Théorie du contrôle par supervision :

Un exemple d'application d'une approche décentralisée sur un système de production manufacturière

Laurent Piétrac — Samir Chafik — Eric Niel

Laboratoire d'Automatique Industrielle, INSA de Lyon
Bâtiment Antoine de Saint Exupéry
25 Avenue Jean Capelle, 69621 Villeurbanne CEDEX
{laurent.pietrac ; samir.chafik ; eric.niel}@lai.insa-lyon.fr

RÉSUMÉ.
La théorie du contrôle par supervision, ou théorie de Ramadge et Wonham, repose sur le principe de la synthèse. Elle permet la séparation claire entre le modèle du procédé et le modèle des spécifications. Cette théorie a de nombreuses extensions et applications dans le cadre des Systèmes à Événements Discrets (SED). Néanmoins les problèmes de la taille des modèles et de leur implantation sont toujours évoqués comme frein à son utilisation. Dans cet article nous allons traiter un cas réel avec une approche apportant une solution à ces deux problèmes. Notre premier apport réside dans la méthode de modélisation utilisée qui exploite en profondeur les atouts de l'approche décentralisée. Le second consiste à définir des règles permettant de choisir une trajectoire de commande parmi celles possibles, afin de l'implanter dans un automate programmable.

ABSTRACT.
The supervisory control theory, also called theory of Ramadge and Wonham, lies on the synthesis principle. It enables a sharp distinction between the process model and the specifications model. This theory has great extensions and applications in the discrete event systems (DES) studies. Nevertheless the models size and their implementation are problems usually point out as a brake to the application of this theory. In this article we suggest an approach that induce an answer to this two problems. This approach will be explained through an example. Our first contribution is in the used method, which is based on a decentralized approach. The second one leads to define rules that allow us to choose a command trajectory among those possible in order to implement it in a programmable logic controller.

MOTS-CLÉS : *théorie du contrôle par supervision, synthèse, approche décentralisée, propriétés, programmation API, conception.*

KEYWORDS: *supervisory control theory, decentralised concept, properties, PLC programming, modelling.*

APII-JESA. Volume 38 – n° 3-4/2004, pages 315 à 346

1. Introduction

Donner l'assurance que le système conçu est sûr de fonctionnement implique qu'il est possible de démontrer que celui-ci respecte un certain nombre de propriétés. Dans le cadre des Systèmes à Événements Discrets (SED), la théorie du contrôle par supervision (ou théorie RW), initiée par Ramadge et Wonham (Ramadge *et al.*, 1987), est justement basée sur la séparation entre le modèle du procédé et le modèle des propriétés que doit respecter ce procédé (propriétés appelées spécifications dans le cadre de la théorie). Son utilisation implique donc l'expression des spécifications ce qui participe à une plus grande sûreté de fonctionnement du système conçu. Cette théorie basée sur la théorie des langages et sur les automates à états permet de définir des propriétés des modèles, notamment la contrôlabilité, et conduit à une étude plus rigoureuse des SED.

Les travaux initiaux de Ramadge et Wonham (Ramadge *et al.*, 1987) utilisent la notion d'événement pour définir des trajectoires correspondant aux évolutions possibles du système. Avec un point de vue centralisé le modèle du procédé décrit toutes les trajectoires réalisables par le système alors que le modèle des spécifications décrit ce qu'il ne peut pas faire ou au contraire ce qu'il doit absolument faire. Cette approche, dite approche dynamique, est donc basée sur la notion de succession d'événements. Une autre approche, dite statique, est au contraire basée sur la notion d'état (Kumar *et al.*, 1993) (Khatab, 2000) (Marchand *et al.*, 2002). Quel que soit le langage de modélisation, les spécifications sont des interdictions d'états particuliers. Dans ces deux approches les évolutions considérées du système sont des évolutions logiques puisque le temps n'est pas considéré sous forme d'une variable. D'autres travaux ont donc étendu la théorie en considérant des systèmes dont l'évolution dépend à la fois de l'occurrence d'événements et de l'instant auquel cette occurrence a lieu (Brandin *et al.*, 1994) (Khatab, 2000) (Altisen *et al.*, 2002).

Notre objectif dans cet article est de proposer une approche mettant en œuvre la théorie de Ramage et Wonham pour l'étude de la commande des Systèmes Automatisés de Production (SAP) et non pas de comparer cette théorie à d'autres approches. Les problèmes à résoudre sont la taille des modèles et la détermination des instructions de programmation des Automates Industriels Programmables (API) à partir de ces modèles. La recherche de la diminution de la taille des modèles nous a amenés à nous placer dans le cadre de l'approche dynamique car elle a donné lieu à de nombreux travaux cherchant à résoudre ce problème par la remise en cause du point de vue centralisé : point de vue hiérarchique (Zong *et al.*, 1990), modulaire (Ramadge *et al.*, 1988) ou décentralisé (Lin *et al.*, 1990). Comme pour l'ensemble de ces travaux, nous utiliserons des automates à états, ce qui nous permettra de nous concentrer sur le problème de la taille des modèles et d'écarter les problèmes liés à la modélisation du temps.

Dans la prochaine section nous présenterons le système qui nous a servi de test (Regimbal, 2002). Nous ferons ensuite quelques rappels sur la théorie du contrôle

par supervision et sur les extensions nécessaires à la compréhension de cet article. Nous présenterons alors notre proposition couvrant toutes les étapes de modélisation du procédé et des spécifications, de choix, d'interprétation puis d'implantation des trajectoires de commande. L'ensemble des étapes de la méthode proposée sera ensuite appliqué à un cas réel, ce qui permettra de comparer les modèles à ceux qui auraient été obtenus par l'approche centralisée.

2. Présentation du système étudié

2.1 Le poste de travail

Le système étudié est implanté au sein de l'Atelier Inter-établissement de Productique (AIP-PRIMECA) Rhône-Alpes Ouest. Il s'agit d'une chaîne de production de type « transfert libre » comportant un convoyeur central et six convoyeurs de dérivation. Les pièces à traiter sont portées par des palettes se déplaçant, par frottement, sur les convoyeurs. Chaque convoyeur de dérivation dessert un poste de travail, qui peut être un poste d'assemblage, d'usinage ou de contrôle. Tous les postes de travail ont des structures identiques, aussi notre étude ne portera-t-elle que sur un seul poste.

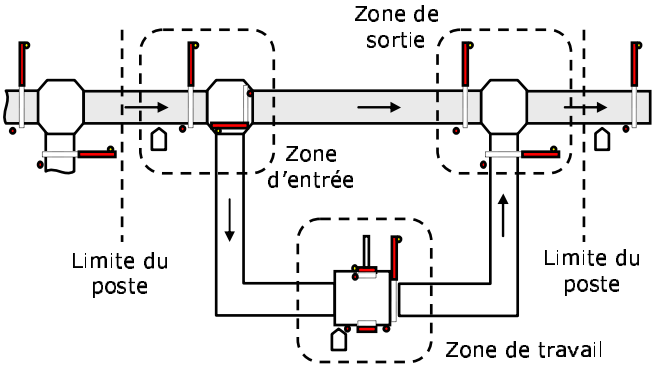


figure 1. *Structure d'un poste de travail*

Un poste de travail peut être décomposé en trois zones fonctionnelles entre lesquelles les palettes circulent sur les différents convoyeurs. Sur la figure 1 les palettes arrivent du poste précédent par la gauche. La zone d'entrée permet de déterminer si la pièce transportée par la palette doit continuer son chemin vers le poste suivant (en passant par la zone de sortie) ou doit être dérivée vers la zone de travail du poste considéré. Le choix du trajet se fait grâce aux informations contenues dans l'étiquette magnétique portée, en plus de la pièce, par chaque palette. Ces informations sont stockées sous forme d'une liste des zones de travail par lesquelles doit passer la palette pour que la réalisation de la pièce soit complète.

Évidemment, c'est dans la zone de travail que sont réalisées les opérations sur la pièce, tandis que la fonction de la zone de sortie est d'interdire les collisions entre les pièces provenant de la zone de travail et celles arrivant de la zone d'entrée.

2.2 La zone d'entrée

Cette zone comprend quatre composants : un capteur de palette *Ce*, un système de lecture/écriture des étiquettes magnétiques *Lce*, un mécanisme de blocage des palettes *Be* et un aiguillage *Ag* permettant de dévier les palettes. Sur la figure 2 les palettes arrivent de la gauche et suivant le code lu sur l'étiquette magnétique par la borne magnétique, les palettes sont dirigées vers la zone de travail (convoyeur de dérivation) ou vers la zone de sortie (convoyeur central).

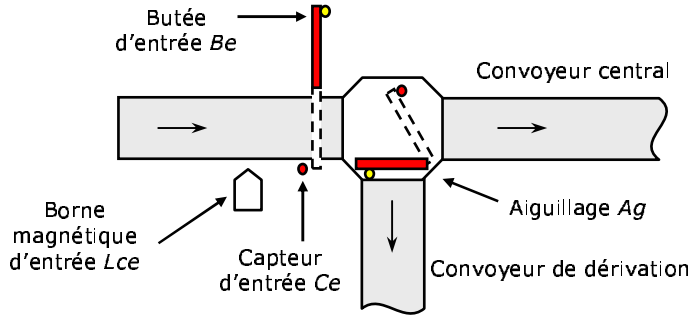


figure 2. *Zone d'entrée*

2.3 La zone de travail

La zone de travail est composée de trois sous-systèmes qui sont la butée *Bt* et le capteur de présence palette *Cp*, l'indexeur *Id* et enfin le système de lecture/écriture magnétique *Lct* (voir figure 3). Lorsqu'une palette est détectée, elle est bloquée puis indexée pour subir une opération d'usinage. Lorsque l'opération est terminée, la gamme de fabrication contenue sur l'étiquette est modifiée afin que cette palette ne revienne pas sur ce poste.

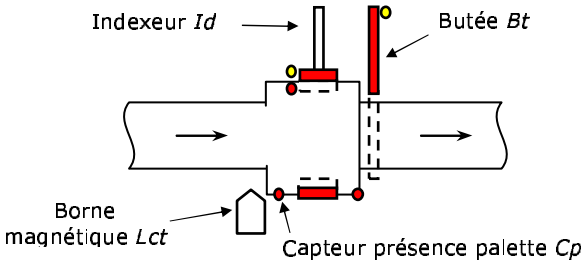


figure 3. *Zone de travail*

2.4 La zone de sortie

Cette zone comprend deux butées *B2* et *B3*. Elle permet de gérer la remise des palettes provenant de la zone de travail sur le convoyeur central. C'est une zone de conflit car il ne peut y avoir qu'une seule pièce dans la zone comprise entre les butées et le capteur d'entrée du poste suivant (figure 4). De plus lorsque deux palettes sont présentes en *B2* et *B3*, il est nécessaire de gérer leur priorité de passage.

