fonctions de transition pour le grafcet du mode dégradé et de reconfiguration de l'éjecteur

```
   //(* Grafcet GF(EJ)  *)

TR_X38_X39 :=  X38 and ((c_vrc) and not(pm));
TR_X39_X40 :=  X39 and (Fe_cer);
TR_X40_X41 :=  X40 and (d1);
TR_X41_X42 :=  X41 and (c_vt);
TR_X42_X43 :=  X42 and (d2);
TR_X43_X38 :=  X43 and (Re_ces);


   //(* Grafcet GR(EJ) *)

TR_X100_X101 := X100 and X6 and fs;
TR_X100_X102 := X100 and X40 and not(fs);
TR_X100_X103 := X100 and X8 and fs;
TR_X100_X104 := X100 and X42 and not(fs);

TR_X101_X100 := X101 and X40;
TR_X102_X100 := X102 and X6;
TR_X103_X100 := X103 and X8;
TR_X104_X100 := X104 and X42;
```

Figure 96 : Extrait de la détermination des fonctions de transition du grafcet du mode dégradé et de reconfiguration de l'éjecteur

```
X38 := (TR_X43_X38) or X38 and not(TR_X38_X39);
X39 := (TR_X38_X39) or X39 and not(TR_X39_X40);
X40 := (TR_X39_X40) or X40 and not(TR_X40_X41);
X41 := (TR_X40_X41) or X41 and not(TR_X41_X42);
X42 := (TR_X41_X42) or X42 and not(TR_X42_X43);
X43 := (TR_X42_X43) or X43 and not(TR_X43_X38);


X100 := ((TR_X101_X100) or (TR_X102_X100) or (TR_X103_X100) or (TR_X104_X100))
        or X100 and not((TR_X100_X101) or (TR_X100_X102) or (TR_X100_X103) or (TR_X100_X104));
X101 := (TR_X100_X101) or X101 and not(TR_X101_X100);
X102 := (TR_X100_X102) or X102 and not(TR_X102_X100);
X103 := (TR_X100_X103) or X103 and not(TR_X103_X100);
X104 := (TR_X100_X104) or X104 and not(TR_X104_X100);

   //(*Forçage des Grafcets*)
if (X101){X4:=0; X5:=0; X6:=0; X7:=0; X8:=0; X9:=0; X40:=1;}
if (X102){X38:=0; X39:=0; X40:=0; X41:=0; X42:=0; X43:=0; X6:=1;}
if (X103){X4:=0; X5:=0; X6:=0; X7:=0; X8:=0; X9:=0; X42:=1;}
if (X104){X38:=0; X39:=0; X40:=0; X41:=0; X42:=0; X43:=0; X8:=1;}
```

Figure 97 : Extrait des équations d'activation des étapes du Grafcet du mode dégradé et de reconfiguration

- Enfin, l'affectation des actions est déterminée comme l'illustre la figure 98.

```
void action()
{
VBB    := X1;
EJ     := X5 or X6 or X7 or X39 or X40 or X41;
VTAS   := X17;
VTEX   := X19;
VRM    := X14 or X15;
VRC    := X11 or X12;
CONV   := X21;
BMC    := X34 or X35;
BME    := X31 or X32;
PINCES := X37;
DVL    := X25 or X26 or X27;
}
```

Figure 98 : Equations d'affectation des actions des grafcets

Avant toute implémentation des grafcets dans l'API, un ensemble de propriétés exprimées sous forme de spécifications et conditions doit être vérifié formellement.

La *première* spécification à vérifier est la propriété de non-blocage des grafcets distribués de la commande ainsi que le grafcet de la reconfiguration de la station de bouchonnage. Ceci assuré par la vérification de la condition « Existe-t-il un chemin amenant à une situation de blocage ? ». Cette condition s'exprime dans le vérificateur d'UPPAAL par la proposition suivante :

- Condition 1 : **E < > deadlock**.

Si le résultat de cette vérification est satisfait, cela signifie qu'aucun blocage n'est trouvé et que les grafcets de commande ne sont pas bloquants. La vérification d'autres propriétés est alors possible.

La *deuxième* spécification consiste à vérifier la propriété d'atteignabilité du grafcet demandé après reconfiguration. Ce qui est assurée par la vérification de la condition « Existe-t-il un chemin où l'étape associée à la demande de reconfiguration est active, et l'étape du grafcet demandé après reconfiguration est désactivée ? ». Cette condition est vérifiée pour toutes les étapes où une reconfiguration est déclenchée. Ce qui est exprimé en UPPAAL par les quatre conditions suivantes :

- Condition 2 : **[E < > X101 and not X40]**
- Condition 3 : **[E < > X102 and not X6]**
- Condition 4 : **[E < > X103 and not X42]**
- Condition 5 : **[E < > X104 and not X8].**

Après une demande de reconfiguration, la vérification qu'un seul grafcet est fonctionnel pour l'éjecteur est nécessaire. Pour cela, il faut vérifier l'existence d'un cas où deux étapes sont actives simultanément dans deux grafcets différents (*troisième* spécification). Cette condition est exprimée par :

- Condition 6 : **[E < > (X4 ou X5 ou X6 ou X7 ou X8 ou X9) et (X38 ou X39 ou X40 ou X41 ou X42 ou X43)].**

Le résultat des tests de vérification formelle de l'ensemble des Grafcets de commande est présenté dans le tableau 13.

Tableau 13 : Résultats de la vérification formelle de l'ensemble des grafcets de la station de bouchonnage

Condition vérifiée	Résultat de la vérification
Condition 1	Satisfaisant
Condition 2	Satisfaisant
Condition 3	Satisfaisant
Condition 4	Satisfaisant
Condition 5	Satisfaisant
Condition 6	Satisfaisant

La vérification satisfaisante de l'ensemble de ces conditions permet le passage à un deuxième test de simulation sur le jumeau numérique avant l'implémentation finale sur le système réel.

9 Vers une implémentation de la commande sur la station de bouchonnage

Le jumeau numérique de la station de bouchonnage (figure 99) fonctionne via la collecte d'une combinaison de données. Cet outil nous permet de simuler les équipements de la station et la chaîne de production plus en détail en incluant sa cinématique. Il devient alors possible de valider les conditions de travail des opérateurs ou encore, de faire fonctionner les automates en

condition réelle et de détecter par exemple des interférences. L'objectif derrière la vérification de la commande établie via le jumeau numérique est la réalisation des tests et des scénarios de fonctionnement pour un objectif de modifications sur les programmes automates si un retour de scénario est faux tandis que l'équipement réel est en arrêt, c'est ce qu'on appelle «virtual comissioning».

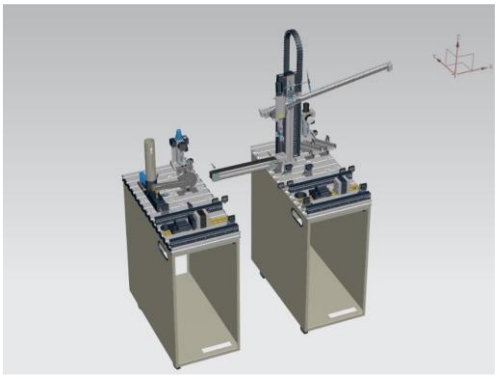


Figure 99 : Jumeau numérique de la station bouchonnage

Après une vérification formelle des propriétés précédentes, il convient soit de traduire les modèles de spécifications grafcet en langage de programmation API ou de récupérer le programme ST établi lors de la vérification formelle sur Uppal. Le programme est alors implémenté dans le simulateur d'API de Siemens PLCSIM-ADVANCED afin de commander le jumeau numérique sous NXMCD (figure 100).

En testant que tous les scénarios simulés correspondent aux fonctionnements souhaités (normal et reconfiguré), la commande est ensuite implémentée dans l'API réel (S7 1500) pour le pilotage de la station de bouchonnage.

Une exécution non-satisfaisante d'un scénario implique un retour vers la phase de modélisation du système et des spécifications. Pour la station réelle, un scénario non-satisfaisant peut être dû également à un problème fonctionnel du jumeau numérique. Cette fois-ci, le jumeau numérique doit être corrigé.

Parmi les scénarios testés nous identifions les deux scénarios suivants qui nous ont permis de modifier les spécifications pour déterminer les tableaux 14 et 15 :

- Scénario 1 : Quand le magasin est vidé, la rentrée du vérin des bouchons blancs est attendue pour alimenter le magasin par un bouchon.
- Scénario 2 : Le bras de manipulation est positionné côté embouteillage avec un bouchon serré par la pince et le vérin de levage est commandé pour descendre et remonter. Il faut que la pince s'ouvre une fois que la position basse du vérin de levage est atteinte.

