MODELISATION DE CONTROLEURS DISCRETS POUR L'INDUSTRIE 4.0

Armand Toguyeni

13ème Conférence Francophone de Modélisation, Optimisation et Simulation- MOSIM'20 – 12 au 14 novembre 2020 - Agadir – Maroc «Nouvelles avancées et défis pour des industries durables et avisées»

MODELISATION DE CONTROLEURS DISCRETS POUR L'INDUSTRIE 4.0

Armand TOGUYENI

Centrale Lille Institut, CRIStAL, UMR 9189 F-59650 Villeneuve-d'Ascq, France
Univ. Lille Nord de France, F-59650, Lille, France
armand.toguyeni@centralelille.fr

RESUME : *Ce papier présente une étude relative au développement de systèmes de production intelligents dans le contexte de l'industrie manufacturière. Nous montrons que de tels systèmes doivent être agile, c'est-à-dire capable de s'adapter rapidement à des contextes changeant comme une production particulière ou une panne. Dans cette étude nous modélisons le système de commande de ces systèmes par une approche en couches et générique. Nous distinguons plusieurs classes de modèles permettant de gérer chacun différentes catégories de flexibilité. L'objectif essentiel de l'étude est de fournir une méthode rigoureuse et systématique permettant de modéliser et de vérifier de manière efficace ce type de système.*

MOTS-CLES : *Modélisation des systèmes, Industrie 4.0, Ingénierie des systèmes, Systèmes à Evènements Discrets, Réseaux de Petri, contrôleur.*

1 INTRODUCTION

L'Industrie 4.0 est un concept clé dans le développement actuel des systèmes de production (Mittal et *al.*, 2019 ; Qu et *al.*, 2019). L'idée de ce concept est la construction de systèmes industriels (production et logistique) qui peuvent adapter leur production en fonction de leur environnement (demandes du marché, défaillances de machines ...). Cette étude concerne la conception de systèmes de production discrets et intelligents. L'agilité de ce type de système dépend de son degré de flexibilité et de ses capacités de reconfiguration (Radziwon et al. 2014).
L'objectif de cette étude est donc la conception du contrôle de tels systèmes afin de profiter des flexibilités matérielles offertes par le système pour adapter en permanence sa production. Pour cela, nous avons choisi de les modéliser par les Colored Petri Nets (CPN) qui offrent la possibilité d'identifier les produits tout en ayant un fort potentiel de concision (Jensen and Kristensen, 2015).
L'objectif de ce travail est de fournir une méthode d'ingénierie systématique pour assurer la conception rigoureuse de modèles de contrôle. Il s'agit d'un complément et d'une amélioration des travaux présentés dans (Toguyeni 2006). Il propose une modélisation plus concise des Gammes Opératoires Etendues (GOE), réduisant ainsi considérablement la combinatoire des modèles en phase de conception. Il propose également la modélisation discrète des allocateurs de ressources, permettant une vérification discrète du modèle global.

Cet article est structuré comme suit. La deuxième partie concerne la problématique et le contexte de l'étude. Dans la troisième partie, nous présenterons notre méthode de modélisation. Dans la quatrième partie nous proposons un modèle générique des GOE. Dans la cinquième partie, nous présentons un modèle d'allocateur de machines permettant de mettre en œuvre à la fois les fonctions de pré-allocation (Toguyeni, 2006) et d'allocation d'une machine. Nous terminerons cette étude par une conclusion et les perspectives de ce travail.

2 PROBLEMATIQUE ET CONTEXTE DE L'ETUDE

Les systèmes de production actuels doivent être agiles afin de pouvoir produire des produits personnalisés en faibles quantités. Cela nécessite de pouvoir fabriquer en temps réel des produits nouveaux et donc d'être facilement reconfigurable. Pour faire face à ces défis, deux concepts clés doivent être pris en compte pour leur conception : la flexibilité et la reconfiguration dynamique (Radziwon et al., 2014).
On distingue plusieurs types de flexibilité : flexibilité du produit, flexibilité de gamme opératoire, flexibilité des fonctions de transformation, flexibilité du transport, flexibilité de la production ... Ces différentes formes de flexibilité vont induire un contrôle plus ou moins complexe. Dans cette étude nous nous intéressons en particulier à la flexibilité de type « open shop ». Du point de vue commande, la problématique majeure d'un open shop est de maitriser l'explosion combinatoire du modèle de comportement. Cette explosion est due au fait que les opérations ne sont pas ordonnées (Toguyeni, 2006). Dans la partie 4, nous proposerons notamment un nouveau modèle de gamme opératoire étendue permettant de traiter ce problème.
La reconfiguration d'un système de production est nécessaire en cas de modification de l'ensemble des ressources du système. Cette modification peut être permanente (cas de l'ajout d'une nouvelle ressource) ou temporaire (cas d'une panne de ressource). Une panne de ressource est un événement imprévu qui doit être traité en ligne, pour gérer les produits qui sont déjà dans le système et pour les produits qui doivent être chargés sur le système. Le concept de reconfiguration a été formalisé il y a quelques années pour définir une nouvelle classe de systèmes de production appelée Système de Production Reconfigurable ou SPR (Koren 2014).