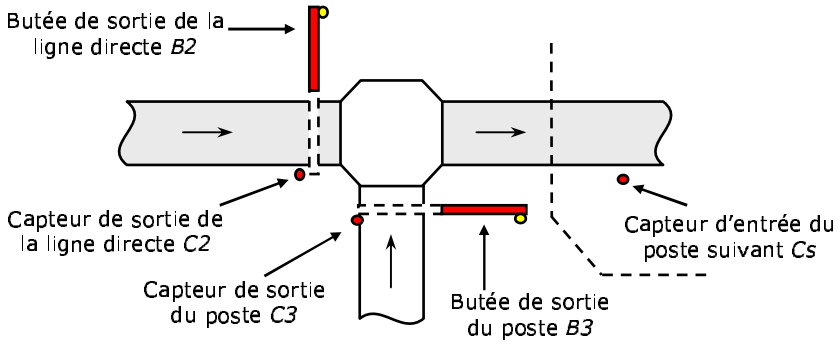


figure 4. *Zone de sortie*

2.5 Interactions entre zones

Les convoyeurs compris entre les trois zones sont d'une taille assez faible qui limite le nombre de palettes potentiellement présentes. Cette limite est de cinq dans les trois cas, ce qui permet d'avoir un stock tampon sur chacun des convoyeurs. Le convoyeur compris entre la zone de sortie et le poste suivant ne peut contenir qu'une seule palette, ce qui sera modélisé comme un stock de capacité unitaire (stock 4 de la figure 5).

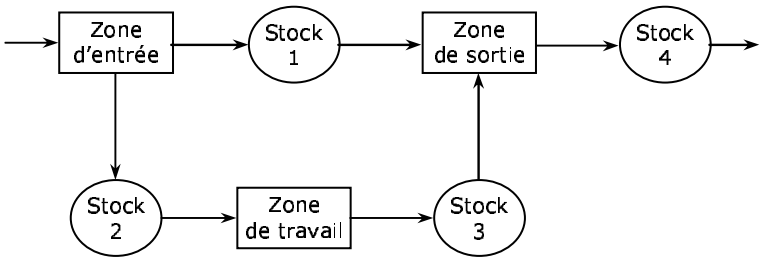


figure 5. *Position des stocks*

3. Théorie du contrôle par supervision

3.1. Approche centralisée

La théorie du contrôle par supervision a été initiée par Ramadge et Wonham en 1987 dans le cadre de la commande des systèmes à événements discrets (SED). Elle repose sur la séparation explicite entre le modèle du procédé et celui des propriétés que doit respecter ce procédé. Ces propriétés seront appelées spécifications par la suite pour respecter la terminologie de la théorie RW. Ces modèles permettent de déterminer, s'il existe, un superviseur (figure 6) dont le rôle est d'interdire l'occurrence d'événements, imposant ainsi au procédé de respecter les spécifications modélisées.

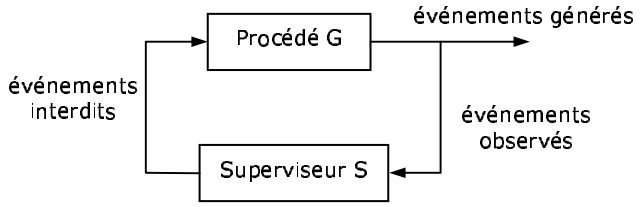


figure 6. *Modèles de la théorie RW*

Par hypothèse, les événements générés par le procédé sont :

- spontanés : aucun mécanisme ne vient contraindre leur occurrence ;
- instantanés : ils ont une durée nulle ;
- asynchrones : aucune horloge ne séquence l'occurrence des événements.

L'ensemble des événements générés par le procédé est appelé alphabet du système, il est noté Σ (Wonham, 2002). Cet alphabet est partitionné en deux sous-ensembles disjoints : l'ensemble des événements contrôlables Σ_c et l'ensemble des événements incontrôlables Σ_{uc}. Il peut aussi être partitionné en deux autres sous-ensembles disjoints : l'ensemble des événements observables Σ_o et l'ensemble des événements inobservables Σ_{uo}. Le superviseur ne génère aucun événement, il observe l'ensemble Σ_o et interdit l'occurrence de certains événements de Σ_c.

Le procédé est modélisé par un automate $G= (X, \Sigma, \delta, x_0, X_m)$ où :

- X est l'ensemble des états de G ;
- Σ est l'ensemble des événements ;
- $\delta : X \times \Sigma \rightarrow X$ est la fonction partielle de transition ;
- x_0 est l'état initial unique ;
- X_m est l'ensemble des états marqués.

Soit Σ^* l'ensemble des mots construits sur l'alphabet Σ augmenté du mot vide ε.

La fonction δ est étendue à la fonction $X \times \Sigma^* \rightarrow X$ définie inductivement par :

- $\delta(x, \varepsilon) = x, x \in X$;
- $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma) = x'$ avec $\sigma \in \Sigma, s \in \Sigma^*$ et $x, x' \in X$.

L(G) représente le langage généré par G : $L(G) = \{s \in \Sigma^* \mid \delta(x_0, s)\ !\ \}$[1]

Les spécifications sont aussi modélisées par un automate E de langage L(E).

Le superviseur S est une fonction définie par :

$$S : L(G) \rightarrow \Gamma \text{ avec } \Gamma = \{\gamma \in Pwr(\Sigma) \mid \gamma \supseteq \Sigma_u\}$$

dans laquelle γ représente l'ensemble des événements autorisés par S et $Pwr(\Sigma)$ l'ensemble de tous les sous-ensembles de Σ.

Sauf dans des cas très simples, les automates G et E ne sont pas construits directement par le concepteur. L'automate G est le composé parallèle des automates G_i modélisant les éléments du procédé tandis que l'automate E est le produit des automates E_i modélisant des spécifications du cahier des charges. Tous ces automates sont construits sur l'alphabet Σ.

Si les spécifications sont contrôlables, l'automate S/G décrivant le procédé supervisé est le produit des automates G et E. Dans le cas contraire, l'algorithme de Kumar (Kumar, 1991) permet d'obtenir l'automate décrivant le plus grand ensemble possible des trajectoires contrôlables du procédé respectant les spécifications. Cet ensemble est appelé le maximum permissif.

3.2. Approche décentralisée

Le contrôle par supervision centralisé, où un seul superviseur contrôle le générateur, unique lui aussi, est un concept simple, général et pouvant être appliqué à une large étendue de systèmes (production, télécommunication, systèmes informatiques). Cependant, l'applicabilité de cette théorie peut être compromise par la complexité des calculs et la taille des superviseurs synthétisés. De nombreux travaux ont cherché à répondre à ces inconvénients. Trois approches sont à la base de ces développements : l'approche modulaire (Ramadge *et al.*, 1988), l'approche décentralisée (Lin *et al.*, 1988) (Lin *et al.*, 1990) et l'approche hiérarchique (Zong *et al.*, 1990). L'approche modulaire propose de coupler non pas un seul mais plusieurs superviseurs au même procédé G. L'approche décentralisée va plus loin puisqu'elle propose également une décomposition du procédé en procédés locaux. Quant à l'approche hiérarchique, elle utilise une décomposition « verticale » du procédé et du superviseur. Le niveau bas du procédé émet des événements permettant de faire évoluer le niveau haut. Les événements du superviseur de

[1] $\delta(x_0, s)\ !$ se lit la fonction delta de x_0 et de s existe.

niveau haut conditionnent l'évolution du superviseur de niveau bas. Nous utiliserons une approche décentralisée, qui correspond bien au système étudié dans lequel différentes zones évoluent en parallèle, avec des interactions liées au flux de pièces.

L'intérêt de cette approche est donc que chaque superviseur local $S_{i,loc}$ supervise un procédé local G_i construit sur un alphabet Σ_i sous-ensemble de l'alphabet global Σ. Cependant il est nécessaire de vérifier les conditions d'existence de ces superviseurs locaux. Il faut également déterminer si le comportement permis par les procédés locaux G_i contrôlés par les superviseurs locaux $S_{i,loc}$ est non bloquant et aussi permissif que celui permis par le procédé global G sous contrôle du superviseur global S. Pour ce faire, il est nécessaire de rappeler la notion de projection naturelle P_i. Elle consiste à éliminer les événements non retenus dans le contrôle i.e. qui n'appartiennent pas à Σ_i.

La projection P_i est définie comme suit :

$$P_i : \Sigma \rightarrow \Sigma_i \cup \{\varepsilon\}$$

$$\sigma \mapsto P_i(\sigma) = \begin{cases} \sigma \text{ si } \sigma \in \Sigma_i \\ \varepsilon \text{ sinon} \end{cases}$$

Notons que P_i, qui est définie sur l'alphabet Σ_i, peut être étendue à une projection $P_{i,ext}$ définie sur un langage telle que :

$$P_{i,ext} : \Sigma^* \rightarrow \Sigma_i^*$$
$$P_{i,ext}(\varepsilon) = \varepsilon \text{ et } P_{i,ext}(\omega\sigma) = P_{i,ext}(\omega)P_i(\sigma) \text{ avec } \omega \in \Sigma^* \text{ et } \sigma \in \Sigma.$$

L'action de $P_{i,ext}$ consiste à éliminer de la chaîne $\omega \in \Sigma^*$ tous les événements σ de ω qui n'appartiennent pas à Σ_i. Afin de simplifier l'écriture, P_i désignera par la suite $P_{i,ext}$.

3.2.1. Contrôle local

Un superviseur local $S_{i,loc}$ contrôle uniquement des événements de Σ_i[2]. En boucle fermée avec G_i ce superviseur $S_{i,loc}$ synthétise le langage $K_{i,loc} = L(S_{i,loc}/G_i)$. Pour vérifier l'existence de $S_{i,loc}$ il est nécessaire et suffisant de vérifier que $K_{i,loc}$ est contrôlable.

3.2.2. Contrôle global

Pour caractériser le fonctionnement global d'un système à événements discrets à partir de modèles locaux définis sur des Σ_i, nous utilisons la projection inverse P_i^{-1} qui est définie, de manière complémentaire à P_i, de la façon suivante :

---

[2] Les événements de Σ_i sont observables pour $S_{i,loc}$ alors que les événements de Σ-Σ_i sont inobservables.

$$P_i^{-1} : Pwr(\Sigma_i^*) \rightarrow Pwr(\Sigma^*)$$
$$P_i^{-1}(\omega) = \{\omega' \in \Sigma^* \mid P_i(\omega') = \omega \text{ avec } \omega \in \Sigma_i^*\}$$

Un superviseur local $S_{i,loc}$ contrôle uniquement les événements appartenant à $\Sigma_{i,c}$[3]. $S_{i,loc}$ peut être étendu à un superviseur global $\widetilde{S}_i$ qui synthétisera alors le sous-langage $\widetilde{K}_i = L(G) \cap P_i^{-1} K_{i,loc}$. Son correspondant centralisé S_i aurait synthétisé le sous-langage $K_i = L(S_i/G)$.

