Utilisation adaptée au niveau des apprenants et validation de la commande des Systèmes Automatisés de Production

Pascale MARANGE[1], François GELLOT[1], Bernard RIERA[1], Jean-Paul CHEMLA[2]
{pascale.marange,francois.gellot,bernard.riera}@univ-reims.fr, jean-paul.chemla@univ-tours.fr,
[1]IUT Reims-Châlons-Charleville et UFR Sciences Exactes et Naturelles,
CReSTIC, Université de Reims-Champagne Ardenne, Moulin de la Housse, BP 1039 – 51687 Reims,
[2] Université François Rabelais de Tours, LUSSI FRE 2448,
Polytech'Tours, Département Productique, 7 av Marcel Dassault, 37000 Tours,

RESUME : L'enseignement de la commande des systèmes à événements discrets nécessite le transfert de savoir (connaissance) et de savoir-faire (compétence). Concernant le deuxième point, même si la simulation offre des possibilités intéressantes, il est important que les apprenants travaillent sur des systèmes automatisés réalistes comprenant de nombreux capteurs et actionneurs et exploitant les technologies modernes des automatismes. Toutefois, l'utilisation de systèmes industriels même simplifiés pose nécessairement des problèmes de sécurité mais aussi des problèmes d'adaptation du système au niveau des apprenants (du novice à l'expert) pour lesquels nous proposons des solutions dans cet article. Pour garantir la sécurité des utilisateurs et de l'équipement, une approche classique de validation de la commande à l'aide d'un filtre situé entre la commande et la partie opérative est proposée. Le filtre comprend des contraintes logiques qui ne doivent en aucun cas être violées. Il nous semble important quelque soit le niveau de l'apprenant de lui donner la possibilité de commander la partie opérative dans sa globalité. Pour cela, il est proposé d'adapter le niveau de difficulté de la commande à l'utilisateur en modifiant le niveau d'automatisation du système mais en conservant une vision globale du système automatisé. Ce travail permet de mettre à la disposition d'étudiants de différents niveaux de l'Université de Reims Champagne-Ardenne (IUT, Licence Pro, Master EEA) et en toute sécurité la machine « Productis ». De plus, ce travail a aussi trouvé un applicatif original en permettant à des enfants de CM2 de réaliser leur premier programme de commande d'un système automatisé. Enfin, les résultats obtenus permettent d'envisager en toute sécurité une utilisation distante des maquettes pédagogiques utilisées dans le cadre des formations en automatique.

Mots clés : Commande, Validation, Systèmes Automatisés de Production, Identification fonctionnelle, Filtre

1 INTRODUCTION

L'utilisation des technologies de l'information et de la communication est une réalité dans le domaine des automatismes. Cette utilisation fournit différentes possibilités pour l'enseignement « pratique » de la théorie des systèmes à événements discrets (S.E.D.). L'idée que nous développons est de donner la possibilité aux étudiants d'utiliser de manière distante ou non, du matériel professionnel (commande et partie opérative) et des progiciels. L'apprentissage de l'automatisme passe par un savoir (théories) et un savoir-faire (application du savoir). Le savoir-faire acquis doit être si possible en adéquation avec les réalités et les besoins du monde industriel. Pour cela, il est important que l'apprenant soit confronté à des problèmes « réels » et utilise donc des parties opératives « réalistes ».

Toutefois, l'utilisation de systèmes automatisés « réels » requiert plus de vigilance de la part de l'enseignant, a un coût important, présente nécessairement des risques de sécurité et de casse, et son entretien exige du personnel qualifié.

La solution que nous proposons dans cet article vise à donner à l'apprenant un champ d'actions sur la machine adapté à son niveau tout en garantissant la sécurité. Pour cela, nous proposons une approche à base de filtre qui permet de valider la commande de l'apprenant de manière pédagogique et en toute sécurité pour le système et les utilisateurs. Elle est basée sur la définition de contraintes logiques qui ne doivent en aucun cas être violées. Afin d'adapter le niveau de difficulté au niveau de l'apprenant, **tout en conservant une vision globale du système automatisé**, nous proposons de modifier le niveau d'automatisation et le niveau d'autonomie de l'étudiant. Il nous semble en effet qu'il est beaucoup plus valorisant pour un étudiant de proposer une commande permettant de faire fonctionner une partie opérative dans sa globalité.