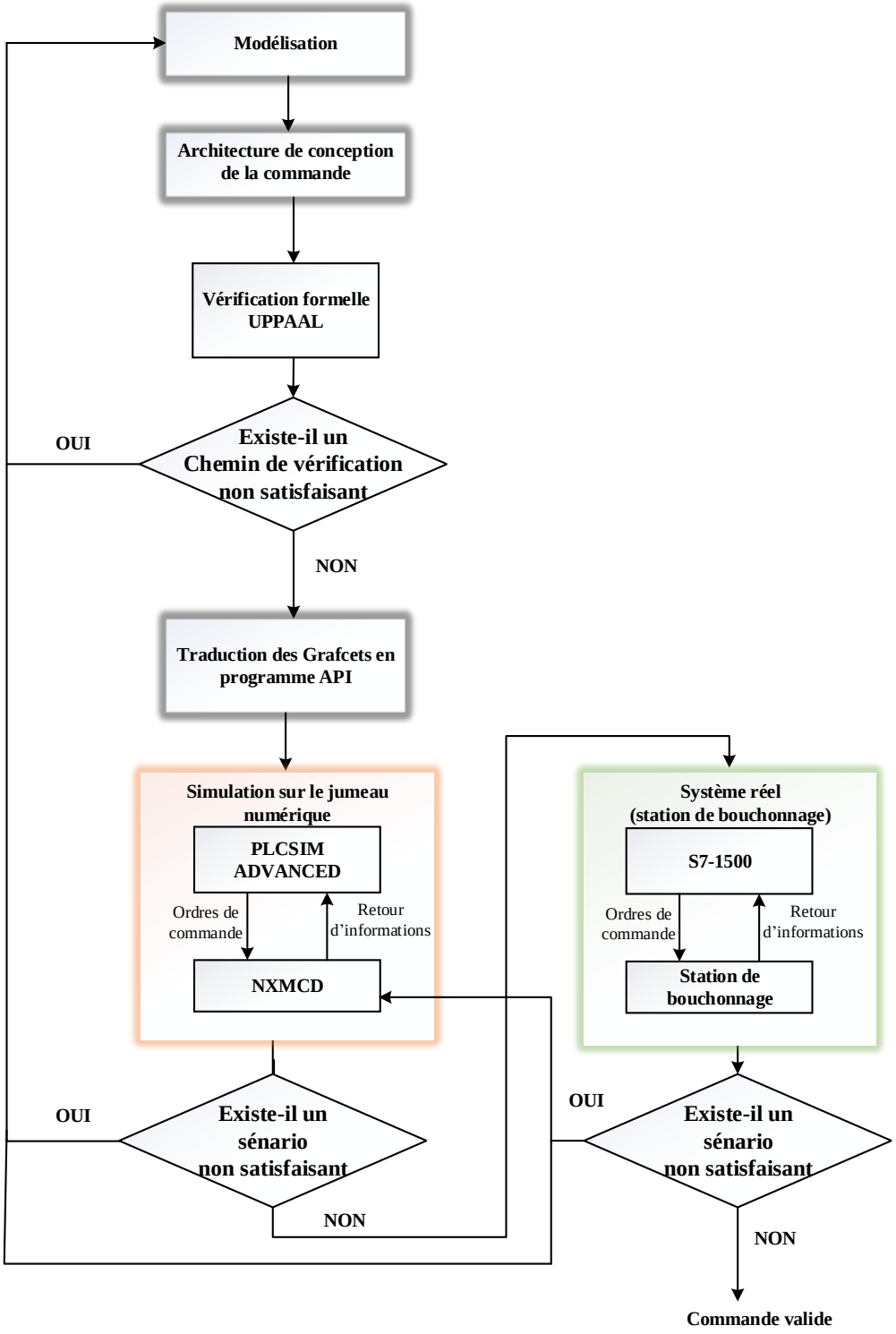


Figure 100 : Démarche d'implémentation de la commande sur le jumeau numérique et le système réel

Tableau 14 : Modifications apportées aux spécifications

Scénario	**Modification**	
	Ancienne spécification	**Nouvelle spécifications**
Scénario 1	- La désactivation du vérin se fait juste après l'activation du capteur c_vbb (aucune spécification n'était exprimée sur la désactivation de l'ordre VBB). **Spécification :** (Inh : VBB) If : (-)	- Le temps mis par le vérin pour faire son aller-retour ne permet pas la descente totale du bouchon, pour cela nous avons ajouté une spécification sur la désactivation de VBB. Une temporisation est nécessaire pour la descente du bouchon **Spécification :** (Inh : VBB) If (T#1s)
Scénario 2	- La désactivation du vérin de levage se fait juste après l'activation du capteur ↑vlb (aucune spécification n'était exprimée sur la désactivation de l'ordre VDL). **Spécification :** (Inh : VDL) If (-)	-La montée du vérin après l'atteinte de sa position basse ne permet pas d'avoir le temps nécessaire à la pince pour desserrer le bouchon. Pour cela nous avons ajouté une temporisation comme spécification sur l'inhibition de VDL. **Spécification :** (Inh : VDL) If (T#1s)

Un premier test sur le jumeau numérique (tableau 15) nous a permis d'identifier une erreur lors de la définition des spécifications globales du vérin de levage (scénario 2). Après la correction de cette spécification, un deuxième test sur le jumeau numérique a été lancé, le résultat cette fois-ci est correct pour les deux scénarios.

Le premier essai sur la station s'est révélé être un échec. Nous avons identifié une erreur au niveau du scénario 1 qui était correct lors de la simulation sur le jumeau numérique. L'erreur qui s'est produite au niveau du vérin des bouchons est liée à l'écart de temps de réaction entre le système réel et le jumeau numérique. Après la correction de la spécification en question et un second essai sur la station, le résultat est correct.

Tableau 15 : Résultats des scénarios avant et après correction des spécifications

Scénario	**Résultat de la simulation**	**Résultat de la simulation après correction n°1**	**Résultat de l'application sur la station**	**Résultat de l'application sur la station après correction n°2**
Scénario 1	Correct	Correct	Faux	Correct
Scénario 2	Faux	Correct	Correct	Correct

10 Conclusion

Dans ce chapitre, nous avons appliqué l'approche de la conception de la commande reconfigurable sur un système automatisé de production existant dans notre laboratoire de recherche. Tout d'abord, le système étudié a été introduit et décrit dans la section 1. Nous nous sommes focalisés ensuite sur une seule station assurant le bouchonnage que nous avons détaillé dans la section 2. Les différents EPO sont modélisés dans la section 3 avec une prise en compte d'un défaut sur un capteur de fin de course appartenant à un EPO (éjecteur) de la station. Ensuite, les contrôleurs locaux des comportements normaux et dégradés de ces EPO sont déterminés (section 4) en passant par une synthèse de la commande basée sur la SCT. En intégrant l'ensemble des spécifications globales, nous établissons les modèles des contrôleurs distribués (section 5) que nous traduisons par la suite en grafcet (section 6) pour un objectif d'implémentation. Par la suite, les spécifications de reconfiguration sont définies et interprétées aussi en grafcet (section 7). Avant l'implémentation des grafcets dans l'API, ils sont vérifiés sur UPPAAL pour s'assurer que la commande établie n'est pas bloquante et la demande de la reconfiguration est toujours atteinte (section 8). Enfin, les grafcets de commande sont implémentés et appliqués sur le jumeau numérique et ensuite sur le système réel (section 9).

Conclusion et perspectives

Les systèmes automatisés de production (SAP) représentent une classe importante de systèmes industriels qui sont devenus complexes et sensibles aux dysfonctionnements avec des conséquences importantes sur la productivité, la qualité de production et la sécurité des biens et des personnes. Un défi majeur est de mettre en œuvre des approches formelles de conception de la commande afin de garantir la sûreté de fonctionnement. En conséquence, les travaux présentés dans ce mémoire portent sur la conception d'une commande reconfigurable et tolérante aux fautes pour les systèmes automatisés de production commandés par des automates programmables industriels.

La majorité des approches proposées dans la littérature pour la reconfiguration ont été développées pour des systèmes où les informations utilisées sont centralisées (Cho et Lim 1999). Lorsque le système automatisé de production à étudier est de grande dimension, la conception des contrôleurs reconfigurables, en se basant sur des méthodes de synthèse centralisée issues de la théorie de contrôle par supervision, s'avère une tâche complexe (Kim et al. 2019). La solution pour assurer la reconfiguration des SAP est très souvent basée sur l'exploitation des redondances matérielles mais cette solution peut être coûteuse pour les entreprises dans le besoin de reconfigurer plusieurs éléments défaillants (Deschamps 2007). Dans le contexte de la SCT, la prise en compte des événements fautifs ainsi que des événements associés à la reconfiguration pour les approches centralisées conduit non seulement à une explosion de l'espace d'états mais aussi à des problèmes d'interprétation des modèles (Zaytoon et Riera 2017). Par ailleurs, la littérature montre qu'il existe peu de méthodologies pour l'implémentation de contrôleurs reconfigurables et plus généralement de contrôleurs dans les automates programmables industriels (Fabian et Hellgren 1998).

Pour répondre à ces problématiques, la démarche de conception des contrôleurs flexibles, reconfigurables et implémentables développée dans ce mémoire propose :

- Un mécanisme de reconfiguration qui se base sur la notion de *flexibilité* pour évaluer différents objectifs de contrôle afin d'atteindre l'efficacité et la réactivité. Pour assurer ce contrôle flexible, notre méthode se base sur l'exploitation des architectures de commande distribués des SAP et sur la détermination d'un ensemble de contraintes de reconfiguration qui permettent la commutation entre les différents contrôleurs en toute flexibilité et sans aucun arrêt du système. En se basant sur une architecture de contrôle distribuée, l'explosion combinatoire de l'espace d'états récurrente dans les approches centralisées de contrôle est évitée tout en facilitant les étapes de modélisation de la partie opérative ainsi que les spécifications à respecter (Tahiri et al. 2019 a).
- Une comparaison entre une démarche de reconfiguration distribuée et centralisée appliquées à un système didactique de transfert pour prouver l'efficacité de l'approche distribuée (Tahiri et al. 2019 b).
- Une redondance non pas au niveau des éléments matériels mais appliquée au niveau des contrôleurs. Pour chaque contrôleur distribué du comportement normal, nous définissons son équivalent dont nous avons intégré la notion d'événements temporisés

pour remplacer le fonctionnement estimé des capteurs défaillants. La synthèse des contrôleurs de fonctionnement dégradé a été valorisée par deux articles dans deux congrès (Tahiri et al. 2016) et (Tahiri et al. 2018).