$\widetilde{S}_i$ est un superviseur qui a la même action de contrôle que $S_{i,loc}$ sur $\Sigma_{i,c}$ mais de plus, par hypothèse, il autorise tous les événements appartenant à $(\Sigma\text{-}\Sigma_{i,c})$. Par conséquent $\widetilde{S}_i$ aura la définition suivante :

$$\widetilde{S}_i : L(G) \rightarrow Pwr(\Sigma)$$
$$t \mapsto S_{i,loc}(P_i(t)) \cup (\Sigma - \Sigma_i)$$

Pour vérifier que les procédés locaux sous contrôle local ont le même comportement que le procédé global sous contrôle global, il faut vérifier que le superviseur décentralisé $\widetilde{S}_i$ peut synthétiser le même fonctionnement que celui d'un superviseur centralisé S_i. Cela revient à vérifier que $\widetilde{K}_i = K_i$. Cette identité ne peut pas être vérifiée dans tous les cas car la propriété d'observabilité des Σ_i par rapport à $S_{i,loc}$ n'est pas conservée par l'union des superviseurs.

Les auteurs de (Lin *et al.*, 1988) ont défini une classe particulière de langage $L_i \subseteq L(G)$ dit P_i-normal si : $L_i = L(G) \cap P_i^{-1} P_i(L_i)$. Le grand avantage de cette propriété de normalité est qu'elle est conservée par l'union, aussi si tous les K_i sont normaux, on aura bien $\widetilde{K}_i = K_i$. En outre, la normalité implique l'observabilité, et il a même été démontré (Cassandras *et al.*, 1999) que sous certaines conditions normalité et observabilité sont équivalentes.

Revenons un instant sur $\widetilde{S}_i$ qui est l'extension du superviseur local $S_{i,loc}$. Pour déterminer cette extension, il faut savoir si $S_{i,loc}$ autorise ou interdit les événements de $(\Sigma - \Sigma_i)$. Notons que lorsqu'un événement $\sigma \in \Sigma_{i,c}$, il peut être autorisé ou interdit par $S_{i,loc}$. Cependant, lorsque cet événement $\sigma \in (\Sigma - \Sigma_{i,c})$ serait-il néanmoins autorisé par $S_{i,loc}$?

Le lemme 1 montre que lorsque $\sigma \in (\Sigma - \Sigma_{i,c})$, il est toujours autorisé par $S_{i,loc}$.

Lemme 1 : soient Σ un alphabet de $L(G)$, $\Sigma_i \subset \Sigma$, K_i un langage normal et $S_{i,loc}$ défini comme précédemment, alors $\forall \sigma \in \Sigma - \Sigma_{i,c}$, σ est autorisé par $S_{i,loc}$.

Preuve : voir (Chafik, 2000) p. 76.

---

[3] $\Sigma_{i,c}$ est l'alphabet local contrôlable.

En conclusion, en supposant la normalité de K_i ($K_i = \widetilde{K}_i$) et la contrôlabilité locale de G, alors le contrôle décentralisé obtenu est optimal dans le sens où le fonctionnement de l'ensemble des superviseurs locaux liés à G est équivalent à celui du superviseur centralisé. Ce résultat montre que si G est localement contrôlable, alors la supervision décentralisée est optimale ($\bigcap_{i=1}^{n} \widetilde{K}_i = \bigcap_{i=1}^{n} K_i$).

3.3. Approche proposée

L'approche décentralisée constitue un cadre formel de décomposition du procédé et du superviseur. C'est une approche générale, utilisable pour l'ensemble des SED. Notre apport consiste plus particulièrement à proposer une démarche d'application de cette approche aux systèmes automatisés de production (SAP). Notre objectif est de mettre en place une approche rigoureuse de construction et d'analyse des modèles permettant en outre une implantation sous la forme de programmes d'automates programmables industriels (API).

Notre contribution couvre l'ensemble des étapes menant à une implantation :

a) La modélisation du procédé et des spécifications ;

b) Le choix de trajectoires de commande ;

c) L'interprétation pour l'implantation de ces trajectoires.

Nous allons développer ces trois points dans les sections suivantes, mais tout d'abord nous allons présenter certaines contraintes de fonctionnement de la partie commande à partir desquelles nous expliquerons et de justifierons notre approche.

3.3.1. Propriétés de la partie commande

Les parties opérative et commande d'un SAP sont des systèmes réactifs dans le sens où chacun réagit aux événements générés par son environnement. En particulier la partie opérative réagit aux ordres de la partie commande tandis que celle-ci évolue en fonction des comptes-rendus de la partie opérative (Balemi *et al.*, 1993). Le programme qui sera implanté dans la partie commande devra donc interpréter les signaux générés par la partie opérative afin de déterminer les ordres et les conditions de réalisation de ces ordres. La plupart du temps ces ordres correspondent aux événements contrôlables tandis que les événements incontrôlables sont interprétés par des comptes-rendus (Charbonnier, 1996). Cette règle n'est pas systématique : les auteurs de (Dietrich *et al.*, 2001) montrent par exemple que lorsque les consignes données par l'opérateur sont prises en compte dans le modèle, celles-ci correspondent à des événements contrôlables alors qu'elles seront des entrées de la partie commande. Cependant nous ne traitons pas ce cas particulier et nous interpréterons donc systématiquement un événement contrôlable comme une évolution des sorties de la partie commande. Quant aux événements

incontrôlables, ils seront systématiquement interprétés comme une évolution des entrées de la partie commande ou comme une évolution de son état interne.

Le comportement de la partie commande doit respecter un certain nombre de contraintes qui ont déjà été évoquées dans (Piétrac *et al.*, 2002). Les auteurs de (Hellgren *et al.*, 2002) ont également défini des contraintes très proches de celles énoncées ici, mais par rapport à une méthode basée sur des Réseaux de Petri paramétrés. Les contraintes du programme de commande, dues au comportement attendu de cette partie commande, sont les suivantes :

C1 : Le programme définit des séquences d'actions déterminées en fonction des comptes-rendus de la partie opérative. Donc il doit générer tous les changements d'états des sorties de la partie commande, et doit être à même d'observer les changements d'états des entrées de la partie commande pertinentes au vu de son état courant.

C2 : Cette obligation d'observer tout changement d'état d'une entrée revient à considérer que la partie commande est infiniment plus rapide que la partie opérative, ce qui est une hypothèse classique de l'étude des SAP (Lhoste *et al.*, 1997). La première conséquence de cette hypothèse est que la partie commande est capable de distinguer deux variations successives de ses entrées. La seconde est que la variation des sorties consécutive à la variation d'une entrée se fait à temps nul, en considérant une base de temps externe à la partie commande.

C3 : Ensuite le programme doit être déterministe par rapport à ses sorties, c'est à dire que dans un état donné il ne peut pas avoir le choix des ordres à donner.

C4 : Il doit aussi être déterministe par rapport au cycle de fonctionnement interne de l'automate. Quel que soit le type d'automate ou le langage utilisé, les conditions d'affectation des sorties sont évaluées de manière séquentielle. Pour chacune de ces conditions l'image interne des sorties est modifiée à chaque évaluation d'une condition. La sortie prend donc comme valeur celle de la dernière condition évaluée. En conséquence il ne peut y avoir plusieurs affectations des sorties sous risque de ne pas avoir, lorsque ce n'est pas la dernière condition qui est validée, le comportement souhaité.

Nous verrons dans les sections suivantes comment ces contraintes ont influencé nos choix méthodologiques.

3.3.2. Modélisation du procédé et des spécifications

Un SAP est un système dans lequel chaque sous-système évolue en fonction de besoins locaux mais aussi en fonction des évolutions des autres sous-systèmes. Ce sont ces interactions entre sous-systèmes qui apportent une certaine complexité dans l'analyse.

Supposons que le système soit décomposable en différentes zones indépendantes les unes des autres. Nous appelons ces zones des zones structurelles. En appliquant l'approche décentralisée il est possible de construire, pour chacune de ces zones, un modèle du procédé G_i et un modèle des spécifications $E_{i,loc}$. Ces modèles étant construits sur des alphabets distincts sont non conflictuels. Par contre cela ne dispense pas de vérifier que chaque spécification $E_{i,loc}$ est contrôlable par rapport à G_i, et que l'ensemble est optimal. Une fois ces vérifications indispensables réalisées, les procédés sous contrôles locaux ($S_{i,loc}/G_i$) pourront être construits. Chacun de ceux-ci génère les trajectoires localement possibles.

Grâce au choix de zones structurelles ayant des alphabets disjoints, nous sommes sûrs que chaque sortie de la partie commande ne sera générée qu'une fois ce qui permettra de respecter la contrainte C4 de déterminisme par rapport au cycle automate et d'obtenir une structure de programme simple.

Dans la plupart des cas, les composants des différentes zones ne peuvent pas avoir un fonctionnement complètement indépendant (c'est le cas sur notre exemple). Ces interactions sont modélisées par de nouvelles spécifications. L'application classique de l'approche décentralisée consisterait à construire les procédés locaux correspondant aux alphabets de ces spécifications puis à vérifier si ces spécifications sont contrôlables par rapport à ces procédés, sont non conflictuelles avec les autres spécifications et enfin si l'ensemble est optimal. Ensuite il faudrait construire les nouveaux procédés sous contrôle et traduire ces modèles en programmes d'API.

Cependant ceci revient à retrouver plusieurs fois des sorties dans le programme, et donc à ne pas respecter la contrainte C4 de déterminisme par rapport au cycle automate. L'application directe de l'approche décentralisée n'est donc pas possible sur les spécifications d'interactions entre zones structurelles. Pour respecter la contrainte C4, nous traduisons ces spécifications sous la forme de conditions supplémentaires d'activations des sorties de la partie commande. Bien sûr cela ne nous dispense pas de vérifier si ces spécifications sont contrôlables, non conflictuelles et si le comportement global est optimal.

La structure finalement utilisée comprend donc des procédés locaux associés chacun à une spécification locale, avec des alphabets locaux disjoints. Le modèle du système comprend également des spécifications d'interactions, dont les alphabets ne sont évidemment pas disjoints avec ceux des spécifications précédentes (figure 7-a1). Dans l'étape de synthèse, seules les spécifications locales sont utilisées pour calculer des procédés sous contrôles locaux (figure 7-a2) ce qui est donc différent de l'application classique de l'approche décentralisée dans laquelle des procédés sous contrôle seraient aussi calculés pour les spécifications d'interactions.

Dans la section suivante nous allons détailler les règles de choix de trajectoires de commande (figure 7-b). Les modèles utilisés étant de types différents, nous distinguerons les règles utilisées pour les modèles de procédés sous contrôles de celles utilisées pour les spécifications d'interactions. Les règles d'interprétation des

trajectoires en programmes d'automates (figure 7-c) seront définies dans la section 3.3.4.

