Ordonnancement d'ateliers à partir de patrons de modélisation basés sur des automates communicants

Pascale Marangé, Alexis Aubry, Jean-François Pétin

ORDONNANCEMENT D'ATELIERS A PARTIR DE PATRONS DE MODELISATION BASES SUR DES AUTOMATES COMMUNICANTS

Pascale Marangé[1,2], Alexis Aubry[1,2] et Jean-François Pétin[1,2]

1 CNRS, CRAN UMR 7039, France
2 Université de Lorraine, CRAN UMR 7039, Boulevard des aiguillettes,
B.P. 70239 F-54506 Vandœuvre-lès-Nancy
{pascale.marange, alexis.aubry, jean-francois.petin}@univ-lorraine.fr

RESUME : *Ce papier propose de montrer comment des patrons de modélisation à base d'automates communicants peuvent être utilisés pour l'ordonnancement d'ateliers de type Job-shop/Flow-shop/Open-shop ou hybride. Les approches classiques de recherche opérationnelle et d'optimisation pour résoudre des problèmes d'ordonnancement nécessitent souvent des prérequis importants et le modèle obtenu est très dépendant de l'application. Suite à des premiers travaux ayant montré la pertinence d'une modélisation par automates communicants et de l'obtention d'un ordonnancement réalisable par recherche d'atteignabilité, l'objectif de ce papier est de présenter comment cette approche de modélisation permet à un utilisateur lambda de construire son modèle sans prérequis particulier. L'autre force de cette modélisation réside dans le fait qu'elle ne nécessite pas d'efforts supplémentaires pour passer d'un type d'atelier à un autre ou pour prendre en compte des modifications dans la structure du problème étudié. Ce papier, après avoir détaillé les patrons de modélisation (machine, opération), présente l'automatisation de l'instanciation de ces modèles à partir d'informations basiques qu'un décideur pourrait connaître. Ceci permet de démontrer que l'approche proposée est générique, simple dans sa modélisation, et évolutive.*

MOTS-CLES : *Ordonnancement, modélisation générique, automate temporisé communicant*

1 INTRODUCTION

L'ordonnancement de la production consiste à définir le cheminement de produits à l'intérieur d'un parc de machines en allouant des ressources pour réaliser les transformations à effectuer, et en définissant les dates de début et de fin de chacune des opérations nécessaires à la fabrication des produits. Cet ordonnancement est généralement réalisé de manière prévisionnelle en considérant un environnement certain et stable. Cependant, pour prendre en compte la forte variabilité des produits et des aléas de fabrications (pannes machines, indisponibilité des opérateurs …), et ainsi augmenter l'adaptabilité des systèmes de production, il est nécessaire de compléter cet ordonnancement prévisionnel par un ou plusieurs ré-ordonnancements réactif. Ce ré-ordonnancement doit exploiter les degrés de flexibilité du système de production reposant sur la redondance fonctionnelle des ressources.

Dans ce contexte, nous avons proposé des premiers travaux qui proposent de définir un ordonnancement admissible pour un atelier de type Job-shop en utilisant une méthode par recherche d'atteignabilité (Marangé et *al.*, 2011). En effet, nous avons montré qu'il était possible, d'une part de modéliser les opérations et les machines par des automates communicants et d'autre part d'utiliser le mécanisme d'appel-réponse pour associer à chaque opération, une machine à un instant donné.

L'objectif de ce papier est de premièrement étendre les modèles présentés dans (Marangé et *al.*, 2011) afin de les rendre génériques puis secondement de montrer que l'utilisation de ces patrons de modélisation permet facilement de modéliser différents types d'atelier. Pour cela, nous organisons le papier de la manière suivante. Dans une première section, le problème d'ordonnancement d'ateliers sera formalisé. Un état de l'art sera présenté pour justifier l'utilisation d'outil du domaine des systèmes à événements discrets (SED) dans la section 3. La section 4 présentera les patrons de modélisation et leur utilisation pour modéliser les différents types d'atelier. Dans la section 5, cette modélisation par patrons est implantée dans un prototype logiciel afin de montrer que les patrons de modélisations peuvent être utilisés de manière transparente pour l'utilisateur. La section 6 permet de discuter de l'approche afin d'évaluer la flexibilité de l'approche proposée par rapport aux méthodes classiques. Enfin, nous concluons et nous donnons des perspectives dans la section 7.