Dans une première partie, 3 paramètres sont définis pour caractériser le niveau de difficulté d'un problème de commande : la granularité, la synchronisation et la hiérarchisation. Nous proposons d'intervenir sur ces trois dimensions en vue de modifier le niveau de difficulté de la commande tout en conservant une vision globale de la partie opérative. Dans une seconde partie, l'approche de validation par filtre sera présentée. Ce filtre est composé d'un premier filtre au niveau fonctionnel, d'un estimateur et d'un deuxième filtre au niveau du système basé sur la définition de contraintes de sécurité (ce qu'il ne faut pas faire). Ce travail a conduit à 3 squelettes de programme API (Automates Programmables Automates) qui sont utilisés à l'Université de Reims

Article available at http://www.j3ea.org or http://dx.doi.org/10.1051/j3ea:2008002

dans différentes formations (IUT et UFR Sciences). La dernière partie de l'article concerne une utilisation originale de ces travaux. En effet, suite au succès rencontré auprès des écoles maternelles il y a 2 ans lorsque nous avons proposé à des enfants de 5 ans de découvrir la magie des automatismes [1] au moyen d'applications réalisées par les étudiants sur la machine « Productis », nous avons décidé cette fois d'offrir la possibilité à des enfants de 9 ans de réaliser leur premier programme de commande en utilisant les développements présentés dans cet article.

2 DEFINITION DU CHAMP D'ACTIONS

Il est évident que la proposition et la définition d'un problème de commande doivent être adaptées à l'étudiant. Le niveau, la connaissance et la compétence d'analyse exigés ne sont pas les mêmes pour un étudiant qui découvre l'automatisme ou pour celui qui suit un cours de spécialisation. Pour rendre l'apprentissage intéressant et motivant, il est important de mettre à la disposition des apprenants des systèmes réels. Cependant, cela induit nécessairement des contraintes qui influencent considérablement le niveau de difficulté d'un problème de commande. Nous formalisons dans les paragraphes suivants les paramètres qui influent directement sur le degré de difficulté ou la complexité d'un problème de commande. Volontairement, le « point de vue de l'étudiant » et sa perception du niveau de difficulté ne seront pas considérés.

2.1 Paramètres liés à la difficulté

Le concept de difficulté est à relier au concept de complexité qui est caractérisé par le nombre élevé de variables, par l'interaction entre ses variables, … La perception de la complexité du système, son analyse, et sa modélisation sont spécifiques aux objectifs pédagogiques que l'enseignant s'est fixé. Les travaux de Lind [2] considèrent que les systèmes de production peuvent être définis selon deux axes: « Moyens-Buts » et « Tout-Partie ». Selon l'auteur, en distinguant les « moyens » des « buts », un système peut être décrit en termes de « buts », de « fonctions » et de « composants physiques ». De plus, chacune de ces descriptions peut être donnée à différents niveaux de l'axe de décomposition « Tout-Partie ».

Nous pouvons lier cette vision du système au niveau de difficulté (ou de complexité) d'un problème de commande. Une partie opérative (PO) est composée et perçue par un ensemble de capteurs et d'actionneurs fixé et difficilement modifiable. La solution la plus simple pour simplifier le système et donc sa commande est de ne pas considérer certains éléments de PO. Toutefois cette solution ne permet pas de conserver la vision globale du système. L'approche que nous proposons consiste à percevoir le système selon différents niveaux de l'axe « Moyens – Buts » en vue de simplifier la perception du système à commander. Il est évident que la complexité de la commande n'est pas liée directement à la dimension de la partie opérative mais au cahier des charges. D'après notre expérience d'enseignant, la difficulté d'un problème de commande peut s'exprimer au travers de 3 paramètres, dépendants les uns des autres, et caractérisant la commande qui répond au cahier des charges : la dimension, la hiérarchisation et la synchronisation entre les éléments.

2.1.1 Granularité ou dimension

La dimension de la commande est directement liée au nombre de sous-ensembles devant être commandés. Elle dépend au niveau le plus bas de la décomposition du nombre de capteurs et d'actionneurs nécessaires pour concevoir la commande décrite par le cahier des charges. Plus la granularité est fine, plus l'effort d'analyse est important pour l'apprenant et plus le niveau de difficulté est élevé. Il nous semble que plus l'apprenant est expert plus il est capable de travailler au niveau le plus bas de l'axe « Moyens-Buts », c'est-à-dire : au niveau des capteurs et des actionneurs.