La Figure 1 décrit un système intelligent qui servira de cas d'étude illustratif tout au long de cette étude. Chaque machine met en œuvre différents types d'opérations d'usinage notées "f_i". L'opération "f_3" est ainsi réalisée par toutes les machines. À l'opposé, l'opération "f_4" est uniquement réalisable par la machine M_4. Elle est donc critique au sens qu'elle ne peut pas être remplacée. Les machines de ce système sont desservies par un convoyeur à bande. Sur ce convoyeur sont positionnés des postes de travail (Z_i) permettant aux robots de palettiser ou de dé-palettiser les pièces chargées sur des palettes afin d'être transportées par le convoyeur auprès d'une machine. Les relations d'accessibilité du système de transport sont illustrées sur la Figure 1 par des flèches orientées montrant la direction du mouvement des pièces et des palettes. Pour passer de Z_1 à Z_3, une pièce a deux possibilités : soit être routée par le chemin IS_1-Z_2-IS_2, soit utiliser directement le chemin IS_5. Ces alternatives illustrent la flexibilité du routage. De même, le robot R_4 peut transférer directement une pièce du stock *IN* à Z_2, le poste de chargement de la machine M_1. En fonctionnement normal, à partir d'*IN*, le stock d'entrée, il charge les pièces sur les palettes bloquées sur le poste Z_1. De même, il peut décharger des pièces finies vers le stock de sortie *PiOUT*. Les IS_i (i $\in$ {1,2,3,4,5,6}) sont des zones de stockage temporaire du convoyeur. Cette étude suppose que la capacité de chaque Z_i est d'une palette et que la capacité des zones IS_i est de 5 palettes.

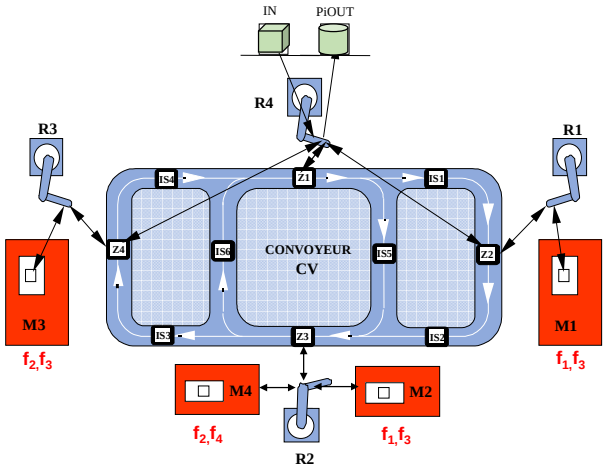

Figure 1 : Exemple de système de production intelligent

3 METHODE DE CONCEPTION DE LA COMMANDE

3.1 Présentation synthétique du processus de conception

Notre processus de conception est basé sur un découplage entre les contraintes des spécifications du produit et les contraintes des spécifications des ressources. Par conséquent, il comprend deux flux de conception : un flux centré sur le produit et un flux centré sur les ressources (Toguyeni 2006).

Cette étude se concentre sur la construction des modèles finaux construits par le flux de produits. Étant donné la distribution de ces modèles sur des ordinateurs séparés communiquant par le biais de réseaux informatiques industriels et de messagerie industrielle, on utilise une approche client/serveur pour prendre en compte la contrainte d'asynchronisme induite par cet environnement (voir (Toguyeni 2006)). Dans nos modèles, cela se traduira par l'utilisation de paires de places sémaphores (Requête/Acknowlege) permettant de synchroniser des processus Réseaux de Petri (RdP) distribués.
Les principaux modèles finaux sont la Gamme Opératoire Etendue (GOE), le Graphe de Coordination du Système de Transport (GCST) et les allocateurs de ressources. Une GOE modélise les différentes opérations appliquées à un produit afin qu'il passe de son état brut à un état fini. Le GCST permet de gérer les opérations de palettisation et de dépalettisation d'une pièce sur une palette circulant sur un tapis roulant, ainsi que les transferts d'une zone de travail à une autre. Les allocateurs de ressources permettent d'affecter les ressources pour réaliser les opérations de transformation des pièces, en fonction de leur gamme de fabrication respective.

3.2 Principes de base de la modélisation

L'ensemble des modèles construits est basé sur notre technique de routage des pièces dans le système de production. Cette technique est inspirée des datagrammes, une technique de routage des paquets dans les réseaux informatiques. Ainsi chaque pièce est routée dans le système de production en fonction de deux paramètres : sa destination finale et l'état des ressources de transformation et de transport.

La destination finale d'une pièce est définie par les allocateurs de ressources de la fonction pilotage. Pour comprendre le principe, nous considérons la **Figure 2**.