- Une implémentation de la commande reconfigurable, dans un Automate Programmable Industriel est assurée tout d'abord par une traduction de l'ensemble des contrôleurs distribués résultant et les spécifications de la reconfiguration en grafcet. Le non-blocage des grafcets de commande et des conditions d'atteignabilité des grafcets reconfigurés sont vérifiés par un outil de vérification et de validation (UPPAAL) (Tahiri et al. 2020).
- En finalité, la méthode proposée pour la reconfiguration est sans intervention manuelle et sans équipements supplémentaires.

En résumé, l'approche de la reconfiguration de la commande proposée est constituée de 4 étapes : (1) modélisation de la partie opérative et des spécifications de sécurité, de vivacité et de reconfiguration en intégrant la notion d'événements temporisés, (2) synthèse des contrôleurs locaux et distribués, (3) traduction des contrôleurs et des spécifications de la reconfiguration en grafcet et finalement (4) la vérification de l'ensemble des grafcets de commande ainsi que les grafcets de reconfiguration avant l'implémentation dans un API.

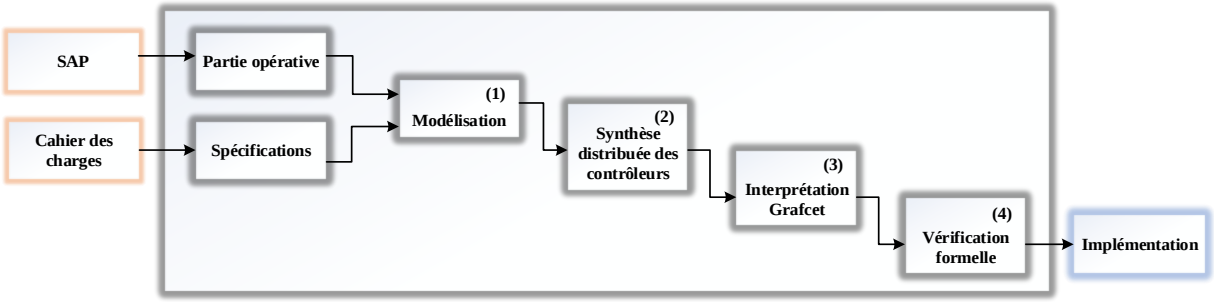


Figure 101 : Etapes de la méthode de reconfiguration proposée

Si la démarche présente des avantages certains, elle comporte des inconvénients liés à :

- La phase de modélisation de la partie opérative (complexité à établir les modèles correspondant à chaque élément de partie opérative).
- La phase de modélisation des contraintes de spécifications (comment s'assurer qu'elles soient suffisantes, qu'elles soient non bloquantes ?).
- L'intégration des spécifications de reconfiguration. Dans le cadre de notre travail, elles sont interprétées par un grafcet séparé gérant les deux modes de comportement (normal et dégradé). Cependant, celles-ci peuvent être prises en compte dès la phase de modélisation pour déterminer un unique grafcet de commande où la reconfiguration peut être exprimée par un simple « OU » entre les deux modes de comportement.
- L'optimisation des modèles générés par la démarche adoptée pour la conception de la commande reconfigurable.
- L'absence d'outil logiciel pour l'automatisation de la démarche.

Les avantages et les inconvénients des travaux proposés permettent d'identifier plusieurs perspectives qui constituent des pistes intéressantes et prometteuses pour des travaux de recherche à venir. Ces pistes concernent :

La modélisation :

L'étape de la conception des modèles de comportements normaux et de comportements dégradés peut être fastidieuse. Le développement d'une bibliothèque de modèles éprouvés, liée aux différents éléments de la partie opérative et de contrôleurs locaux pour différents équipements des SAP, peut-être une réponse à la difficulté de modélisation. Il resterait ensuite à l'opérateur la tâche de définir les contraintes globales pour déterminer les contrôleurs distribués. Par conséquent, la phase de la modélisation deviendrait moins complexe. En choisissant l'équipement, la bibliothèque pourrait offrir au concepteur de la commande les deux modèles des contrôleurs correspondant à ses deux fonctionnements normal et dégradé.

La difficulté se manifeste également au niveau de la modélisation des spécifications locales et globales. Les vérifications assurées d'abord par UPPAAL avant l'implémentation, et ensuite par le jumeau numérique après implémentation permettent d'effectuer des tests facilement et de corriger les spécifications si c'est nécessaire. Il faut améliorer l'aide à la correction.

L'optimisation :

Les grafcets de commande résultants sont automatiquement obtenus à partir de l'étape d'interprétation des contrôleurs distribués. Cependant, l'optimisation de cette interprétation semble possible dans le contexte où certaines étapes générées peuvent parfois être supprimées. En effet, la commande obtenue contient des passages des capteurs d'une lecture de « 0 » à « 1 » qui n'influence pas le fonctionnement du système et qui peuvent être supprimés pour réduire les grafcets de commande résultants.

L'allocation mémoire de l'API :

L'implantation des grafcets de commande dans ses deux modes, normal et dégradé, et l'implantation de la reconfiguration nécessitent une mémoire très importante au niveau de l'API. Ceci n'est plus un problème puisque les API ont évolué et ont des caractéristiques beaucoup plus performantes qu'il y a quelques années. Cependant, l'implémentation de l'ensemble des grafcets montrent que deux tiers des étapes sont redondants. Par conséquent, le programme implémenté dans l'API est en finalité doublé en ligne de code. La perspective à ce niveau est d'étudier l'impact de l'ensemble des lignes supplémentaires sur le temps de cycle API et ce qui peut affecter le temps de production.

L'automatisation de la démarche :

La démarche proposée dans ce mémoire repose sur le concepteur qui intervient au niveau de la modélisation des éléments de partie opérative et des spécifications locales et globales. Les vérifications assurées d'abord par UPPAAL avant l'implémentation, et ensuite par le jumeau numérique après implémentation permettent d'effectuer des tests et de corriger les spécifications si est nécessaire. Une piste envisageable est celle de la proposition d'un outil logiciel aidant le concepteur à élaborer l'ensemble des spécifications. En effet, au même titre que les approches d'ingénierie de systèmes sûrs de fonctionnement basées sur les modèles (Bévan et al. 2012), des modèles conceptuels de haut niveau doivent pouvoir permettre l'interprétation automatique de spécifications. L'idée est d'essayer de bien modéliser un « tout » dans son environnement et pas seulement dans son objectif fonctionnel.

Le types des défauts traités :

Les défauts étudiés dans ce mémoire sont associés aux capteurs bloqués à la lecture d'un « 1 » ou d'un « 0 », mais un défaut d'actionneur peut également détecté. Dans ce cas, il est important de disposer d'une redondance matérielle pour garantir l'action souhaitée. Le principe de la reconfiguration est relativement similaire à la méthode proposée dans le chapitre 3. En effet, la conception des contrôleurs de comportement normal reste identique et le grafcet de reconfiguration gèrera la commutation entre le grafcet associé à l'actionneur défectueux et le grafcet de l'actionneur redondant. Au-delà des problématiques de défauts, ceci montre que cette méthode de conception peut être utilisée pour mettre en œuvre des aspects de flexibilité comme étudiés dans le chapitre 2.

Les axes de reconfiguration :

Toujours dans l'idée de rendre le système plus flexible dans le cadre d'une approche de reconfiguration, il serait pertinent de reconfigurer non plus suite à une défaillance de la partie opérative mais suite à une demande de changement des spécifications du cahier des charges. En effet, l'industrie du futur doit permettre la personnalisation de la production sans pour autant passer par des phases de (re)conception de commande en fonction de la partie opérative disponible, tout en garantissant la sûreté de fonctionnement. Dans ce cadre, un des problèmes de la reconfiguration est d'être capable de choisir le bon mode de fonctionnement futur du système tout en considérant le mode de fonctionnement actuel et l'état du système. En spécifiant des règles de reconfiguration sous forme de contraintes logiques, un filtre logique de reconfiguration pourrait fournir le(s) mode(s) de fonctionnement adapté(s) à l'état actuel du système en respectant les différentes règles. En se plaçant dans le contexte de la reconfiguration, les avantages de l'approche par filtre par rapport aux approches classiques à base d'automates à états ou de RdP seraient les suivantes (Pichard 2018) :

- Modélisation des règles d'exclusivité entre les différents modes de fonctionnement : cette modélisation est complexe à traduire avec les modèles à états finis. Dans l'approche par filtre, la notion de contraintes combinées a été définie et devraient permettent aisément de modéliser les règles d'exclusivité.
- Connaissance de l'état initial du système : dans un modèle à états, il est indispensable de connaitre l'état initial. Le bon fonctionnement du filtre n'est que peu influencé par l'état initial du système. En effet, la seule restriction est que l'état initial respecte les contraintes. Dès lors, même suite à une réinitialisation du contrôleur ou à un changement de configuration, le filtre reste opérationnel.
- Ajout ou suppression de règles de reconfiguration : la modélisation dans les modèles à états doit être revue ce qui peut s'avérer couteux en termes de temps de développement et donc peu réactif. Pour l'approche par filtre, il suffit d'ajouter ou de retirer une contrainte à l'ensemble et de générer le filtre automatiquement.