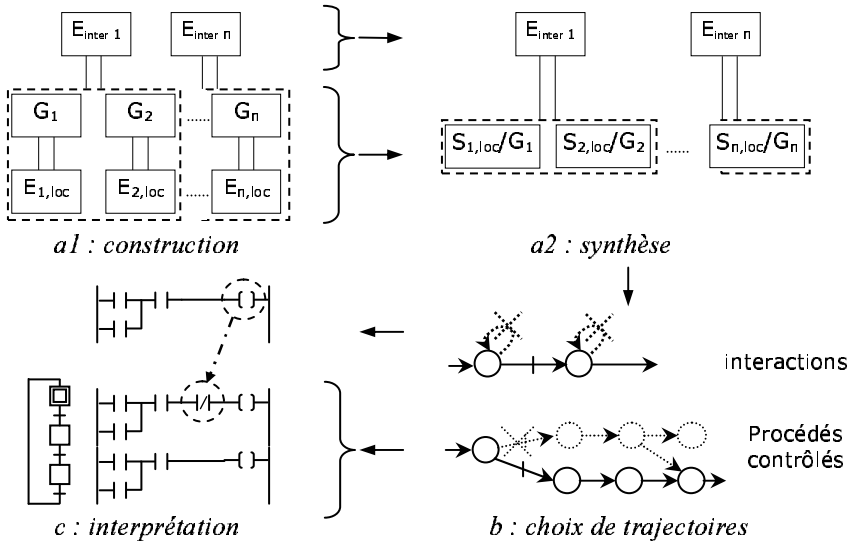


figure 7. *Etapes de la méthode*

3.3.3. Choix des trajectoires de commande

3.3.3.1. Les procédés sous contrôles

Le procédé sous contrôle synthétisé ne définit pas une seule trajectoire possible, c'est-à-dire que dans un état donné, plusieurs événements peuvent avoir lieu. Ces événements peuvent aussi bien être contrôlables qu'incontrôlables, aussi différents cas sont à étudier.

Lorsque plusieurs événements contrôlables sont possibles à partir d'un même état, il est nécessaire d'en choisir un et de supprimer les autres. Les événements contrôlables étant interprétés sous forme de sorties, il est en effet nécessaire de supprimer certaines trajectoires afin que le programme final respecte la contrainte C3 de déterminisme par rapport à ses sorties (exemple figure 8). Sur l'exemple présenté dans cet article, nous n'avons pas rencontré ce cas de figure, aussi nous n'avons pas développé de méthode particulière de choix de trajectoires. Cependant d'autres auteurs (Mohanty et *al.*, 2002) ont proposé des approches de recherche du contrôleur optimal en utilisant des fonctions de coût associées aux trajectoires de commande.

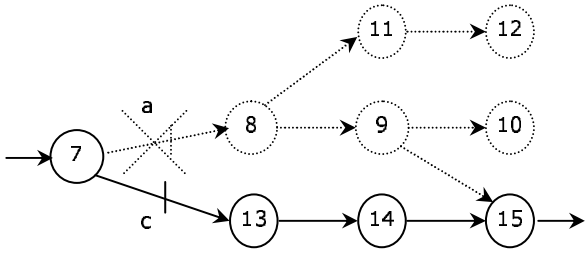


figure 8. *Application de la propriété de déterminisme*

Lorsque plusieurs événements incontrôlables sont possibles à partir d'un état, aucun choix n'est à faire. En effet, cela sera interprété comme plusieurs comptes-rendus de la partie opérative à la partie commande. Ceci est un comportement tout à fait normal du système qui ne nécessite aucune interprétation particulière.

Le dernier cas possible est celui où dans un état donné un événement contrôlable est possible ainsi que des événements incontrôlables. La dernière contrainte inutilisée jusqu'à présent est l'hypothèse sur la rapidité de la partie commande par rapport à la partie opérative (contrainte C2). En terme d'événements, cela signifie que dans un état donné, un événement contrôlable peut être généré plus rapidement qu'un événement incontrôlable. Par conséquent la trajectoire débutant par cet événement incontrôlable ne sera jamais suivie par le système (figure 9).

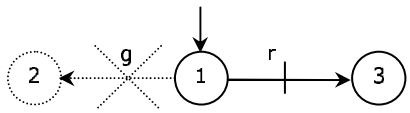


figure 9. *Application de la propriété de réactivité*

Ces différents cas ayant permis de supprimer un certain nombre de trajectoires, quelques états peuvent se retrouver inaccessibles ou non co-accessibles. Il est donc nécessaire de déterminer l'automate émondé de celui obtenu, c'est-à-dire l'automate dans lequel tous les états sont accessibles et co-accessibles. Ainsi, sur l'exemple de la figure 8, les états 8 à 12 sont supprimés car devenus inaccessibles après l'élimination de la transition entre les états 7 et 8.

3.3.3.2. Les spécifications d'interactions

Nous rappelons que les spécifications d'interactions seront traduites sous forme de conditions supplémentaires autorisant ou interdisant des sorties de la partie commande générées par le programme obtenu par la traduction des $S_{i,loc}/G_i$. (figure 7, page 327). Ces spécifications ont pour rôle d'observer le système et d'interdire certaines actions. Cette observation est représentée par une trajectoire d'événements incontrôlables, alors que l'interdiction d'une action correspond à l'interdiction d'un événement contrôlable. Par contre dans chaque état de la

spécification tous les événements qui ne font pas évoluer la spécification et qui ne sont pas interdits sont autorisés. Ces événements autorisés, représentés par une boucle sur l'état, ne sont donc pas des entrées-sorties de la partie commande pertinentes pour le programme puisqu'ils ne modifient pas son état interne ou externe. Ils ne sont donc pas traduits. Ceci revient à ne pas tenir compte des événements autorisés en boucle sur les états des modèles des spécifications.

3.3.4. Interprétation pour implantation de ces trajectoires

Une conséquence de la contrainte C1 est que le modèle traduit en programme doit être un générateur d'événements contrôlables traduits en sorties de la partie commande. Le superviseur ne générant aucun événement ne peut donc pas être le modèle traduit en programme. Par contre, dans l'approche centralisée, ce modèle peut correspondre aux trajectoires synthétisées puisque le modèle du procédé sous contrôle est un générateur d'événements (Liu *et al.*, 2002). Le modèle à traduire peut aussi être le couple procédé – superviseur (ou procédé – spécifications si celles-ci sont contrôlables) dans lequel le procédé génère tous les événements et le superviseur en interdit certains (De Queiroz *et al.*, 2002). Ces deux possibilités permettant de définir les mêmes trajectoires contrôlables, le comportement de la partie commande sera identique dans les deux cas.

Dans notre approche, les alphabets locaux sont une partition de l'alphabet total, c'est-à-dire qu'ils sont disjoints et que l'union de ces alphabets est égale à l'alphabet total. Pour chacun de ces alphabets locaux nous traduirons directement les procédés sous contrôle, ce qui nous garantit d'avoir un programme qui va effectivement générer toutes les sorties de la partie commande. Quant aux spécifications d'interactions entre zones, elles seront traduites sous forme de conditions supplémentaires, nécessaires à la génération de certaines sorties.

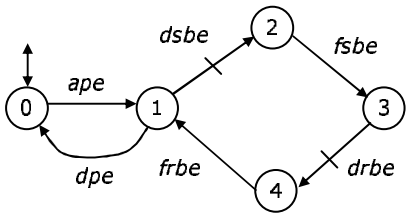
4. Application sur l'exemple

Dans la section 4.1 nous présenterons les modèles des composants du système. Ces modèles sont considérés comme les briques élémentaires dans la phase de modélisation. Nous présenterons ensuite dans la section 4.2 les modèles des spécifications locales ou intra zones (modèles $E_{i,loc}$ de la figure 7-a) ainsi que les alphabets locaux associés. Ceci nous permettra de déterminer dans la section 4.3 les procédés locaux (les G_i de la figure 7-a1) ainsi que les modèles des procédés sous contrôles locaux (modèles $S_{i,loc}/G_i$ de la figure 7-a2). Enfin la section 4.4 présente les modèles des spécifications d'interactions entre zones structurelles. Tous les modèles ayant été construits, nous comparerons notre approche à l'approche centralisée dans la section 4.5 avant de faire les choix de trajectoires dans la section 4.6 (fig. 7-b) puis de présenter l'implantation de ces trajectoires dans la section 4.7 (fig. 7-c).

4.1. Construction des générateurs G_i

4.1.1. La zone d'entrée

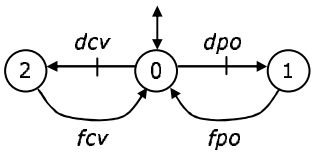
Le modèle de la figure 10 représente le comportement de la butée *Be* qui dépend d'informations provenant du capteur *Ce*. Dans le mode de fonctionnement nominal, qui est le seul étudié, la butée ne peut commencer à sortir (*dsbe*) qu'en présence d'une palette en *Ce* (*ape*). Le modèle a donc été construit dans ce sens.



ape : arrivée d'une palette devant *Ce*
dpe : départ d'une palette de *Ce*
dsbe : demande de sortie de *Be*
fsbe : fin de sortie de la butée *Be*
drbe : demande de rentrée de *Be*
frbe : fin de rentrée de la butée *Be*

figure 10. *$G_{Be\text{-}Ce}$: modèle de l'ensemble butée Be - capteur Ce*

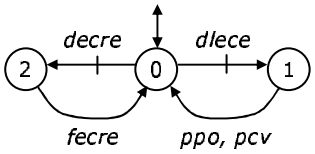
Dans ce modèle, comme ceux de la butée *Bt* (figure 13) et de l'indexage (figure 14), nous avons considéré que la seule suite d'événements à considérer dans le mode nominal était *dsbe*, *fsbe*, *drbe* puis *frbe*. En effet les événements de défaillance ou les interruptions de mouvement n'ont pas été considérés, car non utilisés dans ce mode de fonctionnement. La modélisation explicite de ces événements et trajectoires aurait alourdi la modélisation sans pour autant permettre d'autres comportements du procédé ou améliorer la connaissance du système.



dcv : demande d'aiguillage vers le convoyeur central
fcv : fin d'aiguillage vers le convoyeur central
dpo : demande d'aiguillage vers la zone de travail
fpo : fin d'aiguillage vers la zone de travail

figure 11. *G_{Ag} : modèle de l'aiguillage Ag*

La figure 11 représente les deux choix possibles d'aiguillage : sur le convoyeur central (vers la zone de sortie) ou sur le convoyeur de dérivation (vers la zone de travail).



decre : demande d'écriture sur l'étiquette
fecre : fin de demande d'écriture
dlece : demande de lecture sur l'étiquette
ppo : pièce pour le poste
pcv : pièce pour le convoyeur central

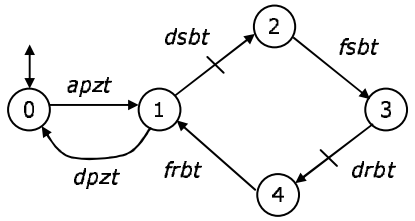
figure 12. *G_{Lce} : modèle de la borne magnétique Lce*

Le modèle de la figure 12 décrit les deux actions possibles de la borne magnétique *Lce* qui sont la lecture et l'écriture. Pour la commande du poste de travail, les seules informations qui nous intéressent, suite à ces actions, sont de savoir si la pièce portée par la palette est destinée au poste (*ppo*) ou non (*pcv*) et que l'écriture est terminée (*fecre*).