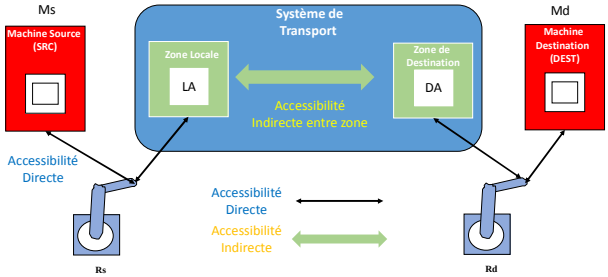

Figure 2 : Schéma générique pour le transport d'une pièce

Le transport d'une pièce d'un lieu source vers un lieu destination est basé sur les relations d'accessibilité (Toguyeni 2006) entre les différentes postes de travail du système de production. Si, nous considérons le système de la Figure 1, Z_2 et M_1 sont en relation d'accessibilité externe car il est nécessaire d'utiliser le robot R_1 pour

transférer des pièces entre ces deux lieux : Z_2 appartient au convoyeur et M_1 représente une machine. À l'inverse, il existe une relation d'accessibilité interne indirecte entre Z_1 et Z_3 car ils appartiennent tous deux au convoyeur (Z_1 et IS_5 d'une part et IS_5 et Z_3 d'autre part sont en relation d'accessibilité directe).
Etant donné une pièce située sur un lieu de départ comme la machine M_s sur la **Figure 2**, la fonction pilotage doit d'abord définir sa destination finale en fonction de la prochaine opération à réaliser du point de vue de sa gamme d'usinage. Compte-tenu de l'état des ressources de transformation, le pilotage va allouer la machine M_d pour réaliser cette opération de transformation. Le GCST de transport sera alors chargé de gérer le transport de la pièce jusqu'à M_d. La route est définie dynamiquement en fonction de l'état des ressources de transport (pannes, saturation des stocks intermédiaires). La présentation de la construction du GCST est en dehors de cette étude. Par la suite nous focalisons sur la construction des allocateurs des ressources de la fonction pilotage et sur la conception d'une GOE générique permettant d'interpréter les gammes de fabrication de chaque pièce dans une approche de type open shop.

3.3 Présentation du formalisme de modélisation

Pour la modélisation de nos contrôleurs discrets, nous avons choisi les Colored Petri Nets (CPN) de Jensen. Dans les CPN, à chaque place doit être affecté un type pour définir la classe d'appartenance des jetons autorisés. On peut associer des expressions, voire des fonctions écrites dans le langage ML. On peut également associer aux transitions, des expressions booléennes appelées « gardes » qui doivent être vraies pour qu'une transition validée soit franchie. D'autre part, le formalisme donne la possibilité de relier différents modèles en fusionnant des places (utilisation de tag de fusion pour simplifier graphiquement la représentation du modèle).

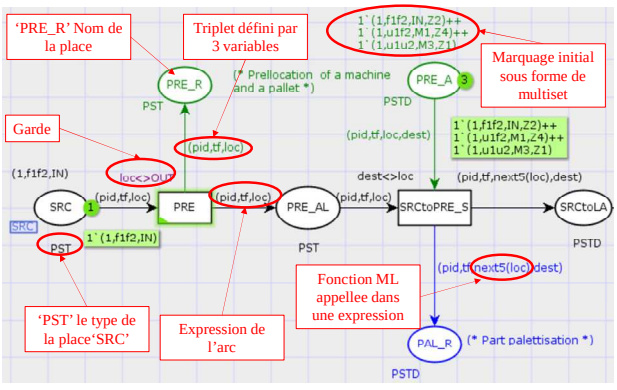

Figure 3 : Synthèse de la syntaxe du Réseau de Petri CPN

La **Figure 3** donne une synthèse de la syntaxe d'un modèle CPN. En raison de la taille limitée de l'article, cette syntaxe et la sémantique sera précisée dans les sections suivantes lors de la présentation des différents modèles.

4 LES GAMMES OPERATOIRES ETENDUES

Une Gamme Opératoire modélise toutes les opérations à appliquer pour transformer une pièce brute en une pièce finie. Elle est étendue afin de prendre en compte ses interactions pour demander au pilotage de lui allouer une machine pour réaliser l'opération de transformation suivante de sa gamme de fabrication, et ensuite, de demander au GCST de coordonner les ressources de transport afin d'acheminer le produit vers cette machine.
Dans (Toguyeni 2006), pour chaque type de pièces caractérisé par une gamme de fabrication distinct, nous définissions un modèle CPN distinct. Compte tenu du nombre d'opérations dans la gamme de fabrication et des flexibilités des machines correspondant aux différentes opérations de transformation, la GOE qui en résultait avait une combinatoire très importante. En effet, pour fabriquer une pièce de type f_1f_3 sur le système de production de la Figure 1, la GOE résultante a 52 places et 29 transitions. En réalité, la gamme de fabrication d'un produit comporte des dizaines d'opérations de traitement et chaque opération peut être effectuée sur des dizaines de machines différentes. Il est donc nécessaire de proposer un nouveau modèle de GOE qui permette de réduire cette combinatoire.