Bibliographie

Abbas, M., et H. ElMaraghy. 2018. « Synthesis and optimization of manufacturing systems configuration using co-platforming ». *CIRP Journal of Manufacturing Science and Technology* 20 (janvier): 51-65. https://doi.org/10.1016/j.cirpj.2017.09.006.

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.

Alexandre, Philippot, Tajer Abdelouahed, François Gellot, et Véronique Carré-Ménétrier. 2005. « Méthodologie de modélisation dans le cadre de la synthèse formelle des SED vol 2, n°2. » *e-STA Sciences et Technologies pour lÁutomatique*. https://hal.archives-ouvertes.fr/hal-02339310.

Altisen, Karine, Pierre-Alain Markey, Nicolas Reynier, et Stavros Tripakis. 2005. « Implémentabilité des automates temporisés ». *Journal Européen des Systèmes Automatisés* 39 (1-3): 395-406. https://doi.org/10.3166/jesa.39.395-406.

Alur, Rajeev, et David L. Dill. 1994. « A Theory of Timed Automata ». *Theoretical Computer Science* 126 (2): 183-235. https://doi.org/10.1016/0304-3975(94)90010-8.

Angelopoulou, Anastasia, Konstantinos Mykoniatis, et Nithisha Reddy Boyapati. 2020. « Industry 4.0: The Use of Simulation for Human Reliability Assessment ». *Procedia Manufacturing*, International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019), 42 (janvier): 296-301. https://doi.org/10.1016/j.promfg.2020.02.094.

Attia, Rachid, Saïd Amari, et Claude Martinez. 2010. « Control of Time-Constrained Dual-Armed Cluster Tools Using (max, +) Algebra ». In *Conference on Control and Fault-Tolerant Systems (SysTol'10)*, 412-17. Nice, France. https://hal.archives-ouvertes.fr/hal-00564167.

Avižienis, Algirdas, Jean-Claude Laprie, et Brian Randell. 2004. « Dependability and Its Threats: A Taxonomy ». In *Building the Information Society*, édité par Renè Jacquart, 91-120. IFIP International Federation for Information Processing. Springer US.

Badihi, Hamed, et Youmin Zhang. 2017. « Passive Fault-tolerant Cooperative Control in an Offshore Wind Farm ». *Energy Procedia*, 8th International Conference on Applied Energy, ICAE2016, 8-11 October 2016, Beijing, China, 105 (mai): 2959-64. https://doi.org/10.1016/j.egypro.2017.03.697.

Baier, Christel, et Joost-Pieter Katoen. 2008. *Principles of Model Checking*. Cambridge, Mass: The MIT Press.

Baker, Albert D. 1998. « A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull ». *Journal of Manufacturing Systems* 17 (4): 297-320. https://doi.org/10.1016/S0278-6125(98)80077-0.

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.

Batchkova, Idilia, Georgi Popov, Hristo Karamishev, et Grigor Stambolov. 2013. « Dynamic reconfigurability of control systems using IEC 61499 standard ». *IFAC Proceedings Volumes*, 15th IFAC Workshop on International Stability, Technology, and Culture, 46 (8): 256-61. https://doi.org/10.3182/20130606-3-XK-4037.00050.

Bauer, Nanette, Sebastian Engell, Ralf Huuck, Sven Lohmann, Ben Lukoschus, Manuel Remelhe, et Olaf Stursberg. 2004. « Verification of PLC Programs Given as Sequential

Function Charts ». In *Integration of Software Specification Techniques for Applications in Engineering: Priority Program SoftSpez of the German Research Foundation (DFG), Final Report*, édité par Hartmut Ehrig, Werner Damm, Jörg Desel, Martin Große-Rhode, Wolfgang Reif, Eckehard Schnieder, et Engelbert Westkämper, 517-40. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-27863-4_28.

Bayart, mireille. 1994. « Instrumentation intelligente – Systèmes automatisés de production à Intelligence distribuée ». Habilitation à diriger les recherches. Université des sciences et technologies de Lille.

Beier, Grischa, André Ullrich, Silke Niehoff, Malte Reißig, et Matthias Habich. 2020. « Industry 4.0: How It Is Defined from a Sociotechnical Perspective and How Much Sustainability It Includes – A Literature Review ». *Journal of Cleaner Production* 259 (juin): 120856. https://doi.org/10.1016/j.jclepro.2020.120856.

Berard, B., M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, et P. Schnoebelen. 2001. *Systems and Software Verification: Model-Checking Techniques and Tools*. Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-662-04558-9.

Berruet, Pascal, Jean-François Petin, Fabien Rigaud, Armand Toguyeni, et Éric Zamaï. 2007. « Architectures de pilotage de procédés industriels ». Text. Ref : TIP083WEB - « Conception et Production ». 10 juillet 2007. https://www.techniques-ingenieur.fr/base-documentaire/42521210-methodes-de-production/download/ag3510/architectures-de-pilotage-de-procedes-industriels.html.

Besseron, Xavier. 2010. « Fault tolerance and dynamic reconfiguration for large scale distributed applications ». Theses, Institut National Polytechnique de Grenoble - INPG. https://tel.archives-ouvertes.fr/tel-00486939.

Bévan, Romain, Pascal Berruet, Florent de Lamotte, Mickaël Adam, Olivier Cardin, et Pierre Castagna. 2012. « Generation of multiplatform control for transitic systems using a component-based approach ». In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, 1-8. https://doi.org/10.1109/ETFA.2012.6489605.

Bironneau, Laurent, Dominique Philippe Martin, et Gilles Parisse. 2010. « Fiabiliser les données d'un système d'information de gestion par la méthode AMDEC : principes et études de cas. » *Revue française de gestion industrielle* 29 (1): 87-108.

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.

Bortolini, Marco, Francesco Gabriele Galizia, et Cristina Mora. 2018. « Reconfigurable manufacturing systems: Literature review and research trend ». *Journal of Manufacturing Systems* 49 (octobre): 93-106. https://doi.org/10.1016/j.jmsy.2018.09.005.

Brandin, B. A., et W. M. Wonham. 1994. « Supervisory control of timed discrete-event systems ». *IEEE Transactions on Automatic Control* 39 (2): 329-42. https://doi.org/10.1109/9.272327.

Bruccoleri, Manfredi, Michele Amico, et Giovanni Perrone. 2003. « Distributed intelligent control of exceptions in reconfigurable manufacturing systems ». *International Journal of Production Research* 41 (7): 1393-1412. https://doi.org/10.1080/1352816031000075170.

Bruccoleri, Manfredi, Zbigniew J. Pasek, et Yoram Koren. 2006. « Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling ».

International Journal of Production Economics 100 (1): 87-100. https://doi.org/10.1016/j.ijpe.2004.10.009.

Cai, K., et W. M. Wonham. 2010. « Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems ». *IEEE Transactions on Automatic Control* 55 (3): 605-18. https://doi.org/10.1109/TAC.2009.2039237.

Cassandras, Christos G, et Stéphane Lafortune. 2008. *Introduction to Discrete Event Systems*. Second Edition. New York. //www.springer.com/us/book/9780387333328.

Cauffriez, Laurent. 2005. « Méthodes et modèles pour l'évaluation de la sûreté de fonctionnement de systèmes automatisés complexes : Application à l'exploitation de lignes de production - Application à la conception de systèmes intelligents distribués ». Thesis, Université de Valenciennes et du Hainaut-Cambresis. https://tel.archives-ouvertes.fr/tel-00289414/document.

Cauffriez, Laurent, Vincent Benard, et Dominique Renaux. 2006. « A new formalism for designing and specifying RAMS parameters for complex distributed control systems: the Safe-SADT formalism ». *IEEE Transactions on Reliability* 55 (3): 397-410.

Chen, Jie, et Ron J. Patton. 1998. « Robust Model-Based Fault Diagnosis for Dynamic Systems | Jie Chen | Springer ». 1998. https://www.springer.com/la/book/9780792384113.

Chiron, Fabien. 2008. « Contribution à la flexibilité et à la rapidité de conception des systèmes automatisés avec l'utilisation d'UML ». clermont-ferrand: Universite blaise pascal.

Cho, Kwang-Hyun, et Jong-Tae Lim. 1999. « Mixed Centralized/Decentralized Supervisory Control of Discrete Event Dynamic Systems ». *Automatica* 35 (1): 121-28. https://doi.org/10.1016/S0005-1098(98)00137-X.

Clarhaut, Joffrey. 2009a. « Design of dependable automated system architectures using temporal sequences of failures ». Theses, Université des Sciences et Technologie de Lille - Lille I. https://tel.archives-ouvertes.fr/tel-00460336.

———. 2009b. « Prise en compte des séquences de défaillances pour la conception de systèmes d'automatisation : Application au ferroutage ». *http://www.theses.fr*. Thesis, Lille 1. http://www.theses.fr/2009LIL10009.

Clarke, E. M., E. A. Emerson, et A. P. Sistla. 1986. « Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications ». *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8 (2): 244-63. https://doi.org/10.1145/5397.5399.