2.1.2 Hiérarchisation

La gestion d'un cycle « normal » sans tenir compte des divers modes de fonctionnement nécessite une commande séquentielle simple. En revanche, un cahier des charges « complet » intégrant différents modes de marche et arrêt nécessite une solution de commande hiérarchisée. L'étape de spécification est alors plus difficile. Dans la pratique, la hiérarchisation apparaît au travers des ordres de forçage de Grafcet par exemple.

2.1.3 Synchronisation

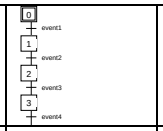
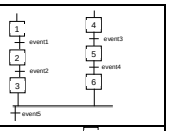
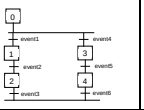
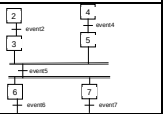
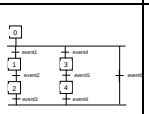
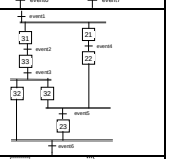
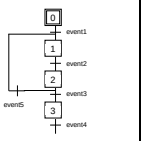
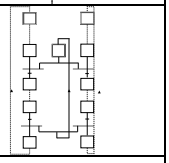
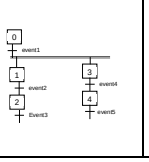
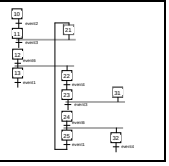
Cycle d'une séquence simple **(très facile)**		Synchronisation de séquences **(moyen))**	
Sélection de séquences **(facile)**		Synchronisation et activation de séquences parallèles **(moyen)**	
Saut d'étapes **(facile)**		Si événement Alors … **(difficile)**	
Reprise de séquence **(moyen)**		Gestion ressource commune **(très difficile)**	
Activation de séquences parallèle **(facile)**		Séquences alternées **(très difficile)**	

fig. 1. Différentes structures de GRAFCET

Les deux paramètres précédents vont faire apparaître des synchronisations dues à la définition de la granularité du système (évolution simultanée d'événements) et à la hiérarchisation engendrée par des priorités entre les différents modes de fonctionnement. Toutefois, les problèmes de commande peuvent aussi

nécessiter pour être résolus des structures particulières de gestion d'événements ou de ressources communes. La figure 1 indique quelques structures en Grafcet souvent rencontrées et leur niveau de difficulté [3].

Nous proposons de jouer sur la perception du système par l'apprenant pour modifier le niveau de difficulté d'un problème de commande.

2.2 Méthodologie pour adapter le niveau de difficulté

La définition d'un cahier des charges suivant l'axe « Moyens - Buts » va permettre de moduler la difficulté induite par chacun des 3 paramètres qui viennent d'être présentés, tout en conservant une vision globale de la PO à commander.

Nous proposons donc d'adapter le niveau de difficulté en établissant une identification fonctionnelle du système et en donnant plus ou moins d'autonomie à l'apprenant. L'identification fonctionnelle va permette de définir des fonctions caractérisant un ordre ou un ensemble d'ordres plus ou moins complexes.

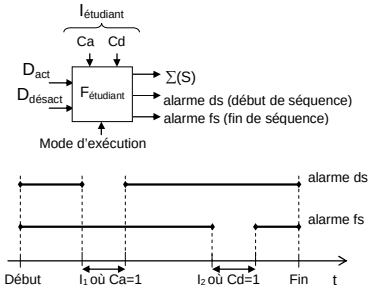


fig. 2. Définition d'une fonction

L'identification fonctionnelle faite par l'enseignant va donc définir le champ d'actions de l'apprenant c'est-à-dire les fonctions ($F_{étudiant}$) qu'il a à sa disposition pour commander le système, ainsi que les conditions d'activation Ca et les buts à atteindre représentés par les conditions de désactivation Cd (figure 2). Cela permet à l'enseignant d'encapsuler des notions de synchronisation et des structures non adaptées à l'apprenant à l'intérieur d'une fonction. Suivant le niveau de l'apprenant, l'enseignant peut choisir d'exécuter la fonction en mode autonome ou non. Dans le premier cas, l'activation et la désactivation de la fonction seront faites automatiquement quand les conditions d'activation et de désactivation sont respectivement vraies. Dans le second cas, l'étudiant doit activer ou désactiver la fonction lorsque les conditions sont remplies, sinon il est averti d'une erreur. Le principe d'exécution d'une fonction est le suivant. L'activation ou la désactivation d'une fonction ne sera effective que si les conditions d'activation ou de désactivation sont présentes. Si l'étudiant fait la demande d'activation (D_{acti}) de la fonction dans l'intervalle I_1, où les conditions d'activation sont présentes (Ca_i), l'envoi est correct et la fonction exécute l'ensemble des « sorties système » ($\Sigma(S)$). Par contre, si la demande est faite à l'extérieur de I_1, une alarme ds_i est émise. Il en est de même pour l'alarme fs_i qui est émise, si la demande de désactivation ($D_{désacti}$) n'est pas faite dans l'intervalle I_2.