L'alphabet local de la zone d'entrée comporte tous les événements générés par *Be – Ce*, *Ag* et *Lce* : Σ_E = {*ape, dpe, dsbe, fsbe, drbe, frbe, dcv, fcv, dpo, fpo, decre, fecre, dlece, ppo, pcv*}.

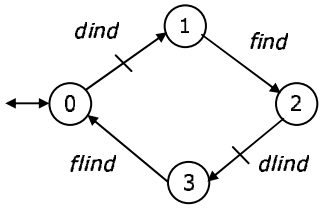
4.1.2. La zone de travail

Le modèle suivant est similaire à celui de la figure 10 car il décrit un ensemble identique et ayant le même comportement. Ici aussi après l'arrivée d'une palette, celle-ci peut partir à moins que la butée ne soit sortie. L'indexage (figure 14) peut être assimilé à un vérin et nous retrouvons donc un modèle identique.



apzt : arrivée d'une palette devant *Cp*
dpzt : départ d'une palette de *Cp*
dsbt : demande de sortie de *Bt*
fsbt : fin de sortie de la butée *Bt*
drbt : demande de rentrée de *Bt*
frbt : fin de rentrée de la butée *Bt*

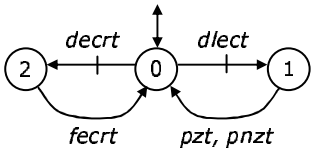
figure 13. $G_{Bt\text{-}Cp}$ *: modèle de l'ensemble butée Bt - capteur Cp*



dind : demande d'indexage
find : fin d'indexage
dlind : demande de libération
flind : fin de libération

figure 14. G_{Ind} *: modèle de l'indexage Ind*

La borne magnétique *Lct* est identique à *Lce*. Son modèle, figure suivante, est identique à celui de la figure 12.



decrt : demande d'écriture sur l'étiquette
fecrt : fin de demande d'écriture
dlect : demande de lecture sur l'étiquette
pzt : pièce à exploiter dans la zone de travail
pnzt : pièce à orienter vers la zone de sortie

figure 15. G_{Lct} *: modèle de la borne magnétique Lct*

L'alphabet local de la zone de travail comporte tous les événements générés par *Bt – Cp*, *Ind* et *Lct* : Σ_T = {*apzt, dpzt, dsbt, fsbt, drbt, frbt, dind, find, dlind, flind, decrt, fecrt, dlect, pzt, pnzt*}.

4.1.3. La zone de sortie

Le modèle de la figure 16 est identique à celui de la figure 13.

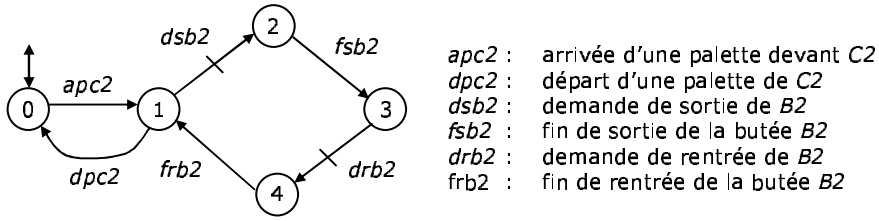


figure 16. *$G_{B2\text{-}C2}$: modèle de l'ensemble butée B2 - capteur C2*

La position initiale de la butée *B3* est différente de celle de la butée *B2*, c'est pourquoi le modèle de la figure 17 est différent du précédent. En effet en position initiale la butée *B2* est rentrée alors que la butée *B3* est sortie. Cela permet, si ces butées ne sont pas commandées, le passage des palettes sur le convoyeur central, tout en évitant les collisions. Par contre le cahier des charges impose que toutes les butées soient rentrées par défaut lorsqu'elles sont commandées.

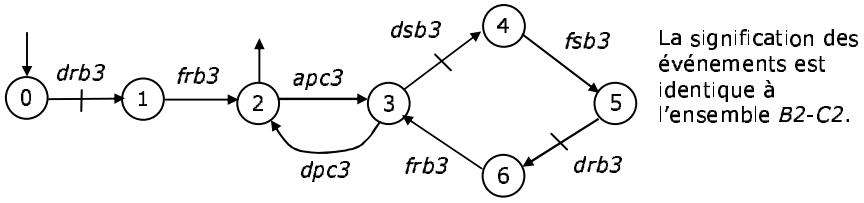


figure 17. *$G_{B3\text{-}C3}$: modèle de l'ensemble butée B3 - capteur C3*

Le modèle de la figure 18 exprime le fait que le détecteur d'entrée du poste suivant ne peut physiquement détecter une pièce que si celle-ci a été libérée par la butée *B2* ou la butée *B3*.

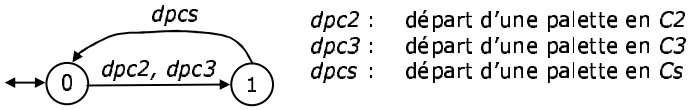


figure 18. *G_{Cs} : modèle du capteur d'entrée du poste suivant Cs*

L'alphabet local de la zone de sortie comporte tous les événements générés par *B2 – C2*, *B3 – C3* et *Cs* : Σ_S = {*apc2, dpc2, dsb2, fsb2, drb2, frb2, apc3, dpc3, dsb3, fsb3, drb3, frb3, dpcs*}.

4.2. Construction des spécifications intra zones structurelles

Deux types de spécifications peuvent contraindre le comportement matériel des procédés (Brandin *et al.*, 1994) (Zaytoon *et al.*, 1999) : les spécifications de sécurité et les spécifications de vivacité. Les spécifications de sécurité représentent ce que le système ne doit pas faire, ce qui consiste à interdire des successions dangereuses d'événements pour le système. Les spécifications de vivacité modélisent ce que le système doit faire pour réaliser sa tâche, ce qui revient à n'autoriser qu'un certain nombre d'événements dans un état donné. Dans l'approche que nous utilisons, la modélisation par des automates définissant les trajectoires possibles, et donc les trajectoires interdites, doit pouvoir intégrer au mieux ces deux types de spécifications.

Les spécifications vont être décrites par des automates sur des alphabets locaux. Ces alphabets permettent de définir les frontières de zones structurelles. Afin de simplifier les modèles, nous avons choisi de décomposer la zone fonctionnelle de sortie (voir figure 1) en deux zones structurelles distinctes qui correspondent aux deux ensembles butée – capteur associé. La figure suivante représente donc les différentes zones structurelles retenues (les rectangles) dont les modèles sont donnés dans cette section, ainsi que les interactions entre ces zones (ellipses) dont les modèles seront donnés en section 4.4.

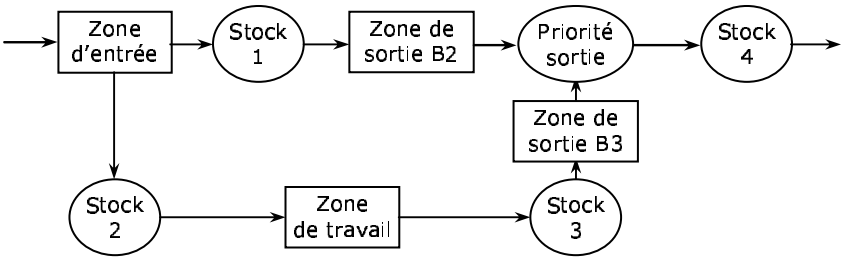
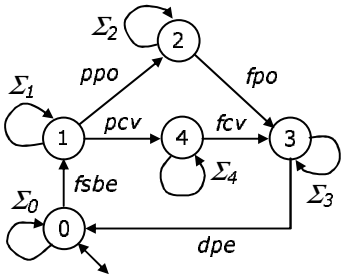


figure 19. *Zones structurelles et interactions*

Pour simplifier la lecture des modèles, l'écriture des boucles sur les états donne la liste (entre guillemets) des événements qui ne permettent **pas** de rester dans cet état. Les événements en gras correspondent aux événements interdits dans cet état. Les autres événements correspondent aux événements qui permettent de changer d'état. Pour chacune des spécifications, nous donnerons tout d'abord une interprétation textuelle puis son modèle formel. Pour chaque zone structurelle, les modèles sont construits sur l'alphabet local de la zone correspondante.

4.2.1. La zone d'entrée

La seule action autorisée au démarrage est la sortie de la butée (*dsbe*). Lorsque la butée a fini de sortir (*fsbe*), la borne doit lire l'étiquette (*dlece*). En fonction du résultat (*ppo* ou *pcv*), l'aiguillage est positionné vers le convoyeur central (*dcv*) ou vers le poste de travail (*dpo*). Ensuite la butée doit rentrer (*drbe*) pour laisser passer la palette (*dpe*) et revenir dans l'état initial.

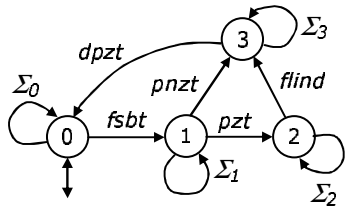


Σ_E = {*ape, dpe, dsbe, fsbe, drbe, frbe, dcv, fcv, dpo, fpo, decre, fecre, dlece, ppo, pcv*}
Σ_0 = Σ_E - {*fsbe*, ***dpo, dcv, dlece, decre*** }
Σ_1 = Σ_E - {*ppo, pcv*, ***drbe, dpo, dcv, decre***}
Σ_2 = Σ_E - {*fpo*, ***drbe, dcv, dlece, decre***}
Σ_3 = Σ_E - {*dpe*, ***dsbe, dpo, dcv, dlece, decre***}
Σ_4 = Σ_E - {*fcv*, ***drbe, dpo, dlece, decre***}

figure 20. *$E_{E,loc}$: spécification de la zone d'entrée*

4.2.2. La zone de travail

Lorsqu'une palette est détectée par le capteur de présence palette *Cp* (*apzt*), la butée sort (*dsbt*) afin de bloquer la palette et une demande de lecture (*dlect*) indique si la palette doit être indexée (*dind* si *pzt*) ou non (*pznt*). En fin de traitement (*flind*) la butée rentre (*drbt*), autorisant ainsi le départ de la palette (*dpzt*).