Corrêa, Henrique L, et Nigel Slack. 1996. « Framework to analyse flexibility and unplanned change in manufacturing systems ». *Computer Integrated Manufacturing Systems* 9 (1): 57-64. https://doi.org/10.1016/0951-5240(95)00038-0.

Cotta, Carlos, Marc Sevaux, et Kenneth Sörensen, éd. 2008. *Adaptive and Multilevel Metaheuristics*. Studies in Computational Intelligence. Berlin Heidelberg: Springer-Verlag. https://www.springer.com/la/book/9783540794370.

Darabi, H., M. A. Jafari, et A. L. Buczak. 2003. « A control switching theory for supervisory control of discrete event systems ». *IEEE Transactions on Robotics and Automation* 19 (1): 131-37. https://doi.org/10.1109/TRA.2002.807551.

Dash, Sourabh, et Venkat Venkatasubramanian. 2000. « Challenges in the industrial applications of fault diagnostic systems - ScienceDirect ». 2000. https://www.sciencedirect.com/science/article/pii/S0098135400003744.

David, R. 1993. « Petri Nets and Grafcet for Specification of Logic Controllers ». *IFAC Proceedings Volumes*, 12th Triennal Wold Congress of the International Federation of Automatic control. Volume 1 Theory, Sydney, Australia, 18-23 July, 26 (2, Part 1): 683-88. https://doi.org/10.1016/S1474-6670(17)49215-9.

Deif, Ahmed M., et Waguih ElMaraghy. 2007. « Investigating Optimal Capacity Scalability Scheduling in a Reconfigurable Manufacturing System ». *The International Journal of*

Advanced Manufacturing Technology 32 (5): 557-62. https://doi.org/10.1007/s00170-005-0354-9.

Deif, Ahmed M., et Waguih H. ElMaraghy. 2006. « A control approach to explore the dynamics of capacity scalability in reconfigurable manufacturing systems ». *Journal of Manufacturing Systems* 25 (1): 12-24. https://doi.org/10.1016/S0278-6125(07)00003-9.

Demri, Amel. 2009. « Reliability estimation of mechatronic systems using functional and dysfunctional analysis ». Theses, Université d'Angers. https://tel.archives-ouvertes.fr/tel-00467277.

Dennis, Louise, Michael Fisher, Nicholas K. Lincoln, Sandor M. Veres, et Alexei Lisitsa. 2010. « An Agent Based Framework for Adaptive Control and Decision Making of Autonomous Vehicles ». *IFAC Proceedings Volumes*, 10th IFAC Workshop on the Adaptation and Learning in Control and Signal Processing, 43 (10): 310-17. https://doi.org/10.3182/20100826-3-TR-4015.00058.

Deschamps, Eric. 2007. « Diagnostic de Services pour la Reconfiguration Dynamique de Systèmes à Evénements Discrets Complexes ». Phdthesis, Institut National Polytechnique de Grenoble - INPG. https://tel.archives-ouvertes.fr/tel-00196462/document.

Durand, Bastien. 2011. « Proposition d'une architecture de contrôle adaptative pour la tolérance aux fautes ». Phdthesis, Université Montpellier II - Sciences et Techniques du Languedoc. https://tel.archives-ouvertes.fr/tel-00684149.

Durka, Jean-Luc, et M. Brun. 1993. « Méthodologie d'évaluation de la sécurité des systèmes automatisés industriels ». In *Automation'93*, 49-59. Paris, France. https://hal-ineris.archives-ouvertes.fr/ineris-00971860.

ElMaraghy, Waguih, Hoda ElMaraghy, Tetsuo Tomiyama, et Laszlo Monostori. 2012. « Complexity in engineering design and manufacturing ». *CIRP Annals* 61 (2): 793-814. https://doi.org/10.1016/j.cirp.2012.05.001.

Elmasry, Shady S., Ayman M. A. Youssef, et Mohamed A. Shalaby. 2015. « A cost-based model to select best capacity scaling policy for reconfigurable manufacturing systems ». *IJMR* 10: 162-83. https://doi.org/10.1504/IJMR.2015.069715.

Fabian, M., et A. Hellgren. 1998. « PLC-based implementation of supervisory control for discrete event systems ». In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, 3:3305-10 vol.3. https://doi.org/10.1109/CDC.1998.758209.

Faraut, G., L. Piétrac, et E. Niel. 2010. « Control law synthesis and reconfiguration using SCT ». In *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*, 576-81. https://doi.org/10.1109/SYSTOL.2010.5675977.

Farid, Amro M. 2017. « Measures of Reconfigurability and Its Key Characteristics in Intelligent Manufacturing Systems ». *Journal of Intelligent Manufacturing* 28 (2): 353-69. https://doi.org/10.1007/s10845-014-0983-7.

Fayollas, Camille. 2015. « Architecture logicielle générique et approche à base de modèles pour la sûreté de fonctionnement des systèmes interactifs critiques ».

Fougères, Alain-Jérôme. 2005. « Projet PLACID - Intégration de □-outils d'analyse fonctionnelle ». https://hal.archives-ouvertes.fr/hal-00576510.

Fraikin, Benoît, Marc Frappier, et Richard St-Denis. 2014. « Supervisory Control Theory with Alloy ». *Science of Computer Programming*, Abstract State Machines, Alloy, B, VDM, and Z, 94 (novembre): 217-37. https://doi.org/10.1016/j.scico.2014.04.016.

Gao, Zhifeng, Bing Han, Guoping Jiang, Jinxing Lin, et Dezhi Xu. 2017. « Active fault tolerant control design approach for the flexible spacecraft with sensor faults ». *Journal of the Franklin Institute* 354 (18): 8038-56. https://doi.org/10.1016/j.jfranklin.2017.09.023.

Gartner. 2019. « Gartner Survey Reveals Digital Twins Are Entering Mainstream Use ». Gartner. 2019. https://www.gartner.com/en/newsroom/press-releases/2019-02-20-gartner-survey-reveals-digital-twins-are-entering-mai.

Gerwin, Donald. 1993. « Manufacturing Flexibility: A Strategic Perspective ». *Management Science* 39 (4): 395-410.

Gharsallaoui, Hajer. 2010. « Control reconfiguration and active accommodation for operating modes of flat systems ». Theses, Ecole Centrale de Lille. https://tel.archives-ouvertes.fr/tel-01088540.

Gonzva, Michaël, Bruno Barroca, Serge Lhomme, Pierre-Etienne Gautier, et Youssef YD DIAB. 2016. « Apport de la sûreté de fonctionnement à l'analyse spatialisée du risque inondation ». *Revue Internationale de Géomatique* 26 (3): 329-61. https://doi.org/10.3166/rig.2016.00003.

Grieves, Michael W., et John Vickers. 2017. « Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems ». In . https://doi.org/10.1007/978-3-319-38756-7_4.

GRUSENMEYER, C. 2000. « Interactions maintenance-exploitation et sécurité. Etude bibliographique. 1. Les tâches de maintenance : définitions et caractéristiques contribuant à leur criticité. » Research Report Notes scientifiques et techniques de l'INRS NS 188. Institut National de Recherche et de Sécurité (INRS). https://hal-lara.archives-ouvertes.fr/hal-01420179.

Guenab, Fateh, Jean-Louis Boulanger, et Walter Schön. 2008. « Sécurité des systèmes de contrôle-commande et signalisation ferroviaire: nouvelle approche d'analyse préliminaire des risques ». In *5ème Conférence Internationale Francophone d'Automatique*, Papier N° 111. Bucarest, Romania. https://hal.archives-ouvertes.fr/hal-00339947.

Gumasta, Kapil, Santosh Kumar Gupta, Lyes Benyoucef, et M. K. Tiwari. 2011. « Developing a reconfigurability index using multi-attribute utility theory ». *International Journal of Production Research* 49 (6): 1669-83. https://doi.org/10.1080/00207540903555536.

Gupta, Yash P., et Toni M. Somers. 1996. « Business Strategy, Manufacturing Flexibility, and Organizational Performance Relationships: A Path Analysis Approach ». *Production and Operations Management* 5 (3): 204-33. https://doi.org/10.1111/j.1937-5956.1996.tb00395.x.

Hadj-Mabrouk, Habib. 2007. « Contribution du raisonnement à partir de cas à l'analyse des effets des erreurs du logiciel. Application à la sécurité des transports ferroviaires ». In *Raisonnement à partir de cas*, édité par J. Renaud, B.-C. Chebel Morello, B. Fuchs, et J. Lieber, 2:123-48. Surveillance, diagnostic et maintenance. Traité IC2, Informatique et systèmes d'information 4. Hermès/Lavoisier. https://hal.archives-ouvertes.fr/hal-02422931.

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.

Hnětynka, Petr, et František Plášil. 2006. « Dynamic Reconfiguration and Access to Services in Hierarchical Component Models ». In *Component-Based Software Engineering*, édité par Ian Gorton, George T. Heineman, Ivica Crnković, Heinz W. Schmidt, Judith A. Stafford, Clemens Szyperski, et Kurt Wallnau, 352-59. Lecture Notes in Computer Science. Springer Berlin Heidelberg.

Huang, Aihua, Fazleena Badurdeen, et I. S. Jawahir. 2018. « Towards Developing Sustainable Reconfigurable Manufacturing Systems ». *Procedia Manufacturing*, 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and

Smart Industry for Good of Humanity, 17 (janvier): 1136-43. https://doi.org/10.1016/j.promfg.2018.10.024.