Cette approche de modélisation par fonction permet de mettre en évidence des erreurs classiques de commande commises par les étudiants telles que la mauvaise activation ou désactivation d'un ordre.

3 VALIDATION DE LA COMMANDE

Les travaux dans le domaine de la validation de la commande visent à vérifier que des propriétés mathématiques sont respectées par le modèle de commande [4], [5]. Le travail entrepris dans le cadre de l'outil UPPAAL [6] définit trois types de propriétés : atteignabilité, sécurité et vivacité. Dans ce travail, nous considérons seulement les contraintes de sécurité, c'est-à-dire ce que le système ne doit pas faire. L'étape de validation peut être faite hors ligne ou en ligne. Dans le premier cas, la commande est complètement validée avant d'être implantée dans le système [7]. Dans le deuxième cas, la validation est faite en temps réel. Pour des raisons de faisabilité pratique, nous nous sommes orientés vers une approche en ligne de validation de la commande basée sur un filtre établi directement dans l'API. De plus, cela permet de s'affranchir des problèmes d'asynchronisme. Dans cette approche de validation, l'idée est d'empêcher les évolutions qui peuvent mener le système à une situation de risques pour les utilisateurs et le système. Nous proposons une approche à base de filtre inspirée des travaux de Cruette [8] (zone 1 fig. 3).

Dans notre approche, nous avons choisi de placer un bloc de validation entre la partie opérative et la commande. Ce bloc contient :

- Un filtre de validation fonctionnel qui valide la cohérence entre la fonction $F_{étudiant}$ et la représentation du système qui lui en est faite par l'intermédiaire des informations $I_{étudiant}$. Ce premier filtre contient des contraintes qui dépendent du champs d'actions de l'apprenant, c'est-à-dire de l'identification fonctionnelle du système (cf. 2.2).
- Un bloc estimateur qui permet d'une part de masquer les valeurs du vecteurs d'entrées (E) et ainsi de remonter l'information nécessaire ($I_{étudiant}$) à l'étudiant et d'autre part d'exécuter la fonction ($F_{étudiant}$), en envoyant la séquence de sorties (S) qui lui sont associées.
- Un filtre de validation système qui permet de valider les contraintes de sécurité. Contrairement au filtre de validation fonctionnel, il est défini une seule fois et il est valable quel que soit le cahier des charges et le niveau de l'apprenant. Il contient les contraintes de sécurité et est indépendant de l'identification fonctionnelle.

Lors de l'exécution du programme de commande, à chaque évolution ($F_{étudiant}$) de la commande, si la fonction est validée ($F_{étudiantValidée}$) par le filtre, elle est envoyée à l'estimateur qui envoie la suite d'évolutions des sorties (S) correspondant à la fonction, sinon le système est arrêté et l'étudiant est alerté de son erreur. Si la séquence de sortie (S) valide toutes les contraintes

de sécurité du filtre de validation système, les sorties ($S_{validée}$) sont envoyées vers la partie opérative qui va évoluer, sinon une alarme système averti d'une erreur et arrête le système.

C'est à l'enseignant de construire les deux filtres de validation et l'estimateur (zone 2 fig. 3). Cela revient à générer un canevas de programmation spécifique à chaque formation (i.e. niveau des apprenants). A l'Université de Reims, nous avons créé pour des API Premium de marque Schneider, 2 fichiers STX avec les mnémoniques utilisées pour définir le champ d'actions de l'apprenant, les parties de programme cachées derrière une fonction et les contraintes à valider. Lors d'une séance de TP, l'étudiant utilise donc le canevas de programmation pour faire sa commande et lors de l'exécution, l'approche de validation détecte les éventuelles erreurs et fait remonter l'information au moyen d'une application de supervision développée sous InTouch. Les paragraphes suivants détaillent les différentes parties du filtre de validation proposée.