Pour cela nous prenons en compte le schéma générique de transfert donné par la **Figure 2** ainsi que le principe de routage présenté au paragraphe 3.2. Sur la base de ces données, nous précisons par un diagramme de séquence, les opérations à mettre en œuvre du point de vue du GOE (**Figure 4**). Il montre que chaque transfert de pièce nécessite systématiquement 5 opérations :
- Une *pré-allocation* sur le lieu d'origine (M_s) pour définir la destination de la pièce (*DA* pour Destination Area).
-Une *palettisation* de la pièce de la machine source M_s sur la zone de palettisation (appelée ici *LA* pour Local Area), en relation directe d'accessibilité avec la machine source.
- Un *transfert de LA vers DA*. *LA* et *DA* sont a priori en accessibilité indirecte. Afin de ne pas modéliser au niveau de le GOE toutes les possibilités de routage entre deux zones caractéristiques en accessibilité indirecte, on se contente de modéliser le fait que la pièce est en transfert entre *LA* et *DA*. La flexibilité du transfert et donc les combinatoires générées par cette flexibilité sont prises en compte par le modèle CPN du GCST.
- Une demande d'*allocation* de la machine de destination M_d lorsque la pièce palettisée arrive en *DA*.
- Une *dépalettisation* de la pièce en *DA* et son *chargement* sur la machine de destination M_d.
Une fois la pièce arrivée en M_d, la GOE doit lancer le programme de fabrication correspondant à l'usinage demandé. Lorsqu'il est terminé, la situation de la pièce usinée sur M_d est exactement la même que la situation initiale considérée sur la machine M_s. Ainsi, il est évident de considérer que M_d joue maintenant le rôle de la machine M_s précédente et, qu'il est alors possible de recommencer toute la procédure décrite ici.

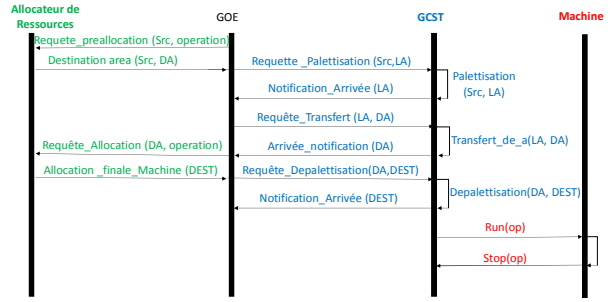

Figure 4 : Diagramme de séquences pour le transfert d'une pièce basé sur le schéma de la Figure 2

Pour être générique, il faut traiter des cas spécifiques tels que le départ de la pièce du stock *IN* et son retour au stock *PiOUT* lorsqu'elle est terminée. En effet, lorsqu'il n'y a plus d'opération de fabrication, il est nécessaire que la fonction de *pré-allocation/allocation* oriente la pièce vers la sortie du système (cas 1). Un autre cas spécifique est celui où la *pré-allocation* désigne la machine actuelle comme celle qui doit effectuer l'opération de fabrication suivante. Cela signifie qu'à ce stade, la *pré-allocation* fonctionne comme une *allocation* et qu'il n'est donc pas nécessaire de transférer la pièce puisqu'elle se trouve au bon endroit (cas 2).

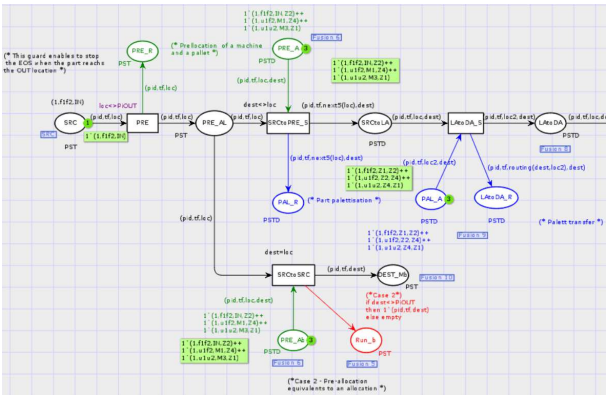

Figure 5 : GOE générique – partie 1

Les figures 5, 6 et 7 montrent le modèle CPN générique d'une GOE. Il est décomposé en 3 figures en raison de sa taille et pour permettre une meilleure lecture. Certaines places sont répétées dans ces figures afin de montrer le lien entre les différentes parties du modèle. Par exemple, la place "*LAtoDA*" (qui signifie un transfert du lieu "*LA*" au lieu "*DA*") est répétée dans les figures 5 et 6. Les places et transitions de couleurs noires sur ces figures modélisent la séquence des états opératoires pour la réalisation d'une opération de transformation. La GOE est conçue comme une fonction qui interprète la gamme de fabrication de chaque pièce qui est modélisée par une liste d'opérations de transformation à réaliser. Les parties vertes représentent les communications asynchrones (interfaces) avec les allocateurs de ressources. Les parties bleues représentent les interfaces avec le GCST. Les parties rouges représentent les interfaces avec les modèles des machines.