Hyun, J.-H., et B.-H. Ahn. 1993. « A Unifying Framework for Manufacturing Flexibility ». *MANUFACTURING REVIEW* 5 (4): 251.

Iung, Benoît. 2019. « From predictive maintenance to prescriptive one: a required evolution to support industry of the future paradigm ». In *Conference on Complexity Analysis of Industrial Systems and Advanced Modeling, CAISAM 2019*. Ben Guerir, Morocco. https://hal.archives-ouvertes.fr/hal-02126720.

Jennings, Nicholas R. 2000. « On agent-based software engineering ». *Artificial Intelligence* 117 (2): 277-96. https://doi.org/10.1016/S0004-3702(99)00107-1.

Jiang, Jin, et Xiang Yu. 2012. « Fault-tolerant control systems: A comparative study between active and passive approaches ». *Annual Reviews in Control* 36 (1): 60-72. https://doi.org/10.1016/j.arcontrol.2012.03.005.

Jiménez, Jose-Fernando. 2017. « Dynamic and Hybrid Architecture for the Optimal Reconfiguration of Control Systems: Application to Manufacturing Control ». Université de Valenciennes et du Hainaut-Cambresis.

Kapitanov, A. V. 2017. « Manufacturing System Flexibility Control ». *Procedia Engineering*, International Conference on Industrial Engineering, ICIE 2017, 206 (janvier): 1470-75. https://doi.org/10.1016/j.proeng.2017.10.663.

Khelassi, Ahmed. 2011. « New Methodology for Active Fault Tolerant Control Design with Respect to System Reliability ». Theses, Université Henri Poincaré - Nancy 1. https://tel.archives-ouvertes.fr/tel-01746195.

Khoumsi, A., et H. Chakib. 2014. « Decentralized supervisory control of discrete event systems moving decisions closer to actions ». In *2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 02:280-87. https://doi.org/10.5220/0005010702800287.

Kim, D.-Y., J.-W. Park, S. Baek, K.-B. Park, H.-R. Kim, J.-I. Park, H.-S. Kim, et al. 2019. « A Modular Factory Testbed for the Rapid Reconfiguration of Manufacturing Systems ». *Journal of Intelligent Manufacturing*, mars. https://doi.org/10.1007/s10845-019-01471-2.

Komenda, J., S. Lahaye, J. -L. Boimond, et T. van den Boom. 2018. « Max-plus Algebra in the History of Discrete Event Systems ». *Annual Reviews in Control* 45 (janvier): 240-49. https://doi.org/10.1016/j.arcontrol.2018.04.004.

Komenda, Jan, et Jan H. van Schuppen. 2007. « Control of Discrete-Event Systems with Modular or Distributed Structure ». *Theoretical Computer Science* 388 (1): 199-226. https://doi.org/10.1016/j.tcs.2007.07.049.

Konstantopoulos, Ioannis K., et Panos J. Antsaklis. 1999. « An Optimization Approach to Control Reconfiguration ». *Dynamics and Control* 9 (3): 255-70. https://doi.org/10.1023/A:1008394805721.

Korbicz, J, J-M Koscielny, Z Kowalczuk, et W Cholewa. 2004. « Fault Diagnosis - Models, Artificial Intelligence, Applications | Józef Korbicz | Springer ». 2004. https://www.springer.com/us/book/9783540407676.

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.

Kristoffersen, Kåre J., Francois Laroussinie, Kim G. Larsen, Paul Pettersson, et Wang Yi. 1997. « A Compositional Proof of a Real-Time Mutual Exclusion Protocol ». In *TAPSOFT '97: Theory and Practice of Software Development*, édité par Michel Bidoit et Max Dauchet, 565-79. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/BFb0030626.

Kumar, Ratnesh, et Shigemasa Takai. 2012. « A Framework for Control-Reconfiguration Following Fault-Detection in Discrete Event Systems* ». *IFAC Proceedings Volumes*, 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, 45 (20): 848-53. https://doi.org/10.3182/20120829-3-MX-2028.00289.

Lafortune, Stéphane, Feng Lin, et Christoforos N. Hadjicostis. 2018. « On the history of diagnosability and opacity in discrete event systems ». *Annual Reviews in Control* 45 (janvier): 257-66. https://doi.org/10.1016/j.arcontrol.2018.04.002.

Lafou, Meriem, Luc Mathieu, Stéphane Pois, et Marc Alochet. 2016. « Manufacturing System Flexibility: Product Flexibility Assessment ». *Procedia CIRP*, Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems, 41 (janvier): 99-104. https://doi.org/10.1016/j.procir.2015.12.046.

Lan, Jianglin, et Ron J. Patton. 2016. « A new strategy for integration of fault estimation within fault-tolerant control ». *Automatica* 69 (juillet): 48-59. https://doi.org/10.1016/j.automatica.2016.02.014.

Landers, R. G., B. -K. Min, et Y. Koren. 2001. « Reconfigurable Machine Tools ». *CIRP Annals* 50 (1): 269-74. https://doi.org/10.1016/S0007-8506(07)62120-9.

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.

Lennartson, Bengt, Oskar Wigström, Martin Fabian, et Francesco Basile. 2014. « Unified Model for Synthesis and Optimization of Discrete Event and Hybrid Systems ». *IFAC Proceedings Volumes*, 12th IFAC International Workshop on Discrete Event Systems (2014), 47 (2): 86-92. https://doi.org/10.3182/20140514-3-FR-4046.00140.

Li, Ben, João Carlos Basilio, Manel Khlif-Bouassida, et Armand Toguyéni. 2017. « Polynomial Time Verification of Modular Diagnosability of Discrete Event Systems ». *IFAC-PapersOnLine*, 20th IFAC World Congress, 50 (1): 13618-23. https://doi.org/10.1016/j.ifacol.2017.08.2387.

Lin, F., et W. M. Wonham. 1987. « Decentralized Supervisory Control of Discrete-event Systems ». *IFAC Proceedings Volumes*, 10th Triennial IFAC Congress on Automatic Control - 1987 Volume IX, Munich, Germany, 27-31 July, 20 (5, Part 9): 163-68. https://doi.org/10.1016/S1474-6670(17)55027-2.

Lucas Silva, André, Richardson Ribeiro, et Marcelo Teixeira. 2017. « Modeling and control of flexible context-dependent manufacturing systems ». *Information Sciences* 421 (décembre): 1-14. https://doi.org/10.1016/j.ins.2017.08.084.

Lyke, J. C., C. G. Christodoulou, G. A. Vera, et A. H. Edwards. 2015. « An Introduction to Reconfigurable Systems ». *Proceedings of the IEEE* 103 (3): 291-317. https://doi.org/10.1109/JPROC.2015.2397832.

M. Macktoobian, et W. M. Wonham. 2017. « Automatic reconfiguration of untimed discrete-event systems ». In *2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 1-6. https://doi.org/10.1109/ICEEE.2017.8108839.

Machado, José J.B., Bruno Denis, Jean-Jacques Lesage, Jean-Marc Faure, et Jaime Fereira. 2006. « LOGIC CONTROLLERS DEPENDABILITY VERIFICATION USING A PLANT MODEL ». In *3rd IFAC Workshop on Discrete-Event System Design, DESDes'06, Rydzyna (Poland), 26-28 September 2006*, 37-42. Poland. https://hal.archives-ouvertes.fr/hal-00361815.

Matta, A., T. Tolio, F. Karaesmen, et Y. Dallery. 2000. « A new system architecture compared with conventional production system architectures ». *International Journal of Production Research* 38 (17): 4159-69. https://doi.org/10.1080/00207540050204993.

Mehrabi, M. G., A. G. Ulsoy, et Y. Koren. 2000. « Reconfigurable Manufacturing Systems: Key to Future Manufacturing ». *Journal of Intelligent Manufacturing* 11 (4): 403-19. https://doi.org/10.1023/A:1008930403506.

Mehta, B. R., et Y. J. Reddy. 2015. « Chapter 3 - Distributed control system ». In *Industrial Process Automation Systems*, édité par B. R. Mehta et Y. J. Reddy, 75-133. Oxford: Butterworth-Heinemann. https://doi.org/10.1016/B978-0-12-800939-0.00006-1.

Moigne, Jean-Louis Le. 1994. *La théorie du système général: Théorie de la modélisation*. FeniXX.

Moïsio, Gilbert. 2016. « Sûreté de Fonctionnement ». In *Méthodologies Appliquées*. https://methodes.pressbooks.com/chapter/surete-de-fonctionnement/.

Moor, Thomas. 2016. « A discussion of fault-tolerant supervisory control in terms of formal languages ». *Annual Reviews in Control* 41 (janvier): 159-69. https://doi.org/10.1016/j.arcontrol.2016.04.001.

Morel, G., C. E. Pereira, et S. Y. Nof. 2019. « Historical survey and emerging challenges of manufacturing automation modeling and control: A systems architecting perspective ». *Annual Reviews in Control*, février. https://doi.org/10.1016/j.arcontrol.2019.01.002.

Mortureux, Yves. 2016. « Fondamentaux de l'analyse de risque », 40.

Nguyen, Dang-Trinh. 2015. « Diagnostic en ligne des systèmes à événements discrets complexes: approche mixte logique/probabiliste », 133.

Niguez, Julien, Saïd Amari, et Jean-Marc Faure. 2017. « Active Fault-Tolerant Control of Timed Automata with Guards ». *IFAC-PapersOnLine*, 20th IFAC World Congress, 50 (1): 13648-53. https://doi.org/10.1016/j.ifacol.2017.08.2398.