2 FORMALISATION DU PROBLEME D'ORDONNANCEMENT D'ATELIERS

Un problème d'ordonnancement d'ateliers qu'il soit de type Job-shop, Open-shop ou Flow-shop est défini par un ensemble de paramètres caractérisant les produits et l'atelier étudiés et un ensemble de variables de décision à fixer. Nous nous intéressons aux ateliers flexibles qui sont caractérisés par le fait qu'une opération peut être

réalisée sur plusieurs machines et que les machines peuvent réaliser plusieurs opérations.

2.1 Paramètres

Un atelier de production est défini par un ensemble $\mathcal{J}$ de produits j devant être fabriqués sur un ensemble $\mathcal{M}$ de machines m et suivant une gamme de fabrication $\mathcal{O}_j^J$ d'opérations o_{jk}.
L'exécution de l'opération o_{jk} ($k^{\text{ième}}$ opération de la gamme $\mathcal{O}_j^J$ du produit j) requière une machine m qualifiée pour cette opération $\mathcal{O}_j^J$ occupant celle-ci pendant une durée d_{jmk}. Le fait qu'une machine m est qualifiée pour une opération o_{jk} est sue grâce à la fonction Q définie telle que :

$$Q(o_{jk}, m) = \begin{cases} 1 \text{ si la machine } m \text{ est qualifiée pour l'opération } o_{jk} \\ 0 \text{ sinon} \end{cases} \quad (1)$$

Chaque machine m peut subir des périodes d'indisponibilité $u_{ml} = [SD_{ml}, ED_{ml}]$ où SD_{ml} (respectivement ED_{ml}) est la date de début (respectivement la date de fin) de la période d'indisponibilité. Ces périodes d'indisponibilité peuvent être, par exemple, des opérations de maintenance préventive qui sont généralement planifiées par avance.

Pour les ateliers de type Job-shop, les produits ont des gammes différentes *a priori*. Dans le cadre des ateliers de type Open-shop les opérations d'une gamme peuvent être exécutée dans un ordre quelconque. Dans le cadre d'un atelier Flow-shop flexible, un ensemble de stages de fabrication est défini. Chaque stage doit être traversé par tous les produits dans un ordre prédéfini et connu. Chaque stage est composé par un ensemble de machines pouvant réaliser un ensemble d'opérations.

2.2 Variables de décision

Ordonnancer un atelier consiste, pour chaque produit, à (i) affecter une machine à chaque opération de sa gamme, et (ii) à séquencer ces opérations (définir une date de début et de fin à chaque opération), tout en satisfaisant un ensemble de contraintes.

- x_{jkm} est la variable qui permet de définir l'affectation des opérations aux machines

$$x_{jkm} = \begin{cases} 1 \text{ si l'opération } o_{jk} \text{ est affectée à la machine } m \\ 0 \text{ sinon} \end{cases} \quad (2)$$

- y_{jk} est la variable qui permet de fixer la date de début d'une opération o_{jk}
- c_{jk} est la variable qui permet de fixer la date de fin d'une opération o_{jk} et $c_{jk} = y_{jk} + \sum_{M_{jk}} x_{jkm} * d_{jkm}$ (3)

L'ensemble des contraintes qui doivent être satisfaites est :

- (C1) la capacité de la machine: chaque machine ne peut traiter qu'une opération à la fois.
- (C2) la qualification de la machine : une machine ne peut être affectée à une opération que si la machine est qualifiée pour cette opération.
- (C3) non préemption : les opérations ne peuvent être interrompues.
- (C4) affectation unique : une et une seule machine est affectée à chaque opération. En d'autres termes, une opération ne peut être réalisée sur deux machines différentes.
- (C5) séquencement : suivant le type d'atelier, des contraintes sur le séquencement des opérations peuvent exister :
  - Cas des ateliers Job-shop et Flow-shop : l'ordre d'exécution des opérations d'un produit j est fixée par sa gamme, c'est à dire que si $k_1 < k_2$ alors l'opération o_{jk_1} doit être exécutée avant l'opération o_{jk_2}
  - Cas d'un atelier Open-shop : l'ordre d'exécution des opérations d'un produit j n'est pas défini, mais les opérations ne peuvent pas s'exécuter au même instant, c'est à dire que si $k_1 \neq k_2$ alors l'opération o_{jk_1} doit être exécutée avant l'opération o_{jk_2} ou inversement.