3.1 Filtre de validation fonctionnel

A partir de l'identification fonctionnelle choisie par l'enseignant, dans ce filtre, les deux contraintes suivantes pour chacune des fonctions sont présentes: $D_{act} \wedge Ca_i = 1$, $D_{désact} \wedge Cd_i = 1$. Ces contraintes traduisent que la demande d'activation ou de désactivation ne peut se faire que si les conditions d'activation ou de désactivation sont présentes. Si le mode autonome n'a pas été choisi, la définition de ces contraintes est inutile.

3.2 Estimateur

Suite à l'identification fonctionnelle choisie par l'enseignant, il est nécessaire d'implanter dans l'estimateur les parties de commande encapsulées dans les fonctions.

3.3 Filtre de validation système

La définition des contraintes de sûreté du filtre de validation système est un problème difficile. Pour définir de manière exhaustive les contraintes, des modèles comportementaux de la partie opérative sont nécessaires. Notre approche est pragmatique et vise à proposer une classification des divers types de contraintes de sécurité. Cependant, leur définition impose de bien connaître la PO. Il convient de noter que ce travail est fait une seule fois car ces contraintes sont valides quelque soit le cahier des charges. Des méthodes comme l'AMDEC (Analyse des Modes de Défaillances, de. leurs Effets et de leur Criticité) peuvent être utiles dans cette étape. Dans la suite du papier, le filtre de validation reçoit en entrées, d'une part les ordres envoyés par la commande (appelés événements commandables : Evc) et d'autre part l'état de la partie opérative représenté par les capteurs dont les changements sont appelés événements non commandables (Evnc). Trois types de contraintes de sécurité ont été définis [9]:

- Les contraintes de sécurité statiques (CSS) exprimant des impossibilités physiques et techniques entre des éléments du système.
- Les contraintes de sécurité dynamiques (CSD) représentant l'occurrence d'un événement qui n'est pas compatible avec l'état du système. Deux types de CSD ont été définis en fonction de l'occurrence du type d'événement commandable ou non.
- Les contraintes de sécurité séquentielles (CSSéq).

Tous les états de la PO utiles à la définition des contraintes dynamiques ne sont pas forcément observables au moyen de capteurs. Il est donc parfois nécessaire de reconstruire l'état, au moyen d'un Grafcet par exemple.

Il est à noter que la distinction faite entre les contraintes, nous permet de générer de manière automatique une explication aux erreurs commises.