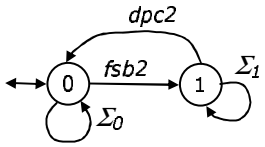


Σ_T = {*apzt, dpzt, dsbt, fsbt, drbt, frbt, dind, find, dlind, flind, decrt, fecrt, dlect, pzt, pznt*}
Σ_0 = Σ_T - {*fsbt*, ***dind, dlect, decrt***}
Σ_1 = Σ_T - {*pnzt, pzt*, ***dind, drbt, decrt***}
Σ_2 = Σ_T - {*flind*, ***drbt, dlect, decrt***}
Σ_3 = Σ_T - {*dpzt*, ***dind, dsbt, dlect, decrt***}

figure 21. *$E_{T,loc}$: spécification de la zone de travail*

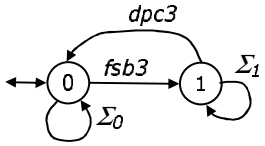
4.2.3. La zone de sortie

D'après le procédé, une butée ne peut sortir (*fsb2* ou *fsb3*) qu'après l'arrivée d'une palette (*apc2* ou *apc3*). Par contre chaque butée ne pourra ressortir qu'après le départ de la palette concernée (*dpc2* ou *dpc3*) et donc l'arrivée d'une nouvelle palette.



Σ_{B2} = {*apc2, dpc2, dsb2, fsb2, drb2, frb2*}
Σ_0 = Σ_{B2} - {*fsb2*}
Σ_1 = Σ_{B2} - {*dpc2*, ***dsb2***}

figure 22. *$E_{B2,loc}$: spécification de la zone de sortie - butée B2*



Σ_{B3} = {*apc3, dpc3, dsb3, fsb3, drb3, frb3*}
Σ_0 = Σ_{B3} - {*fsb3*}
Σ_1 = Σ_{B3} - {*dpc3*, ***dsb3***}

figure 23. *$E_{B3,loc}$: spécification de la zone de sortie - butée B3*

4.3. Synthèse des trajectoires valides

De manière classique les trajectoires locales valides sont obtenues par synthèse d'une spécification et de son procédé local. Il faut donc d'abord déterminer ces procédés locaux.

L'automate G modélisant le procédé global résulte de la composition parallèle des automates de chaque composant G_j. Chaque automate d'un procédé local G_i est ensuite déterminé par la projection sur son alphabet local Σ_i de G :

$$G_i = P_i(G) = P_i(\| G_j)$$

Les procédés locaux sont donc déterminés par rapport à l'alphabet de chaque spécification. Or, dans notre cas, les alphabets locaux Σ_E, Σ_T, Σ_{B2} et Σ_{B3} sont disjoints donc les procédés locaux peuvent être déterminés directement à partir des automates des composants de ces zones :

$$G_E = \| (G_{Be\text{-}Ce}, G_{Ag}, G_{Lce})$$

$$G_T = \| (G_{Bt\text{-}Cp}, G_{Ind}, G_{Lct})$$

$$G_{B2} = P_{B2} (G_{B2\text{-}C2} \| G_{Cs}) = G_{B2\text{-}C2}$$

$$G_{B3} = P_{B3} (G_{B3\text{-}C3} \| G_{Cs}) = G_{B3\text{-}C3}$$

Les quatre spécifications sont contrôlables. Les procédés sous contrôle sont obtenus par :

$$S_{E,loc}/G_E = G_E \times E_{E,loc} \qquad S_{T,loc}/G_T = G_T \times E_{T,loc}$$

$$S_{B2,loc}/G_{B2} = G_{B2} \times E_{B2,loc} \qquad S_{B3,loc}/G_{B3} = G_{B3} \times E_{B3,loc}$$

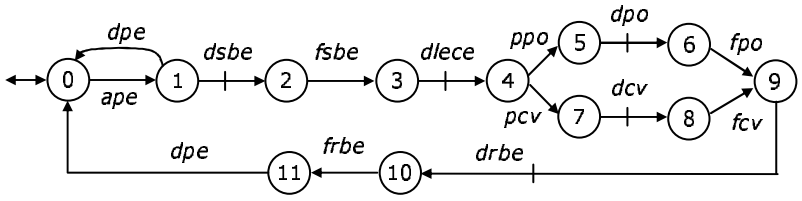


figure 24. $S_{E,loc}/G_E$ *: Trajectoires de commande de la zone d'entrée*

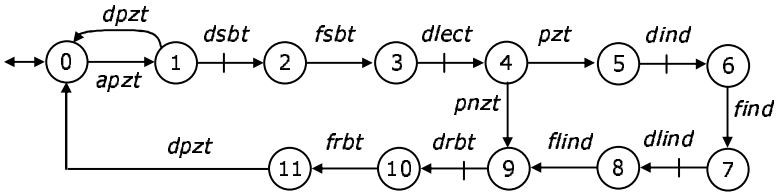


figure 25. $S_{T,loc}/G_T$ *: Trajectoires de commande de la zone de travail*

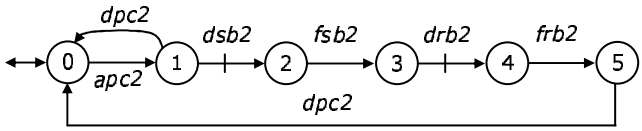


figure 26. $S_{B2,loc}/G_{B2}$ *: Trajectoires de commande de la zone de sortie – B2 et C2*

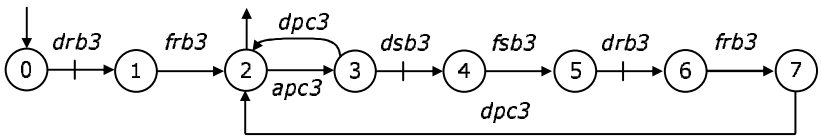


figure 27. $S_{B3,loc}/G_{B3}$ *: Trajectoires de commande de la zone de sortie – B3 et C3*

Ces trajectoires sont contrôlables. Les spécifications sont non conflictuelles car construites sur des alphabets disjoints. Le système est donc non bloquant.

4.4 Interactions entre zones structurelles

Les spécifications construites précédemment portaient sur des alphabets locaux permettant de définir des zones structurelles disjointes. Les stocks se trouvent entre plusieurs zones structurelles et permettent de définir leurs interactions. Pour les

stocks 1, 2 et 3, les événements permettant d'augmenter le nombre de pièces dans le stock font partie de l'alphabet de la zone amont et ceux diminuant ce nombre de celui de la zone aval. Le stock 4 est particulier puisque deux zones (B2 et B3) peuvent alimenter ce stock. Pour ce dernier nous aurons donc deux spécifications : une gérant la taille du stock et l'autre gérant une priorité entre les deux zones B2 et B3.

4.4.1. Stock 1

Le stock 1 est alimenté par les pièces provenant de la zone d'entrée ayant été dirigées vers la zone de sortie. La spécification de ce stock sera construite sur l'union des alphabets de la zone d'entrée et de la zone B2 : $\Sigma_{ST1} = \Sigma_E \cup \Sigma_{B2}$.

Pour qu'il y ait une pièce en plus dans ce stock il faut qu'il y ait eu succession des deux événements *fcv* et *dpe*. Pour retirer une pièce du stock, il suffit de vérifier l'événement *dpc2*. Les états pairs permettent donc de connaître le nombre de pièces en stock, quant aux états impairs ils indiquent qu'un transfert est en cours. Sur les états les boucles sont les suivantes : $\Sigma_0 = \Sigma_{ST1}$ - {*fcv*}, $\Sigma_1 = \Sigma_{ST1}$ - {*dpe*}, $\Sigma_2 = \Sigma_{ST1}$ - {*fcv*, *dpc2*}, $\Sigma_{10} = \Sigma_{ST1}$ - {*dpc2*,***dcv***}. Le seul événement interdit est donc l'aiguillage des pièces vers ce stock (*dcv*) lorsqu'il contient déjà cinq pièces.

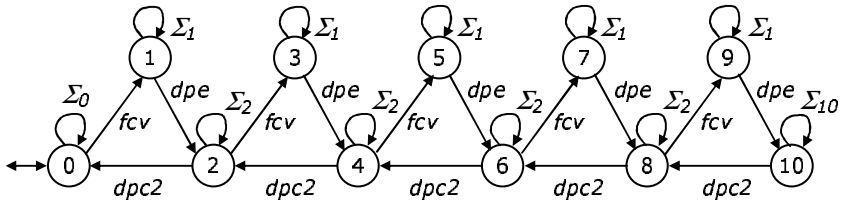


figure 28. *Spécification du stock 1*

4.4.2. Stock 2

Le principe est identique. Cette fois $\Sigma_{ST2} = \Sigma_E \cup \Sigma_T$. Les deux événements qui indiquent l'arrivée d'une pièce dans le stock sont *fpo* et *dpe*,.alors que *dpzt* indique le retrait d'une pièce. Les boucles sont $\Sigma_0 = \Sigma_{ST2}$ - {*fpo*}, $\Sigma_1 = \Sigma_{ST2}$ - {*dpe*}, $\Sigma_2 = \Sigma_{ST2}$ - {*fpo*, *dpzt*} et $\Sigma_{10} = \Sigma_{ST2}$ - {*dpzt*, ***dpo***}. L'événement interdit est l'aiguillage des pièces vers ce stock (*dpo*) lorsqu'il contient cinq pièces.

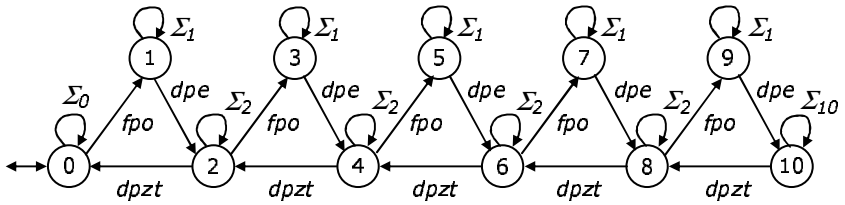


figure 29. *Spécification du stock 2*

4.4.3. Stock 3

Pour le stock 3, le principe est à nouveau le même mais cette fois-ci un seul événement *dpzt* permet d'indiquer l'ajout d'une pièce dans le stock. L'événement *dpc3* correspond au retrait d'une pièce. Les boucles sont $\Sigma_0 = \Sigma_{ST3}$ - {*dpzt*}, $\Sigma_1 = \Sigma_{ST3}$ - {*dpzt*, *dpc3*} et $\Sigma_5 = \Sigma_{ST3}$ - {*dpc3*, ***drbt***} avec $\Sigma_{ST3} = \Sigma_T \cup \Sigma_{B3}$.