2.3 Optimalité, robustesse et réactivité

Les problèmes d'ordonnancement sont souvent associés à des objectifs de résolution qui consiste à trouver un ordonnancement admissible dont les performances sont optimales vis-à-vis d'un critère. Il existe quasiment autant de critères que de cas d'application même si une liste de critères classiques pourrait être proposée. Le plus répandu est certainement le critère qui consiste à évaluer le temps de réalisation de l'ensemble des produits. Un ordonnancement optimal est donc un ordonnancement admissible qui optimise un critère défini *a priori*.
Appréhender un problème d'ordonnancement consiste souvent à considérer que l'environnement de production est certain (les données sont connues et certaines : demande, temps de production…) et stable (il n'y a pas d'aléas de production : les machines ne subissent pas de pannes, il n'y a pas de rebut…). Dans ce contexte, l'ordonnancement optimal, calculé selon ces hypothèses optimistes, risque de voir ses performances détériorées lors de sa mise en œuvre en pratique dans un contexte qui ne peut correspondre exactement à l'environnement théorique. Pour faire face à ces perturbations (incertitudes des données et aléas de production), (Aubry, 2007) présente trois approches :

- l'approche purement proactive, permet de calculer, hors ligne (avant production), un ou plusieurs ordonnancements avec des propriétés de robustesse. Pour cela, les perturbations attendues sont introduites dans le modèle : les variables incertaines du problème suivent des lois de probabilité (modèle stochastique) ou appartiennent à des intervalles (continus, discrets ou flous). Cela nécessite une connaissance importante sur les perturbations, ce qui n'est pas toujours facile en pratique. On privilégie, dans cette approche, des performances globales (notion de robustesse) sur un ensemble de données (loi de probabilité, intervalle) plutôt que la seule performance locale (notion d'optimalité) sur des données statiques fixées (une seule valeur à chaque variable du problème).
- l'approche purement réactive calcule un ordonnancement en ligne (pendant la production) à partir des connaissances disponibles.
    - l'approche proactive/réactive consiste à coupler les deux précédentes approches. Parmi les ordonnancements calculés hors ligne, on sélectionne en ligne la meilleure solution par rapport aux données réelles. Si des perturbations sont survenues et ne permettent pas d'appliquer une des solutions calculées hors ligne, il y a deux possibilités : réparer la solution courante ou calculer une nouvelle solution (ce qui revient à appliquer une approche purement en ligne)

Nous nous plaçons dans un contexte purement en ligne où l'optimalité n'est pas pertinente. En effet, trouver un ordonnancement en temps réel est plus important car il faut continuer à produire.

3 ETATS DE L'ART

3.1 Approches classiques d'ordonnancement

L'ordonnancement d'ateliers flexibles de type Job-shop, Flow-shop ou Open-shop est un problème communément admis comme étant NP-difficile (Garey et *al.*, 1976). Ce qui signifie qu'il n'existe pas de méthode « rapide » pour résoudre le problème de manière optimale. La résolution est traditionnellement abordée par des méthodes issues des outils de la recherche opérationnelle. En raison de sa complexité, la plupart des approches de résolution sont basées sur des métaheuristiques ou des méthodes de résolution approchée. Les méthodes de résolution optimale sont principalement basées sur Programmes Linéaires à Variables Mixtes (MILP en anglais). Parmi les travaux existants, nous pouvons citer les travaux de (Vilcot and Billaut, 2011 ; Rabiee et *al.*, 2012) pour le problème d'ordonnancement Job-shop flexible ou ceux de (Gao et *al.*, 2006 ; Zribi et *al.*, 2008 ; Rajkumar et *al.*, 2010) pour la planification Job-shop flexible en tenant compte de contraintes d'indisponibilité pour maintenance. L'ensemble de ces approches est généralement dédié à un problème spécifique. Pour les méta-heuristiques, elles nécessitent souvent un ensemble initial de solutions admissibles dont la recherche est en soi un problème non-trivial et constitue une limite pour l'application de ces méthodes.

Approche de résolution	Obtention de la solution		Modélisation du problème		
	Optimalité	Efficience	Complexité de modélisation	Généricité	Evolutivité
MILP	+	-	-	-	-
Méta-heuristique	-	+	-	-	-
Heuristique dédiée	-	+	-	-	-

Tableau 1 : Proposition d'analyse pour les méthodes classiques de recherche opérationnelle

Le tableau 1 permet d'évaluer ces approches classiques selon deux points de vue et cinq critères :

- Obtention de la solution
    - Optimalité : est-ce que l'approche permet de trouver une solution optimale ?
    - Efficacité : est-ce que l'approche de résolution permet de résoudre le problème en un temps raisonnable ?
- Modélisation du problème
    - Complexité de modélisation : est-ce que la modélisation du problème par l'utilisateur nécessite des prérequis importants ?
    - Généricité des modèles : est-ce que les modèles obtenus sont généralisables par rapport aux différents types d'ateliers ?
    - Evolutivité : est-ce que les modèles obtenus peuvent facilement être modifiés pour ajouter des spécificités ou des nouvelles contraintes?