Les modèles CPN sont basés sur les structures de données ML suivantes :

(Définition de la localisation d'un produit *)*
colset OSA=with
IN|INvZ1|Z1vZ2|Z2vZ3|Z3vZ4|Z1vZ3|Z3vZ1|Z4vZ1|Z1v
OUT|Z2vM1|M1vZ2|Z3vM2|Z3vM4|M2vZ3|M4vZ3|Z4v
M3|M3vZ4|M1|M2|M3|M4|OUT|
IS1|IS2|IS3|IS4|IS5|IS6|Z1|Z2|Z3|Z4;
(Définition des états de fabrication des produits bruts, semi-finis et finis *)*
colset TRSF=with
f1|u1|f2|u2|f3|u3|f4|u4|f5|u5|f1f2|u1f2|f1u2|u1u2|f3f1f4|u
3f1f4|u3u1f4|u3f1u4|u3u1u4;
(Définition de l'état d'un produit *)*
*colset PST=product PID*TRSF*OSA;*
(Définition de l'état d'une pièce et de sa destination finale*)*
*colset PSTD=product PID*TRSF*OSA*OSA;*

Dans un modèle CPN, il est nécessaire d'affecter un type à chaque place du modèle. Ainsi, les places de la GOE sont de deux types : *PST* ou *PSTD*. Le type *PST* définit l'état d'un produit du point de vue d'une opération de transformation (type *TRSF*) et du point de vue sa localisation dans le système de production (type *OSA*). La place "*SRC*" (**Figure 5**) est la première place du modèle. Elle définit chaque produit sous la forme d'un triplé *(produit_identification, gamme_fabrication, localisation)*. Ainsi, le jeton *(1,f1f2,IN)*, spécifie que l'on a une pièce de type f_1f_2 présente sur le stock d'entrée des pièces *IN* et qu'elle a pour identifiant la valeur 1. L'identifiant de chaque pièce est défini par rapport à son type et permet de savoir qu'elle pièce est traitée par rapport à l'en cours du système.
La place *"N_SRC"* (figure 11) représente la fin du GOE conformément au diagramme de séquences de la **Figure 4**. Elle est également de type *PST*. En fait les places "*SRC*" et "N_*SRC*" sont fusionnées par le tag 'src' de couleur bleu dans les figures. Cela permet de dire que ces deux places n'en font en fait qu'une, permettant de donner au modèle de GOE un fonctionnement récursif. En effet, lorsque le jeton arrive dans la place "*N_SRC*", cela signifie que toutes les opérations nécessaires pour réaliser une opération de transformation de la gamme d'usinage ont été exécutées. Par exemple, par rapport au produit qui était modélisé par le jeton *(1,f1f2,IN)* dans "*SRC*", on pourra retrouver un jeton *(1,u1f2,M2)* dans "*N_SRC*". Cela signifie que le produit se trouve sur la machine M2, sur laquelle a été réalisée l'opération de transformation *f1*, conduisant à l'obtention d'un produit semi-fini de type *u1f2*. La fusion des deux places fait que l'on a alors le jeton *(1,u1f2,M2)* dans la place "*SRC*", indiquant que l'on doit à présent recommencer tout le processus du GOE pour réaliser l'opération *f2*.
Il est à noter la garde définie par l'expression "*loc<> PiOUT* " qui est associée à la transition "*PRE*". Cette condition permet de ne pas relancer la GOE pour une pièce qui a été sortie sur le stock de sortie des pièces "*PiOUT*".

Certaines des places d'interfaces du GOE sont de type *PSTD*. Ce type ajoute la localisation de destination (qui est de type *OSA*) au type *PST* précédent. Par exemple, c'est le cas de la place "*PRE_A*" (**Figure 5**) qui est une place d'interface avec l'allocateur de resources. Par exemple, le jeton *(1, f1f2, IN, Z3)*, signifie que la pièce 1 localisée sur le stock IN doit être transportée vers le poste Z3 du convoyeur. C'est une *pré-allocation* qui sera expliquée dans la section suivante. La conséquence c'est que toutes les places d'interfaces avec le GCST (places de couleur bleue sur les figures) sont de type *PSTD*.

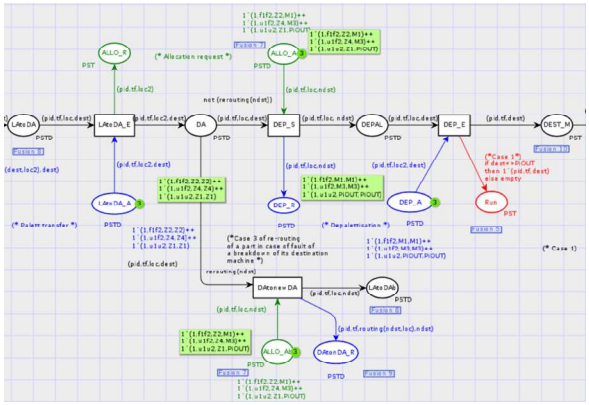

Figure 6 : GOE générique – partie 2

Les place "*PRE_R*", "*PRE_A*" et "*PRE_Ab*" (**Figure 5**) servent à gérer une demande de préallocation. "*PRE_R*" est le sémaphore de requête et "*PRE_A*" est le sémaphore de réponse. La transition "*SRCtoSRC*" modélise la situation d'une *pré-allocation* équivalente à une *allocation* (à noter que les places "*PRE_A*" et "*PRE_Ab*" sont fusionnées par le tag "*fusion 7*"). Cette situation survient quand la machine allouée correspond à la machine même sur laquelle se trouve la pièce (cas 2).