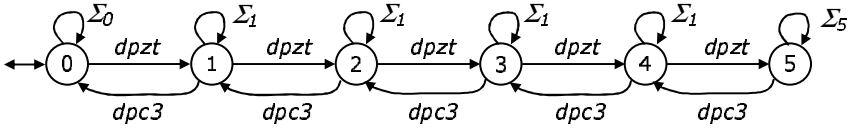
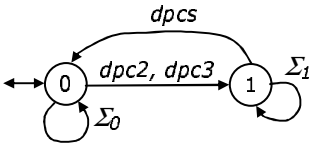


figure 30. *Spécification du stock 3*

4.4.4. Stock 4

Pour le stock 4, l'alphabet utilisé est celui de la zone fonctionnelle de sortie, Σ_S. Une seule pièce doit être comprise entre les butées *B2 – B3* et le capteur d'entrée de la zone suivante *Cs*. Ceci est modélisé par une spécification de stock ne comportant qu'une pièce. Ce stock est alimenté suite aux événements *dpc2* et *dpc3*, il est vidé suite à l'événement *dpcs*. A partir du moment où le stock est plein il faut empêcher les butées de rentrer (*drb2* et *drb3*).



$\Sigma_0 = \Sigma_S$ - {*dpc2, dpc3*}
$\Sigma_1 = \Sigma_S$ - {*dpcs*, ***drb2, drb3***}

figure 31. *Spécification de stock 4*

4.4.5. Priorité de la zone de sortie

Lorsqu'une palette arrive en *C2* ou *C3*, elle ne passe pas toujours immédiatement. Dans ce cas nous avons choisi de libérer les palettes dans leur ordre d'arrivée. Pour cela il faut garder un historique de cet ordre. Sur l'automate

suivant, les états 1 et 2 correspondent à l'arrivée en premier de la palette en *C2*. Dans ce cas la rentrée de la butée *B3* (*drb3*) est interdite tant que la palette en *C2* n'est pas partie (*dpc2*). Les états 3 et 4 correspondent au cas contraire. Cette fois c'est la rentrée de la butée *B2* (*drb2*) qui est interdite tant que la palette en *C3* n'est pas partie (*dpc3*). L'alphabet utilisé pour cette spécification est aussi Σ_S.

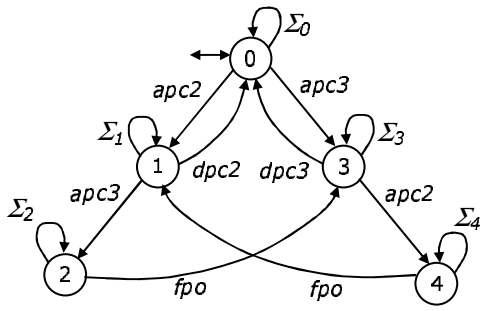


figure 32. *Spécification de priorité de la zone de sortie*

4.4.6. Propriétés des spécifications de stock

Les quatre spécifications de stock, ainsi que celle de priorité, n'interdisent que des événements contrôlables. Ces spécifications sont donc toutes contrôlables. Nous avons par ailleurs vérifié qu'elles sont non conflictuelles entre elles et avec les spécifications intra zones énoncées précédemment.

4.5. Comparaison avec l'approche centralisée

Dans l'approche centralisée, le modèle du procédé est obtenu par la composition parallèle de tous les automates composant le procédé. Cet automate G comporte 189 000 états et 2 029 500 transitions. La construction du modèle des spécifications nécessite tout d'abord d'étendre le modèle de chaque spécification à l'alphabet global Σ. Ceci revient à rajouter en boucle sur chaque état de S_i les événements appartenant à $\Sigma - \Sigma_i$. Le modèle global est ensuite obtenu par produit de tous les modèles de spécifications. Il comporte 541 200 états et 17 222 300 transitions. Quant au modèle des trajectoires valides, il est obtenu à partir des deux modèles précédents. Ce modèle n'a pu être déterminé car son calcul nécessite une capacité mémoire supérieure aux capacités du logiciel TCT[4] que nous avons utilisé pour l'ensemble de cette étude.

[4] TCT est un logiciel permettant de mettre en œuvre la théorie RW à partir d'automates à états. Il est téléchargeable sur le site de Wonham. Voir (Wonham, 2002) pour l'adresse Web.

Dans notre approche, les modèles G_E, G_T, G_{B2} et G_{B3} des procédés locaux comportent respectivement 45, 60, 5 et 7 états et 189, 232, 6 et 8 transitions. Les modèles de spécifications sont aussi de taille réduite : le plus important comporte 11 états et 329 transitions (modèle du stock 2, figure 29 page 338). Le calcul des trajectoires valides se fait facilement puisque le plus gros modèle obtenu, celui de la zone d'entrée, comporte 12 états et 14 transitions (modèle de $S_{E,loc}/G_E$, figure 24 p. 336).

L'approche centralisée n'est donc pas utilisable sur cet exemple, alors que l'approche proposée nous a permis de travailler sur des modèles de taille plus en rapport avec la difficulté du problème posé. Ces modèles sont de taille comparable avec ceux qui auraient été obtenus par d'autres approches utilisées pour l'étude de tels systèmes : RdP, grafcet…

4.6. Choix de trajectoires

En appliquant les règles énoncées dans la section 3.3.2. peu de trajectoires sont supprimées pour chaque zone. Les modèles obtenus respectent bien évidemment les spécifications, et pourront donc être traduits en instructions d'automates programmables.

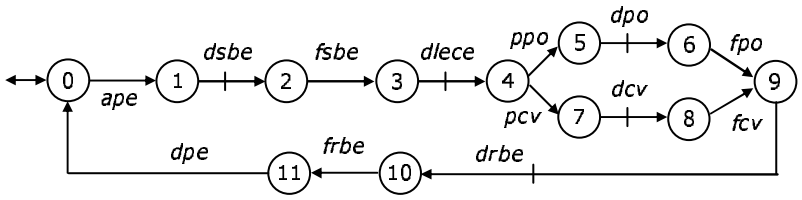


figure 33. *Trajectoires de commande choisies de la zone d'entrée*

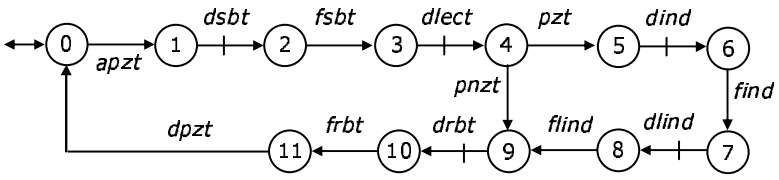


figure 34. *Trajectoires de commande choisies de la zone de travail*

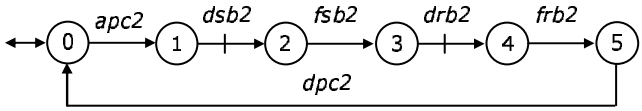


figure 35. *Trajectoires de commande choisies de la zone de sortie - B1 et C1*

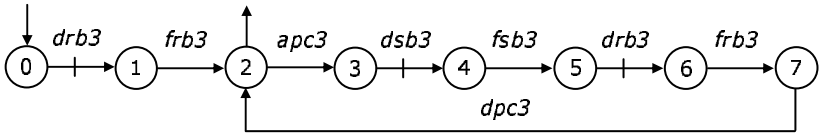


figure 36. *Trajectoires de commande choisies de la zone de sortie - B2 et C2*

Sur les spécifications inter zones, stocks et priorité, seules les boucles sont supprimées. Les modèles ne sont donc pas reproduits ici.

4.7. Implantation de la commande

La commande opérationnelle de chaque poste de travail est assurée par un automate unique de type Premium de la marque Schneider. Chaque automate de poste communique, par l'intermédiaire de bus Uni-Telway, avec les deux stations XGS de lecture/écriture utilisées sur le poste. Il communique également avec un terminal de dialogue XBT Magélis. Le logiciel PL7-pro permet d'écrire les programmes de ces automates en utilisant des langages basés sur la norme IEC 61131-3.

Nous cherchons à obtenir des programmes ayant une structure facilitant la compréhension et l'analyse. Nous utilisons donc le Sequential Function Chart (SFC) pour la structure séquentielle du programme et le langage à contacts (LD) pour les réceptivités et les traitements. Les graphes SFC et les réceptivités font partie du traitement MAST, alors que les traitements sont regroupés dans le traitement POST. Le langage littéral structuré (ST) est utilisé pour les sous-routines de gestion des stations XGS (dans le traitement MAST) ainsi que pour la programmation de boîtes fonctionnelles de comptage-décomptage. Les parties programmées en ST ne seront pas traitées ici.

La traduction des trajectoires locales nécessite tout d'abord la traduction des événements. Les événements incontrôlables sont traduits sous la forme d'expressions combinatoires d'entrées ou de variables internes. Les événements contrôlables correspondent à des actions simultanées sur des sorties ou sur des variables internes. Ces traductions correspondent à des préoccupations logiques ou

technologiques. Par exemple, sur la figure 37 représentant la traduction des trajectoires de la zone de travail, la variable %M42 correspond à la fin d'une temporisation introduite pour que la butée de travail ait effectivement fini de sortir. D'autres temporisations sont introduites pour que les mouvements des palettes se terminent correctement ou que les opérations de lecture et écriture se fassent.

La traduction nécessite également de considérer des règles de traduction des structures des automates sous forme de structures de SFC. Ainsi, l'alternance événement incontrôlable – événement contrôlable se traduit assez naturellement sous forme d'une alternance (transition, réceptivité) – (étape, action). De même le choix de deux trajectoires commençant par un événement incontrôlable s'interprète sous la forme d'un choix de séquence. Par contre nous avons traduit la succession de deux événements incontrôlables par une succession transition - étape – transition afin de bien distinguer les deux occurrences d'événements correspondants.

La figure 38 donne un exemple de la programmation de la gestion d'un stock. Elle représente les entrées et sorties de la DFB gérant le stock 3 (figure 5). Les fronts descendant des entrées *%I3.1* et *%I3.3* correspondent respectivement aux événements *dpzt* et *dpc3* (figure 34 et figure 36). La variable %M53 indique que le stock 3 est plein.