Nooruldeen, A., et K. W. Schmidt. 2015. « State Attraction Under Language Specification for the Reconfiguration of Discrete Event Systems ». *IEEE Transactions on Automatic Control* 60 (6): 1630-34. https://doi.org/10.1109/TAC.2014.2358811.

Ostroff, J. S., et W. M. Wonham. 1985. « A temporal logic approach to real time control ». In *1985 24th IEEE Conference on Decision and Control*, 656-57. https://doi.org/10.1109/CDC.1985.268574.

Panetto, Hervé. 1991. « Contribution to Automation Engineering: Prototyping of Production Systems and Machines ». Theses, Université Henri Poincaré - Nancy I. https://tel.archives-ouvertes.fr/tel-00562037.

Paoli, Andrea, Matteo Sartini, et Stéphane Lafortune. 2008. « A fault tolerant architecture for supervisory control of discrete event systems ». *IFAC Proceedings Volumes*, 17th IFAC World Congress, 41 (2): 6542-47. https://doi.org/10.3182/20080706-5-KR-1001.01103.

———. 2011. « Active fault tolerant control of discrete event systems using online diagnostics ». *Automatica* 47 (4): 639-49. https://doi.org/10.1016/j.automatica.2011.01.007.

Papadopoulos, Yiannis, et John McDermid. 2001. « Automated Safety Monitoring: A Review and Classification of Methods », 32.

Park, Kijung, et Gül E. Okudan Kremer. 2015. « Assessment of static complexity in design and manufacturing of a product family and its impact on manufacturing performance ». *International Journal of Production Economics* 169 (novembre): 215-32. https://doi.org/10.1016/j.ijpe.2015.07.036.

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.

Pichard, Romain. 2018. « Contribution à la Commande des systèmes à événements discrets par filtre logique ». *http://www.theses.fr*. Thesis, Reims. http://www.theses.fr/2018REIMS025.

Pichard, Romain, Alexandre Philippot, Ramla Saddem, et Bernard Riera. 2019. « Safety of Manufacturing Systems Controllers by Logical Constraints With Safety Filter ». *IEEE Transactions on Control Systems Technology* 27 (4): 1659-67. https://doi.org/10.1109/TCST.2018.2827329.

Piriou, Pierre-Yves. 2015. « Contribution to model Based Safety Analysis for dynamic repairable reconfigurable systems ». Theses, Université Paris-Saclay. https://tel.archives-ouvertes.fr/tel-01251556.

Potiron, Katia, Amal El Fallah Seghrouchni, et Patrick Taillibert. 2013. *From Fault Classification to Fault Tolerance for Multi-Agent Systems*.

Powell, David, Jean Arlat, Yves Deswarte, et Karama Kanoun. 2011. « Tolerance of design faults ». In *Dependable and Historic Computing*, édité par J.L. Lloyd C.B. Jones, 428-52. Lecture Notes in Computer Science 6875. Springer. https://hal.archives-ouvertes.fr/hal-00761140.

Procaccia, Henri, Eric Ferton, et Marc Procaccia. 2011. *Fiabilité et maintenance des matériels industriels réparables et non réparables*. Lavoisier.

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.

Quagliaro, Laurence. 1993. « Une nouvelle méthode pour l'analyse quantitative de la sûreté de fonctionnement des systèmes : la méthode des graphes fictifs ». *http://www.theses.fr*. Thesis, Compiègne. http://www.theses.fr/1993COMPD657.

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.

Rayhane, Hassan. 2004. « Surveillance des systèmes de production automatisés : détection et aide au diagnostic ». Theses, Institut National Polytechnique de Grenoble - INPG. https://tel.archives-ouvertes.fr/tel-00169988.

Ribot, Pauline. 2009. « Vers l'intégration diagnostic/pronostic pour la maintenance des systèmes complexes ». Phdthesis, Université Paul Sabatier - Toulouse III. https://tel.archives-ouvertes.fr/tel-00450835/document.

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.

Rohée, Benoit. 2008. « Contribution à la conception d'applications de pilotage des systèmes manufacturiers ». Phdthesis, Université de Reims - Champagne Ardenne. https://tel.archives-ouvertes.fr/tel-00410093/document.

Rösiö, Carin, Tehseen Aslam, Karthik Banavara Srikanth, et Savin Shetty. 2019. « Towards an assessment criterion of reconfigurable manufacturing systems within the automotive industry ». *Procedia Manufacturing*, 7th International conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2018), 28 (janvier): 76-82. https://doi.org/10.1016/j.promfg.2018.12.013.

Roussel, Jean-Marc, Jean-Marc Faure, Jean-Jacques Lesage, et Antonio Medina. 2004. « Algebraic approach for dependable logic control systems design ». *International Journal of Production Research* 42 (14): 2859-76. https://doi.org/10.1080/00207540410001705266.

Roussel, Jean-Marc, et Alessandro Giua. 2005. « DESIGNING DEPENDABLE LOGIC CONTROLLERS USING THE SUPERVISORY CONTROL THEORY ». *IFAC*

Proceedings Volumes, 16th IFAC World Congress, 38 (1): 56-61. https://doi.org/10.3182/20050703-6-CZ-1902.01434.

Sachidananda, Madhu, John Erkoyuncu, Daniel Steenstra, et Sandra Michalska. 2016. « Discrete Event Simulation Modelling for Dynamic Decision Making in Biopharmaceutical Manufacturing ». *Procedia CIRP*, The Second CIRP Conference on Biomanufacturing, 49 (janvier): 39-44. https://doi.org/10.1016/j.procir.2015.07.026.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen, et D. Teneketzis. 1995. « Diagnosability of discrete-event systems ». *IEEE Transactions on Automatic Control* 40 (9): 1555-75. https://doi.org/10.1109/9.412626.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen, et D.C. Teneketzis. 1996. « Failure diagnosis using discrete-event models ». *IEEE Transactions on Control Systems Technology* 4 (2): 105-24. https://doi.org/10.1109/87.486338.

Sánchez, Antonia M., et Francisco J. Montoya. 2006. « Safe Supervisory Control Under Observability Failure ». *Discrete Event Dynamic Systems* 16 (4): 493-525. https://doi.org/10.1007/s10626-006-0022-8.

Sayama, H. 2015. *Introduction to the Modeling and Analysis of Complex Systems*. https://textbooks.opensuny.org/introduction-to-the-modeling-and-analysis-of-complex-systems/.

Schafaschek, Germano, Max H. de Queiroz, et José E. R. Cury. 2014. « Local Modular Supervisory Control of Timed Discrete-Event Systems ». *IFAC Proceedings Volumes*, 12th IFAC International Workshop on Discrete Event Systems (2014), 47 (2): 271-77. https://doi.org/10.3182/20140514-3-FR-4046.00074.

Sethi, Andrea Krasa, et Suresh Pal Sethi. 1990. « Flexibility in Manufacturing: A Survey ». *International Journal of Flexible Manufacturing Systems* 2 (4): 289-328. https://doi.org/10.1007/BF00186471.

Shu, S., et F. Lin. 2014. « Fault-Tolerant Control for Safety of Discrete-Event Systems ». *IEEE Transactions on Automation Science and Engineering* 11 (1): 78-89. https://doi.org/10.1109/TASE.2013.2264809.

Simon, Gyula, Tamás Kovácsházy, et Gábor Péceli. 2002. « Transient Management in Reconfigurable Control Systems ». Technical Report. Budapest University of Technology and Economics.

Sohaleh, Hamed. 2017. « RECONFIGURABLE MANUFACTURING SYSTEM:AN ENABLER FOR COMPETITIVENESS FOR TODAY'S INDUSTRY ». In .

Sokolowski, John A., et Catherine M. Banks. 2012. *Handbook of Real-World Applications in Modeling and Simulation*. John Wiley & Sons.

Son, Sung-Yong, Tava Lennon Olsen, et Derek Yip-Hoi. 2001. « An approach to scalability and line balancing for reconfigurable manufacturing systems ». *Integrated Manufacturing Systems* 12 (7): 500-511. https://doi.org/10.1108/09576060110407815.

Staroswiecki, Marcel, et mireille Bayart. 1994. *Actionneurs intelligents*. Paris : Hermès.

Stefanovski, Jovan D. 2018. « Passive fault tolerant perfect tracking with additive faults ». *Automatica* 87 (janvier): 432-36. https://doi.org/10.1016/j.automatica.2017.09.011.

Sülek, Ayse Nur, et Klaus Werner Schmidt. 2013. « Computation of Fault-Tolerant Supervisors for Discrete Event Systems* ». *IFAC Proceedings Volumes*, 4th IFAC Workshop on Dependable Control of Discrete Systems, 46 (22): 115-20. https://doi.org/10.3182/20130904-3-UK-4041.00023.

Swamidass, Paul M., et William T. Newell. 1987. « Manufacturing Strategy, Environmental Uncertainty and Performance: A Path Analytic Model ». *Management Science* 33 (4): 509-24.

Tahiri, Imane, Philippot Alexandre, V. Carre-Menetrier, et A Tajer. 2018. « Timed synthesis control approach for tolerant-fault control of Discrete Event Systems (DES) ». In

ICCAD'18 : IEEE-International Conference on Control, Automation and Diagnosis. MARRAKECH, Morocco. https://hal.archives-ouvertes.fr/hal-02113921.