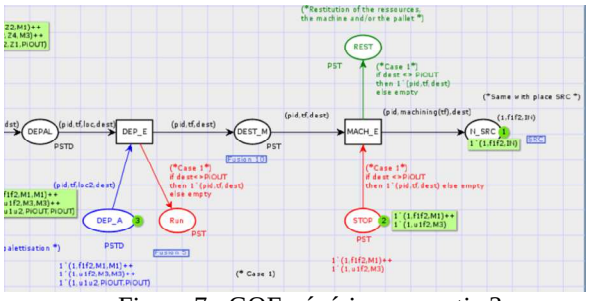

Figure 7 : GOE générique – partie 3

Dans ce cas, le jeton modélisant la pièce va directement de la place "*SRC*" au lieu modélisant sa présence sur la machine de destination. Par conséquent, la place "*DEST_Mb*" (respectivement "*Run_b*") de la **Figure 5** est fusionnée avec la place "*DEST_M*" (respectivement "*Run*") de la **Figure 6** par le tag "*fusion10*" (respectivement "*fusion 5*"). Une construction similaire est effectuée dans la **Figure 6** par la transition "*DAtonewDA*". Elle modélise la réorientation d'une pièce vers une nouvelle destination ("*newDA*") en cas de défaillance de la machine de destination "*Md*" pendant le transfert de "*LA*" à "*DA*".

5 LE PILOTAGE

Dans notre approche, le rôle du pilotage est de résoudre les indéterminismes résiduels de la partie commande. C'est une fonction intelligente qui peut être mise en œuvre de différentes manières et avec différents formalismes. Cette fonction prend notamment en charge l'allocation des ressources, afin d'arbitrer les demandes conflictuelles. Les ressources considérées sont les palettes, les machines, les robots, les aiguillages et les jonctions du convoyeur. En raison de la place limitée, dans cette étude, nous avons choisi de ne présenter que l'allocation des machines. Nous avons choisi ici d'utiliser le formalisme des CPN. Cela permet de vérifier une GOE, en intégrant le pilotage comme faisant partie de son environnement.

Une originalité de ce travail, est de distinguer deux étapes en matière d'allocation d'une machine : la *pré-allocation* et l'*allocation* finale. La *pré-allocation* permet d'augmenter les performances du système de production.

Pour comprendre l'idée, revenons au schéma générique du transport d'un produit donné par la **Figure 2**. Du point de vue du transport, il est nécessaire d'effectuer l'affectation de la machine de destination (M_d) avant le départ de la pièce. Si sa capacité est d'une unité, cela signifie qu'on ne pourrait plus affecter cette machine à d'autres produits. Par conséquence, tous les autres produits qui devraient être usinés en M_d seraient bloqués sur leur localisation courante, au lieu de se rapprocher de M_d en temps masqué. Cela limiterait donc la productivité du système. Pour une opération de transformation donnée, la *pré-allocation* consiste à indiquer le poste de chargement (noté *DA*) de la machine où le produit doit être transformé. Cette *pré-allocation* n'exige donc pas que la machine soit libre. Elle a juste besoin qu'elle ne soit pas défaillante. Ainsi, l'allocation finale est demandée seulement lorsque le produit est arrivé sur *DA*, donc à proximité de la machine M_d. Si la machine de destination choisie par la *pré-allocation* n'est pas défaillante, l'allocation consiste à faire passer son statut de libre (free) a occupé (occupied), puis à indiquer en conséquence à la GOE la destination finale comme correspondant à M_d.

Dans le GOE, la *pré-allocation* est modélisée par la paire de places "*PRE_R*"/"*PRE_A*"

L'allocation d'une machine correspond donc à la définir comme lieu final de destination d'un produit. C'est la raison pour laquelle dans notre modèle de GOE générique présenté à la section précédente, nous avons la paire de places "*ALLO_R*"/"*ALLO_A*" qui modélisent respectivement la demande d'allocation et la réponse donnée par l'allocateur de ressources.

En fait, la pré-allocation et l'allocation fonctionnent très étroitement. Elles dépendent toutes deux de l'état de la machine et de l'emplacement de la pièce au moment où elles sont requises. Pour la pré-allocation, la machine doit être disponible (pas en panne) et la demande doit être faite à un endroit source tel que M_s (**Figure 2**). Pour

l'allocation, la machine doit être libre et opérationnelle et la demande doit est faite sur *DA* le poste de chargement de M_d, la machine allouée. En cas de panne de cette machine (survenu au cours de l'acheminement), la demande d'allocation est implicitement interprétée par le pilotage comme une requête de *pré-allocation*. Il y a alors réacheminement de la pièce vers une autre machine mettant en œuvre la même fonction que celle qui est devenue défectueuse lors du transfert de la pièce vers *Md* (cf. transition "*DAtonewDA*" sur la figure **Figure 6**).
A contrario, en cas de demande de pré-allocation, si la machine pré-allouée est la machine sur laquelle se trouve la pièce, la requête est interprétée par le pilotage comme une demande d'allocation (cf. transition "*SRCtoSRC*" dans la **Figure 5**) Ces deux cas montrent qu'il est nécessaire d'implémenter ensemble la pré-allocation et l'allocation. Par conséquent, c'est le même contrôleur qui met en œuvre les deux fonctions. Le comportement se différencie en fonction de la localisation de la pièce au moment de la demande et en fonction de l'état de la machine.