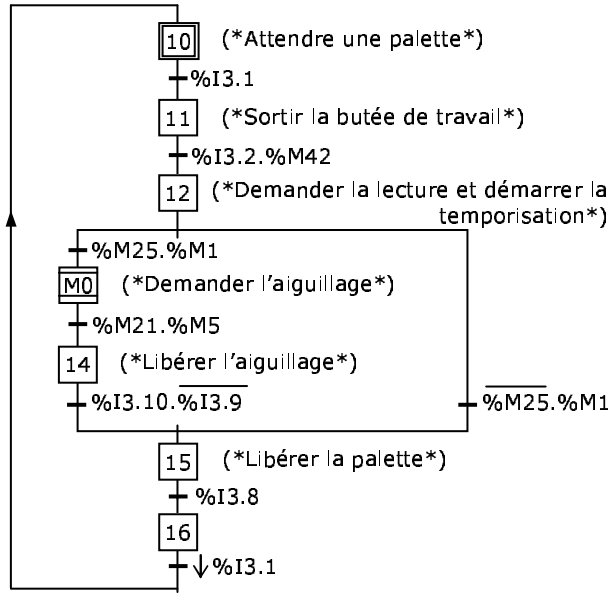


figure 37. *SFC de la zone de travail*[5]

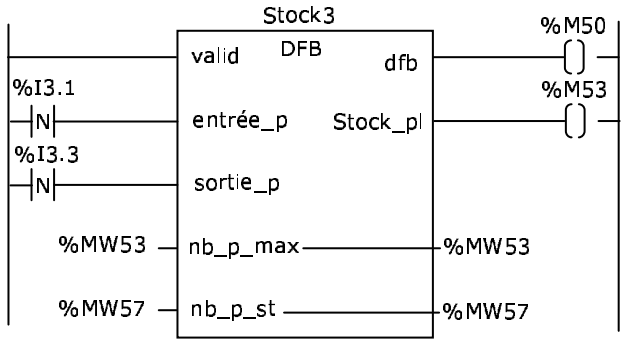


figure 38. *DFB de gestion du stock 3*

A part les variables internes représentant les fins de temporisation, le POST, dont une partie est donnée figure suivante, ne comprend que des actions

[5] Les réceptivités ont été incluses sur la figure dans un souci de lisibilité : elles sont en réalité programmées sous forme de réseaux LD. Les parenthèses associées aux étapes sont des commentaires.

maintenues. Ceci est dû aux choix technologiques sur les vérins et les distributeurs. La vérification de l'état du stock, par l'intermédiaire de *%M53*, est directement rajoutée dans les conditions autorisant la libération d'une palette dans la zone de travail. Ceci correspond à l'interdiction de l'événement *drbt* dans la spécification du stock 3.

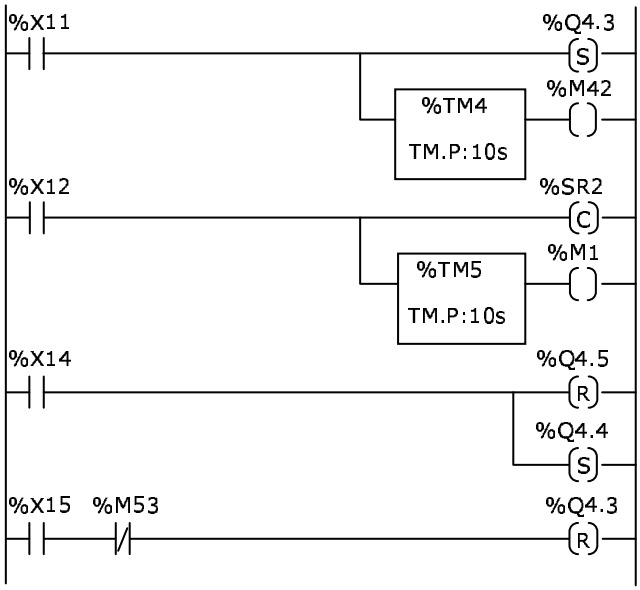


figure 39. *Traitements POST de la zone de travail*

5. Conclusion

La théorie du contrôle par supervision est une théorie intéressante car elle permet la séparation entre les modèles du procédé et ceux des spécifications. Elle permet en outre de définir pour les SED des propriétés telles que la contrôlabilité, le blocage, etc. Cependant l'approche classique d'utilisation de cette théorie est très rapidement confrontée au problème d'explosion combinatoire de la taille des modèles. En outre les hypothèses de base de cette théorie ne correspondent pas avec celles couramment admises pour les SAP, ce qui rend l'utilisation des modèles difficile.

Nous avons présenté dans cet article une démarche d'utilisation de la théorie RW. Elle repose sur le découpage du système en zones structurelles distinctes. Ceci revient à partager l'alphabet global en alphabets locaux disjoints. Nous utilisons l'approche décentralisée pour déterminer les procédés locaux sous contrôle et pour

vérifier les propriétés de non-blocage et d'optimalité de l'ensemble des modèles. Par contre nous nous écartons de l'utilisation courante de l'approche décentralisée en considérant des spécifications d'interactions permettant de restreindre les comportements possibles des procédés sous contrôle des différentes zones. Notre apport théorique a aussi consisté à proposer des règles de choix de trajectoires basées sur les contraintes comportementales de la partie commande. Les trajectoires obtenues peuvent ainsi être traduites en programmes d'automates programmables.

A travers l'exemple présenté dans cet article nous avons montré une illustration détaillée et complète de l'application de la théorie RW. Les modèles obtenus nous montrent que des démarches d'utilisation de la théorie RW permettent d'étudier des problèmes de taille industrielle sans être confronté à une explosion combinatoire. De plus ces résultats ont été obtenus sans pour autant faire de concession sur la rigueur de la démarche ni sur la preuve de propriétés. Nous avons ainsi témoigné de la nécessité de travailler aussi bien sur les méthodes que sur les concepts pour développer cette théorie.

A travers cet exemple détaillé nous espérons avoir montré que, contrairement à une idée reçue, celle-ci est applicable sur des problèmes de taille importante. Nous pensons aussi que cet exemple peut servir de base de comparaison avec d'autres approches de conception formelle de la commande de SED et notamment d'autres approches de synthèse. Une autre perspective, que nous n'avons pas abordé, est le développement de logiciels facilitant la construction et l'exploitation des modèles par des non spécialistes, permettant ainsi le transfert de cette théorie vers l'enseignement et le milieu industriel.

Nous tenons à remercier Luc Regimbal, dont le travail au LAI et les réflexions ont servi de base à cet article.

Bibliographie

Altisen K., Gössler G., Sifakis J., « Scheduler modeling based on the Controller synthesis Paradigm », *Real-Time Systems*, vol. 23, 2002, p. 55-84.

Balemi S., Hoffmann G. J., Gyugyi P., Wong-Toi H., Franklin G. F., « Supervisory control of a rapid thermal multiprocessor », *IEEE Transactions on Automatic Control*, vol. 38, n°7, 1993, p. 1040-1059.

Brandin B. A., Wonham W. M., « Supervisory control of timed discrete-event systems », *IEEE Transactions on Automatic Control*, vol. 39, n°2, février 1994, p. 329-342.

Cassandras C. G., Lafortune S., *Intoduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999, 822 p.

Chafik S., Niel E., « Hierachical-decentralized solutions of supervisory control », *3rd Mathmod*, Vienne, Autriche, vol. 2, 2000, p. 787-790.

Chafik S., Niel E., « Du contrôle par Supervision Hierarchique et Distribuée », *MSR2001 : modélisation des systèmes réactifs*, Toulouse, France, 17-19 Octobre 2001, p. 55-70.

Charbonnier F., Commande supervisée des systèmes à événements discrets, Thèse de doctorat, Institut National Polytechnique de Grenoble, 1996.

De Queiroz M. H., Cury J. E. R., « Synthesis and implantation of local modular supervisory control for a manufacturing cell », *6th international workshop on discrete event systems (WODES'02)*, Zaragoza, Espagne, 2-4 octobre 2002, p. 377-382.

Dietrich P., Malik R., Wonham W. M., Brandin B. A., « Implementation considerations in supervisory control », *proceedings of SCODES'01*, Paris, France, juillet 2001, p. 27-38.

Hellgren A., Lennartson B., Fabian M., « Modelling and PLC-based implementation of modular supervisory control », *6th international workshop on discrete event systems, WODES'02*, Zaragoza, Espagne, 2-4 octobre 2002, p. 371-376.

Khatab A., Contrôle et contrôle stabilisant des systèmes à événements discrets : application au recouvrement des défaillances, Thèse de doctorat, INSA de Lyon, 2000.

Kumar R., Supervisory synthesis technics for discrete event dynamics systems, Thèse de doctorat, Université du Texas, 1991, 198 p.

Kumar R., Garg V. K. Et Marcus S. I., « Predicates and predicate transformers for supervisory control of discrete event dynamical systems », *IEEE transactions on Automatic Control*, vol. 38, n°2, 1993, p. 232-247.

Leduc R. J., PLC implementation of a DES supervisory for a manufacturing testbed : an implementation perspective, Ma. Sc. Thesis, Dept. of Elec. & Comp. Engrg., Univ. of Toronto, 1996.

Lhoste P., Faure J.-M., Lesage J.-J., Zaytoon J., « Comportement temporel du grafcet », *Journal Européen des Systèmes Automatisés*, vol. 31, n°4, 1997, p. 685-711.

Lin F., Wonham W. M., « Decentralized Supervisory Control of Discrete-Event Systems », *Inform.Sci.*, vol.. 44, n°3, 1988, p. 199-244.

Lin F., Wonham W. M., « Decentralized Control and Coordination of Discrete-Event Systems with Partial Observation », *IEEE Transactions on Automatic Control*, vol.35, n°12, 1990, p.1330-1337.

Liu J., Darabi H., « Ladder logic implementation of Ramadge-Wonham supervisory controller », *6th international workshop on discrete event systems, WODES'02*, Zaragoza, Espagne, 2-4 octobre 2002, p. 383-389.

Marchand H., Gaudin B., « Supervisory Control Problems of Hierarchical Finite State Machines », *41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, décembre 2002, p. 1199-1204.

Mohanty S. R., Chandra V., Kumar R., « A framework for optimal control of discrete event systems », *IEEE Transactions on Robotics and Automation*, soumis en 2002, 24 p. (http://clue.eng.iastate.edu/~rkumar/jour.html#optimal)

Piétrac L., Chafik S., Regimbal L., « Application de la théorie de la supervision : un exemple de conception de programmes d'API », *Conférence Internationale Francophone d'Automatique*, *CIFA '2002*, Nantes, France, 8-10 juillet 2002, 6 p.

Ramadge P. J., Wonham W. M., « Supervisory control of a class of discrete event processes », *SIAM journal on Control and Optimization*, vol. 25, n°1, 1987, p. 206-230.

Ramadge P. J., Wonham W. M., « The control of discrete event systems », *IEEE Transactions on Automatic Control*, vol. 77, n°1, 1988, p. 81-98.

Regimbal L., Application de la théorie de contrôle par supervision : proposition d'une approche d'implantation décentralisée, Mémoire d'ingénieur CNAM, 2002.

Sengupta R., Lafortune S., « A graph-theortic optimal control problem for terminating discrete event processes », *Discrete Event Dynamic System: Theory and Applications*, vol. 2, n°2, 1992, p. 139-172.

Wonham W. M., Notes on control of discrete-event systems, notes de cours, Depertment of Electrical and Computer Engineering, University of Toronto, http://www.control.toronto.edu/people/profs/wonham/wonham.html, juillet 2002.

Zaytoon J., Carré-Ménétier V., « Grafcet et graphe d'états : comportement, raffinement, vérification et validation », *Journal Européen des Systèmes Automatisés*, vol. 33, n°7, 1999, p. 751-782.

Zong H., Wonham W. M., « On the consistency of hierarchical supervision in discrete-event systems », *IEEE Transactions on Automatic Control*, vol. 35, n°10, octobre 1990, p. 1124-1134.