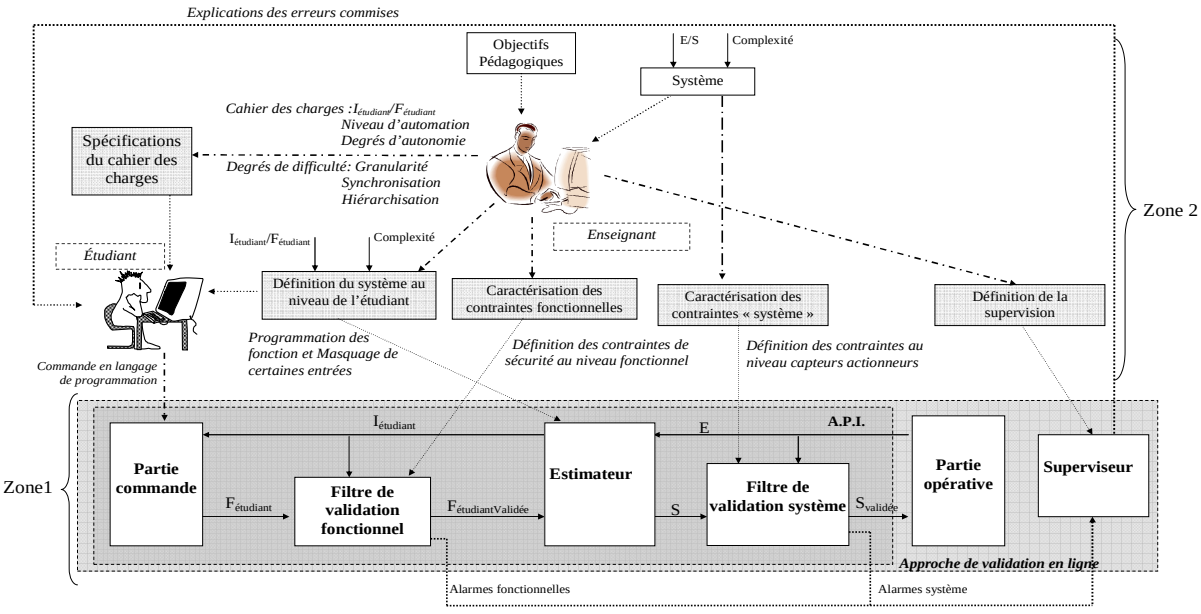


fig. 3. Mise en place de l'approche de validation

4 APPLICATION A LA MACHINE « PRODUCTIS »

4.1 Présentation de la machine Productis

L'Université de Reims Champagne-Ardenne dispose d'une machine de conditionnement de médicaments appelée « Productis ». Celle-ci est conçue pour réaliser le conditionnement de comprimés en flacons de deux tailles par l'intermédiaire de 5 postes : 2 pour la distribution de comprimés par comptage, 1 pour le bouchonnage des gros flacons, 1 pour le bouchonnage des petits flacons et l'évacuation des flacons et 1 dernier manuel pour l'alimentation en palettes. Le système est constitué de 68 entrées et 33 sorties et piloté par 2 API.

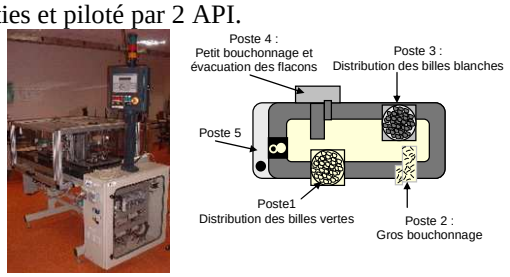


fig. 4. Machine « Productis »

4.2 Identification fonctionnelle

L'identification fonctionnelle du système a conduit à 4 niveaux de décomposition selon un découpage par poste. Le niveau le plus haut correspond à la fonction globale de chaque poste et le niveau le plus bas à l'ensemble des entrées/sorties propres à chaque poste.

4.3 Définition des contraintes de sécurité

Au total 25 contraintes ont été définies. Toutefois, seulement 5 représentent un réel risque de casse au niveau de la machine. Les autres permettent de mettre en évidence des erreurs de programmation des étudiants. A titre d'exemple, les contraintes relatives au poste 2 (gros bouchonnage) sont présentées. Pour bouchonner un flacon, il faut positionner la tête du vérin au dessus du bouchon, descendre, prendre le bouchon par aspiration pour le maintenir, monter, avancer la tête, descendre et souffler le bouchon. Pour le poste 2, les contraintes de sécurité sont donc les suivantes :

- CSS: $AVANCER_2 \wedge RECULER_2 = 0$, $MONTER_2 \wedge DESCENDRE_2 = 0$
- CSD sur un événement non commandable : $AVANCER_2 \wedge \uparrow tête\_avancée_2 = 0$, $RECULER_2 \wedge \uparrow tête\_reculée_2 = 0$, $MONTER_2 \wedge \uparrow tête\_montée_2 = 0$, $DESCENDRE_2 \wedge \uparrow tête\_descendue_2 = 0$
- CSD sur un événement commandable :

$\uparrow AVANCER_2 \wedge tête\_avancée_2 = 0$ (1) $\uparrow AVANCER_2 \wedge tête\_montée_2 = 0$ (2)
$\uparrow RECULER_2 \wedge tête\_reculée_2 = 0$ (3) $\uparrow RECULER_2 \wedge tête\_montée_2 = 0$ (4)
$\uparrow MONTER_2 \wedge tête\_montée_2 = 0$(5) $\uparrow DESCENDRE_2 \wedge tête\_descendue_2 = 0$(6)
$\uparrow PRENDRE_2 \wedge /(tête\_descendue_2 \wedge tête\_reculée_2) = 0$ (7)
$\uparrow LACHER_2 \wedge /(tête\_descendue_2 \wedge tête\_avancée_2) = 0$ (8)

- CSSéq : il n'y a pas de CSSéq sur le poste 2. Par contre sur le poste 4, l'évacuation du flacon est faite par l'intermédiaire d'une pince. Pour éviter la casse du flacon, il faut interdire que la pince descende si elle est fermée. Toutefois, il n'y a pas de capteurs pour déterminer la position de la pince. Il est donc nécessaire de reconstruire cette information pour définir une CSSéq (figure 5)

L'utilisation du filtre est d'une part sécurisante pour l'enseignant et d'autre part permet de réaliser des programmes API permettant d'utiliser la machine « Productis » dans sa globalité pendant des séances de TP de 3 à 4 heures. Dans le cas des étudiants de DUT, l'identification fonctionnelle pour les postes 1 et 3 s'arrête à un niveau élevé, car la structure de commande pour distribuer une bille nous semble trop compliquée pour des automaticiens « débutants ». Nous avons donc défini une fonction « Mettre une bille » avec pour entrées/sorties réduites : *« Présence palette »*, *« Passage d'une bille »* et la sortie *« METTRE UNE BILLE »*. Les étudiants travaillent en mode autonome avec une perception de la « Productis » au travers de 15 entrées et 20 sorties. En revanche, pour les étudiants de Licence Pro et de Master, seul le filtre de sécurité système a été implanté. C'est donc la définition du cahier des charges et des modes de fonctionnement qui va être différent entre les deux enseignements.