Pour implémenter le contrôleur d'allocation des machines, définissons les types et variables suivants dans CPN ML.

(Défini le statut d'une machine *)*
colset STAT=with free|occupied|faulty ;
colset RES=subset OSA with [M1,M2,M3,M4];
colset RAWOP=subset TRSF with [f1,f2,f3,f4];
(Défini une liste d'opérations de transformation *)*
colset LOP=list RAWOP;
(Défini une machine par rapport à son état et sa liste d'opérations de transformation *)*
*colset MACH=product RES*STAT*LOP*OSA;*
(Défini le type machine alloué *)*
colset ALMAC=product
*RES*PID*TRSF*LOP*OSA*OSA;*

Ainsi le type MACH, permet de spécifier une machine du point de vue de l'allocateur. Par exemple, le jeton *(M2,free,[f1,f3,f1f2],Z3)*, indique que la machine *M2* est libre, et qu'elle peut réaliser des pièces de type *f1*, *f3* ou *f1f2*. La valeur *Z3* indique le lieu dans la système à partir duquel on peut faire une demande d'allocation. Dans tous les autres emplacements, toute demande d'allocation reçue par la place "*ALLO_R*" (demande d'allocation) est considérée comme une demande de pré-allocation. Cette place du modèle de l'allocateur (**Figure 8**) est fusionnée avec la place "*ALLO_R*" de la GOE (**Figure 8**).
Ces types sont utilisés pour définir le type des places du CPN de la **Figure 8**. Ensuite, définissons les variables qui sont utilisées pour écrire les expressions du modèle.

(Identifiants de pièces *) var pid,pid2:PID;*
(Variables de localisation *) var loc, loc2,dest,ndst : OSA;*
*(*Variables des opérations d'usinage *) var tf,tf2 : TRSF;*
(Liste d'operations d'usinage réalisées par une machine *) var ltf:LOP;*
(Variable statut d'une machine *) var st:STAT;*
(Variable définissant une machine *) var m:RES;*

En utilisant ces variables, on peut écrire les expressions des arcs CPN. Par exemple, l'expression de l'arc entre la place *"Standby"* et la transition "*Reqproc*" est *(m,st,ltf,loc)*. Cette expression signifie que la transition "*Reqproc*" est validée s'il y a au moins un jeton de ce type à la place "*Stanby*" et un jeton d'expression *(pid,tf,loc2)* dans la place "*ALLO_R*". La place "*Standby*" est la place par défaut qui modélise l'état d'une machine disponible.

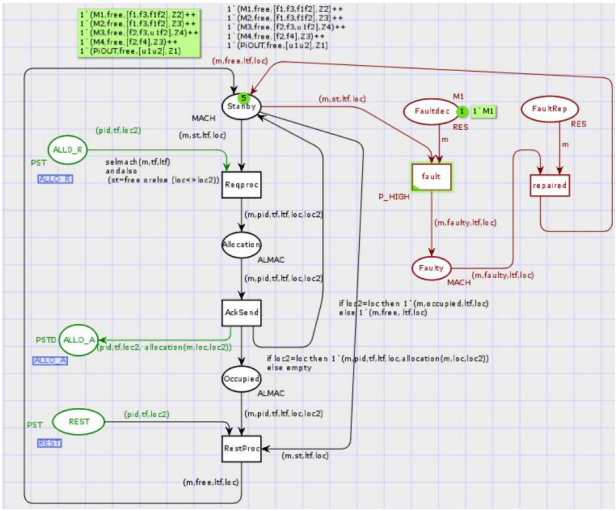

Figure 8 : Allocateur de machines

Pour sélectionner une machine qui peut être allouée, définissons la fonction ML "*selmach*". Sa définition est la suivante :
fun selmach(r:RES, ope:TRSF, lope:LOP)=
if lope=nil then false
else if ope=hd lope then true
else selmach(r,ope, tl lope) ;

Cette fonction est utilisée dans la garde de la transition "*Reqproc*" afin de vérifier si la machine '*r*' met en œuvre l'opération '*ope*'. Cette opération doit faire partie de la liste des opérations de '*r*'. Cette fonction ML est définie de manière récursive.
De la même manière, définissons en ML, la fonction d'allocation.
fun allocation(r:RES,l:OSA,l2:OSA)=
if l2=l then r else l;

Cette fonction est utilisée dans l'expression de l'arc entre la transition "*AckSend*" (Acquittement envoyé) et la place "*ALLO_A*". Elle définit la destination d'une pièce. En pratique, bien que nommée *allocation*, elle implémente également la *pré-allocation*. En effet, si '*l2*' le lieu de demande correspond à '*l*', le lieu de chargement de la ressource '*r*', alors la destination est la ressource '*r*', et c'est donc une allocation. Sinon la destination est '*l*', et c'est donc une pré-allocation.