Tahiri, Imane, Alexandre Parant, François Gellot, Alexandre Philippot, et Véronique Carre-Menetrier. 2020. « Design and application of a reconfigurable control to a cyber-physical system », 7 juillet 2020.

Tahiri, Imane, A. Philippot, V. Carre-Menetrier, et Abdelouahed Tajer. 2019. « Two Cases of Study for Control Reconfiguration of Discrete Event Systems (DES) ». In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. Prague, Czech Republic. https://doi.org/10.5220/0007977301200129.

Tahiri, Imane, Alexandre Philippot, Véronique Carré-Ménétrier, et Abdelouahed Tajer. 2019a. « Time-Based Estimator for Control Reconfiguration of Discrete Event Systems (DES) ». In . Paris-France.

———. 2019b. « Approche distribuée pour la reconfiguration de la commande des systèmes manufacturiers ». In *MSR 2019 - 12ème Colloque sur la Modélisation des Systèmes Réactifs, Nov 2019, Angers, France*. Angers, France. https://hal.archives-ouvertes.fr/hal-02432725.

Tahiri, Imane, A Tajer, Y Qamsane, et A. Philippot. 2016. « Contribution à la commande des systèmes à événements discrets temporisés (SEDTS) ». In *Conférence Francophone d'Optimisation et Simulation (MOSIM)*. Montréal, Canada. https://hal.archives-ouvertes.fr/hal-02114230.

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.

Thiare, Ousmane. 2007. « Exclusion mutuelle de groupes dans les systèmes distribués ». Phdthesis, Université de Cergy Pontoise. https://tel.archives-ouvertes.fr/tel-00198862.

Thomas, Edouard. 2009. « Contribution to dynamic decision-making for predictive maintenance: formalisation of an opportunity principle. » Theses, Université Henri Poincaré - Nancy I. https://tel.archives-ouvertes.fr/tel-00420000.

Thomas, Edouard, Eric Levrat, Benoît Iung, et Pierre Cocheteux. 2009. « Opportune maintenance and predictive maintenance decision support ». In *13th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'2009*, édité par IFAC, CDROM. Moscou, Russia: IFAC. https://hal.archives-ouvertes.fr/hal-00403887.

Trentesaux, Damien. 2002. « Pilotage hétérarchique des systèmes de production », 117.

Tuptuk, Nilufer, et Stephen Hailes. 2018. « Security of smart manufacturing systems ». *Journal of Manufacturing Systems* 47 (avril): 93-106. https://doi.org/10.1016/j.jmsy.2018.04.007.

Upton, David M. 1994. « The Management of Manufacturing Flexibility ». *California Management Review*, janvier. https://doi.org/10.2307/41165745.

Verlinde, Christian. 1989. « Contribution à l'étude des architectures de systèmes automatisés ». l"Institut Polytechnique de Lorraine INPL: Université de Nancy.

Walden, D, D, G Roedler J, K Forsberg, R Hamelin D, et T Shortell M. 2015. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition*. San Diego.

Wallner, Heinz Peter. 1999. « Towards sustainable development of industry: networking, complexity and eco-clusters ». *Journal of Cleaner Production* 7 (1): 49-58. https://doi.org/10.1016/S0959-6526(98)00036-5.

Wang, Guo Xin, Si Han Huang, Yan Yan, et Jing Jun Du. 2017. « Reconfiguration Schemes Evaluation Based on Preference Ranking of Key Characteristics of Reconfigurable Manufacturing Systems ». *The International Journal of Advanced Manufacturing Technology* 89 (5): 2231-49. https://doi.org/10.1007/s00170-016-9243-7.

Wang, Jun, Shaoping Wang, Xingjian Wang, Cun Shi, et Mileta M. Tomovic. 2016. « Active fault tolerant control for vertical tail damaged aircraft with dissimilar redundant actuation system ». *Chinese Journal of Aeronautics* 29 (5): 1313-25. https://doi.org/10.1016/j.cja.2016.08.009.

Wang, Wencai, et Yoram Koren. 2012. « Scalability planning for reconfigurable manufacturing systems ». *Journal of Manufacturing Systems* 31 (2): 83-91. https://doi.org/10.1016/j.jmsy.2011.11.001.

Wardi, Y., C. G. Cassandras, et X. R. Cao. 2018. « Perturbation analysis: A framework for data-driven control and optimization of discrete event and hybrid systems ». *Annual Reviews in Control* 45 (janvier): 267-80. https://doi.org/10.1016/j.arcontrol.2018.04.003.

Whitehead, B.M, F Boschee, et R.H Decker. 2012. *The principal: Leadership for a global society*.

Wonham, W. M., Kai Cai, et Karen Rudie. 2018a. « Supervisory Control of Discrete-Event Systems: A Brief History ». *Annual Reviews in Control* 45 (janvier): 250-56. https://doi.org/10.1016/j.arcontrol.2018.03.002.

———. 2018b. « Supervisory control of discrete-event systems: A brief history ». *Annual Reviews in Control*, avril. https://doi.org/10.1016/j.arcontrol.2018.03.002.

Wonham, W. M., et P. J. Ramadge. 1988. « Modular Supervisory Control of Discrete-Event Systems ». *Mathematics of Control, Signals and Systems* 1 (1): 13-30. https://doi.org/10.1007/BF02551233.

Yangui, Rahma. 2016. « UML/B modeling for the safety requirements validation of railway operating rules ». Theses, Ecole Centrale de Lille. https://tel.archives-ouvertes.fr/tel-01450737.

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.

Zhang, Youmin, et Jin Jiang. 2008. « Bibliographical review on reconfigurable fault-tolerant control systems ». *Annual Reviews in Control* 32 (2): 229-52. https://doi.org/10.1016/j.arcontrol.2008.03.008.

Zhong, H., et W. M. Wonham. 1990. « On the consistency of hierarchical supervision in discrete-event systems ». *IEEE Transactions on Automatic Control* 35 (10): 1125-34. https://doi.org/10.1109/9.58555.

Ziller, Roberto, et Klaus Schneider. 2003. « Reducing Complexity of Supervisor Synthesis ». *IFAC Proceedings Volumes*, 2nd IFAC Conference on Control Systems Design (CSD '03), Bratislava, Slovak Republic, 7-10 September 2003, 36 (18): 183-91. https://doi.org/10.1016/S1474-6670(17)34666-9.

Ziv, Nicolas. 2018. « Enrichment of functional analysis for the construction sector by the integration of systems engineering and constructibility : application to the multifunctional metro ». Theses, Université Paris-Est. https://tel.archives-ouvertes.fr/tel-02144013.

Résumé

Les travaux présentés dans ce mémoire de thèse entrent dans le cadre de la reconfiguration de la commande des systèmes automatisés de production (SAP) vus comme une classe de Systèmes à Evénements Discrets (SED). La reconfiguration est basée sur la théorie de contrôle par supervision (SCT) et déclenchée suite à une détection des défauts sur la partie opérative (commande tolérante aux fautes). La contribution de ce mémoire est basée sur une synthèse de contrôle sûre fondée sur des propriétés temporisées. Si un défaut de capteur est détecté, le contrôleur du comportement normal est reconfiguré en un comportement dégradé où les informations temporisées compensent les informations perdues sur le capteur défectueux. Le basculement entre le comportement normal et le comportement dégradé est assuré par des règles de reconfiguration. L'objectif principal de notre méthode est d'implémenter le contrôle obtenu dans un automate programmable industriel (API). L'approche développée dans la thèse repose sur une architecture de contrôle distribuée des systèmes automatisés de production pour éviter l'explosion combinatoire récurrente des approches à base de SCT. Pour répondre aux différents objectifs, nous proposons une méthode pour traduire les contrôleurs distribués des deux modes de fonctionnement, ainsi que les règles de reconfiguration en différents grafcets implantables dans un API. L'implémentation de ces différents modèles est vérifiée par une technique de model-checking avant d'être testé sur un jumeau numérique. Enfin, nous appliquons notre contribution à un système réel pour illustrer nos résultats.

Mots clés : Reconfiguration, commande tolérante aux fautes, systèmes automatisés de production, systèmes à événements discrets, conception de la commande, théorie de contrôle par supervision, jumeau numérique

Abstract

The research results presented in this thesis falls within the framework of control reconfiguration of Automated Production Systems (APS) seen as a class of Discrete Event Systems (DES). The reconfiguration is based on Supervisory control theory (SCT) and triggered following a plant fault detection (fault tolerant control). The faults considered in this report are sensor faults (a sensor which remains blocked on reading 1 or on reading 0). The main contribution is based on a safe control synthesis founded on timed properties. In fact, if a sensor fault is detected, the controller of the normal behavior is reconfigured to a timed one where the timed information compensates the information lost on the faulty sensor. The switch between normal and reconfigured behaviors is ensured using reconfiguration rules. The main objective of our method is to implement the obtained control into a PLC. To meet the different objectives, we propose a method to translate the distributed controllers of the two operating modes, as well as the reconfiguration rules into different Grafcets implantable in a PLC. The implementation of these different models is verified by a model-checking technique before being tested on a digital twin. Finally, we apply our contribution to a real system to illustrate our results.

Keywords: Reconfiguration, fault tolerant control, manufacturing systems, discrete event systems, control design, supervisory control theory, digital twin

Discipline : AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE

Spécialité : Automatique et Traitement de Signal

Université de Reims Champagne-Ardenne

CRESTIC - EA 3804