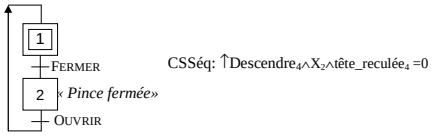
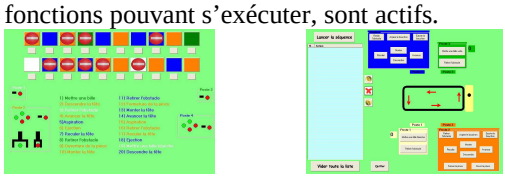


fig. 5. CSSéq pour gérer la ressource commune

4.4 Activité avec des automaticiens « en herbe »

Nous avons exploité les développements réalisés en vue de permettre à des enfants de 9 ans de réaliser leur premier programme API sur la « Productis ». Cette fois-ci, contrairement à ce qui avait été présenté lors du CETSIS 2005 [1], nous voulons non seulement faire découvrir un système automatisé à des enfants mais surtout leur faire programmer le système. En mettant du matériel à la disposition de novices, les problèmes de sécurité qui ont été soulevés dans cet article sont amplifiés. En effet, nous nous adressons à un public qui n'a pas de connaissances en commande de systèmes et qui est donc susceptible de commettre de nombreuses erreurs. Après discussion avec l'institutrice partenaire du projet, il est apparu que l'enfant à cet âge est capable de comprendre le séquencement de fonctions simples. Nous avons donc décidé d'exploiter notre approche dans le cadre d'un mode de fonctionnement non autonome et en évitant toute notion de parallélisme. Les enfants sont bien entendu trop jeunes pour utiliser un éditeur de programmation de type PL7-PRO. Nous avons donc développé deux IHM de supervision permettant d'agir sur le système En accord avec l'institutrice, l'activité avec les enfants se déroule en deux parties. Dans la première, l'enfant a à sa disposition une interface (figure 6.a) avec 20 boutons représentant 20 fonctions du système. Cette activité a pour objectif de permettre à l'enfant de comprendre la fonction cachée derrière chacun des boutons. Pour cela, l'enfant appuie sur un bouton de l'interface et cela provoque le mouvement ou les mouvements correspondants à la fonction sur la machine. Il est à

noter que selon l'état du système, tous les boutons ne sont pas actifs. Seuls les boutons associés à des fonctions pouvant s'exécuter, sont actifs.



a) Mode pas à pas *b) Mode séquence*

fig. 6. Interfaces

La deuxième partie utilise une interface (figure6.b) où l'enfant va pouvoir programmer sa propre séquence de mouvement pour réaliser un flacon de médicaments. Lors du déroulement de la séquence, avant l'exécution de la fonction suivante (envoi d'un ordre), le programme vérifie que la fonction valide l'ensemble des contraintes. Si elles le sont, la fonction est exécutée sur le système et le système de validation passe à la fonction suivante. Sinon, le système de validation indique à l'enfant la ou les contraintes qui ne sont pas respectées afin de l'aider à modifier sa commande (séquences) pour réaliser son flacon.

Supposons par exemple que l'enfant propose la séquence suivante sachant qu'il y a une palette au poste 1, que la tête du poste 2 est rentrée en position montée et l'aspiration du poste 2 n'est pas active.

$\ldots \rightarrow$ *LIBERER*$_1$ $\rightarrow$ *DESCENDRE*$_2$ $\rightarrow$ *PRENDRE*$_2$ $\rightarrow$ *AVANCER*$_2$ $\rightarrow$ *DESCENDRE*$_2$ $\rightarrow$ *LACHER*$_2$ $\rightarrow \ldots$

Lors du déroulement de la séquence, l'arrivée de l'ordre Liberer_palette_poste$_1$ valide l'ensemble des contraintes et il est donc transmis au système qui l'exécute. Il en est de même pour l'envoi des ordres DESCENDRE$_2$ et PRENDRE$_2$. Par contre, lors de l'envoi de AVANCER $_2$, la contrainte (2) n'est plus validée d'où l'arrêt du système. Le système de validation informe l'enfant sur la contrainte qui n'est pas respectée. A lui ensuite de modifier sa séquence de commande.