Dans la **Figure 8**, toute la partie du modèle en couleur verte représente l'interface avec la GOE. La partie en violet modélise la gestion des pannes et des réparations des machines. Le reste du modèle de couleur noire représente le processus de *pré-allocation/allocation*. Il convient de noter l'utilisation d'une expression alternative pour les arcs entre la transition "*AckSend*" et les places "*Standby*" et "*Occupied*" respectivement. En cas de *pré-allocation* (cas où *loc2<>loc*), aucun jeton n'est ajouté à la place "*Occupied*" et un jeton de la machine est remis en place "*Attente*" avec la valeur libre (‘*free*’) pour l'état de la machine. Dans le cas d'une *allocation*, le jeton de la machine est mis à la fois à la place "*Standby*" et à la place "*Occupied*" avec la valeur occupée (‘*occupied*’) pour son statut. On peut être surpris que le jeton soit remis dans la place "*Standby*". Cela permet à d'autres pièces de pouvoir demander des *pré-allocations* même si la machine est en train de fabriquer une pièce.

La garde de transition '*RestProc*' est définie par l'expression *selmach(m,tf,ltf) andalso (st=free orelse (loc<>loc2))*. Cette expression signifie qu'on fait une préallocation si la position de la pièce (*loc2*) et différente de la position de demande d'allocation de la machine (*loc*). Si la pièce est en *loc*, il faut que la machine choisie soit libre (*st=free*).

Lorsqu'une machine est restituée après évacuation de la pièce déchargée sur *DA* (dans le cas où un jeton est placé dans la place "*REST*"), il est nécessaire de réinitialiser à la valeur "*free*" le champ d'état du jeton de la machine dans la place "*Standby*". C'est pourquoi la transition '*RestProc*' est validée simultanément par les places '*Standby*' et '*Occupied*'. Lorsque cette transition est déclenchée, elle retire les jetons des deux places et remet dans la place "Standby" un jeton dont la valeur du champ d'état a été réinitialisée à la valeur "*free*".

6 CONCLUSION

Cette étude est à la fois un complément et une amélioration des travaux que nous avons proposés dans (Toguyeni 2006). Son objectif principal était de proposer une solution à l'explosion combinatoire des gammes opératoires étendues (GOE). Pour cela nous avons proposé une modélisation récursive de la gamme. La récursivité est obtenue en parcourant la GOE pour chaque opération de la pièce définie dans une liste d'opérations. Cela est possible grâce aux formalismes des CPN qui permettent de manipuler des listes. Quand la liste des opérations a été complètement exploitée la gamme s'arrête avec l'arrivée de la pièce sur le stock de sortie.

Dans cette étude, nous avons également proposé une modélisation des allocateurs des ressources illustrant l'intérêt qu'ils prennent en compte conjointement les fonctions de *pré-allocation* et d'*allocation*. Cela permet d'exploiter au maximum les flexibilités pour la reconfiguration dynamique du système.

Les perspectives de ce travail sont doubles. Dans un premier temps, il s'agit de développer la vérification formelle de telles modèles, soit en exploitant les fonctionnalités des outils de CPN Tools comme la possibilité d'utiliser le model-checker Ask-CTL (Christensen and Mortensen, 1996), soit en proposant des méthodes de vérifications formelles basées sur le calcul d'invariants d'un modèle CPN. Une autre perspective concerne la transcription automatique des modèles CPN dans les langages de la norme IEEC 61131-3 pour la programmation des automates programmables industriels ou des langages de haut niveau. L'exploitation des techniques de l'ingénierie dirigée par les modèles permettrait la transcription automatique dans du code exécutable sur des calculateurs industriels.

REFERENCES

Christensen S. and K. H Mortensen, 1996. Design/CPN ASK-CTL Manual, version 0.9. Available from: http://cpntools.org/wp-content/uploads/2018/01/askctlmanual.pdf [accessed 11 May 2018]

Jensen K. and L. M. Kristensen, 2015. Colored Petri nets: a graphical language for formal modeling and validation of concurrent systems. *Communications of the ACM, 58(6)*, p. 61-70.

Koren Y, 2014. Reconfigurable Manufacturing System. *Preprint of the CIRP Encyclopedia of Production Engineering*. Springer Berlin Heidelberg, p. 1035-1039.

Mittal, S., Khan, M. A., Romero, D. and Wuest, T. (2019). Smart manufacturing: characteristics, technologies and enabling factors. *Proceedings of the Institution of Mechanical Engineers*, Part B: Journal of Engineering Manufacture, 233(5), p. 1342-1361.

Qu, Y. J., Ming, X. G., Liu, Z. W., Zhang, X. Y. and Hou, Z. T. (2019). Smart manufacturing systems: state of the art and future trends. *The International Journal of Advanced Manufacturing Technology*, 103(9-12), p. 3751-3768.

Radziwon A., Bilberg A., Bogers M. and E. S. Madsen, 2014. The smart factory: exploring adaptive and flexible manufacturing solutions. *Procedia Engineering, 69*, p. 1184-1190.

Toguyeni A., 2006. Design of modular and hierarchical controllers for reconfigurable manufacturing systems. *Computational Engineering in Systems Applications, IMACS Multiconference on. IEEE*, p. 1004-1011