5 CONCLUSIONS ET PERSPECTIVES

Cet article traite de l'utilisation de parties opératives « réalistes » dans le cadre de l'enseignement de la commande des systèmes automatisés. Cette mise à disposition de matériel soulève deux problèmes : Comment faire piloter un système dans sa globalité alors que le niveau des étudiants diffère ? et comment sécuriser le système ?

- Pour adapter le niveau de difficulté de la commande au niveau de l'apprenant, nous proposons de modifier le niveau d'automatisation sans changer la taille du système. Le principe est de définir un champ d'actions adapté à l'apprenant à partir d'une identification fonctionnelle du système, ce qui permet de garder une vision globale du système.
- Pour sécuriser le système, la conception de deux filtres de validation permet de garantir la sûreté du système. Un « filtre de validation système » permet de valider les ordres envoyés vers la partie opérative. Ce filtre est basé sur des contraintes logiques qui sont classifiées en fonction de leurs types : CSS, CSD et CSSéq. Un second filtre dit « filtre de validation fonctionnel » est construit afin d'adapter la complexité de la commande au niveau de l'apprenant.

Cette approche a été validée avec des étudiants de l'Université de Reims et des enfants de 9 ans, qui ont pu ainsi réaliser rapidement leur premier programme de commande sur un système industriel de packaging. Ces travaux vont permettre l'année prochaine à des étudiants de niveau BAC+1 et BAC+2 d'exploiter sans risque la machine « Productis » dans le cadre de TP d'initiation à la programmation des API au lieu d'utiliser la simulation. De plus, dans le cadre de nos activités de développement de la culture scientifique et technique en région Champagne-Ardenne, la machine « Productis » sera accessible aux écoles primaires via internet afin de permettre à de jeunes enfants de découvrir le monde de l'automatisme. Il est à noter que ces problématiques se retrouvent aussi en milieu industriel, où les différents « acteurs » qui sont amenées à intervenir sur un système de production, ne sont pas tous experts en commande et n'ont pas obligatoirement la connaissance globale du système au niveau capteurs actionneurs. L'approche proposée dans ce papier pourrait être une solution pour sécuriser le système et adapter les champs d'actions au niveau de l'utilisateur.

6 REMERCIEMENT

Ce travail a été réalisé dans le cadre d'un projet financé par la région Champagne Ardenne.

7 BIBLIOGRAPHIE

[1] Riera B., Gellot F., Marangé P., Chemla J-P., Sayed Mouchawed M., "Un projet original en commande et supervision des systèmes automatisés : Des enfants de 5ans au secours d'animaux malades ! ", *CETSIS'05*, Nancy, France, 25-27 octobre 2005

[2] Lind M., Modeling Goals and Functions of Complex Industrial Plant. Applied Artificial Intelligence, Vol8 No.2,April-June 1994

[3] Bossy J.C., Brard P., Faugère P., Merland C., Le Grafcet, sa pratique et ses applications, ed. Casteilla, nouvelle ed. 1997

[4] Emerson E.A., Van Leeuwen D.J. Temporal and modal logic, editor Handbook of the theoretical Computer Sciences, vol. 9, chapter 16, pages 995-1072, 1990

[5] Lampérière S., Lesage J.J, Formal verification of the sequential part of PLC programs, Proc. Of 5th IFAC Wodes, pp 247-254, Ghent, Belgium, August 2000

[6] Behramm G., David A., Larsen K.G., A tutorial on UPPAAL, novembre 2004

[7] Marangé P., Tajer A., Gellot F., Carré-Ménétrier V., "Synthesis of supervised controller based on Boolean constraints and Boolean automata", INCOM'06: 12th IFAC Symposium Information Control Problems in Manufacturing, may 2006, Vol. 1, p299-304, St Etienne, France

[8] Cruette D., Méthodologie de conception des systèmes complexes a événements discrets : application à la conception et à la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière Thèse de doctorat Université de Lille 1991

[9] Marangé P., Gellot F., Riera B., Safety validation of automation systems : Application for teaching of Discrete Event System control, Icinco'07, Angers, 09-12 mai 2007