Nous avons vu dans la section précédente que nous nous intéressons à l'obtention d'un ordonnancement en temps réel où le caractère optimal importe peu mais où le fait d'obtenir une solution rapidement est important. De plus, ces méthodes sont avant tout destinées à des décideurs n'ayant pas toujours de compétences particulières en recherche opérationnelle, en mathématiques et en algorithme. C'est donc ce deuxième point de vue qui nous intéresse plus particulièrement et on voit que les approches classiques ne satisfont pas ou peu ce point de vue.

3.2 Approches par automates temporisés

Face à ces approches classiques, les méthodes basées sur les systèmes à événements discrets (SED) émergent pour modéliser et pour résoudre des problèmes d'ordonnancement (Kobetski and Fabian, 2009). L'idée de base est de modéliser le problème d'ordonnancement comme un modèle d'états-transitions qui représentent la séquence des opérations et la disponibilité des machines et d'utiliser la recherche d'atteignabilité pour trouver un chemin possible permettant d'atteindre un état souhaité (dans notre cas, l'état où tous les produits sont terminés). La trace allant de l'état initial à l'état souhaité fournit un ordonnancement admissible.

L'efficacité de la théorie des SED pour la modélisation et la résolution d'un problème d'ordonnancement a été explorée dans plusieurs travaux:

- La résolution du problème d'ordonnancement : la recherche d'atteignabilié est un problème largement abordé dans les SED et plusieurs approches sont mises en œuvre dans les outils de model-checking (Clarke and Emerson, 1981). Pour les problème d'ordonnancement, plusieurs algorithmes ont été proposées (Abdeddaim and Maler, 2001) et la performance des approches SED utilisant des automates temporisés (TA) ont été démontrés comme satisfaisants en terme de temps de calcul et en terme de qualité de la solution trouvée (Behrmann et *al.*, 2005 ; Panek et *al.*, 2006, rasmussen et *al.*, 2004).
- problème de modélisation : le principal avantage des modèles SED repose sur la modularité et la représentation graphique ainsi que leurs capacités natives pour modéliser l'évolution dynamique d'un système de production. La modularité permet de modéliser le problème d'ordonnancement comme une combinaison de modèles de produits et de machines. Par exemple, l'approche développée par (Behrmann et al., 2004) et (Subbiah and Engell, 2010) introduit plusieurs modèles pour les gammes opérationnelles (c'est à dire une séquence d'opérations allouée aux machines) et un modèle simplifié des machines à deux états : {occupé, libre}. Cependant, cette approche ne tient pas compte de la flexibilité du Job-shop dans la mesure où les opérations sont associées à une et une seule machine.

Ces travaux antérieurs permettent de conclure que les approches fondées sur les automates sont de bonnes alternatives pour résoudre les problèmes d'ordonnancement. Leurs avantages reposent notamment sur la capacité intrinsèque des modèles à représenter la dynamique de l'environnement productif : états des machines… Ces modèles sont de plus paramétrables et les temps de calcul pour trouver l'état atteignable sont raisonnables.
Cependant, la plupart des applications existantes concernent le problème de Job-shop classique avec des gammes opérationnelles statiques sans vraiment tirer avantage de la modularité des méthodes basées sur les SED. L'objectif de ce papier est justement de montrer comment les automates communicants permettent de modéliser efficacement et facilement tous les types d'atelier classiques et de résoudre leur ordonnancement.

4 PATRONS DE MODELISATION

4.1 Préliminaires

4.1.1 Rappel de l'approche

Nos travaux précédents (Marangé et *al.*, 2011) ont montré l'utilisation du model-checking par recherche d'atteignabilité pour résoudre des problème d'ordonnancement dans le cadre d'ateliers flexibles de type Job-shop. Ces travaux proposaient une première modélisation du produit et des machines par des automates temporisés et plus particulièrement par des automates communicants.

- Le modèle du produit définissait la gamme logique à exécuter sans tenir compte de toute affectation sur une machine dédiée,
- Le modèle de la machine décrit les états de la machine, ainsi que les indisponibilités et les contraintes devant être satisfaites afin d'accepter l'exécution d'une opération,
- La synchronisation entre les modèles de la machine et du produit est soutenu par un mécanisme d'appel / réponse (Alur and Dill , 1994 ; Lemattre et *al.*, 2011) .

Cette approche est basée sur la même philosophie que le concept du contrôle par le produit (Pétin et *al.*, 2007) , où le produit devient un acteur lors de la production pour le choix de la ressource pouvant exécuter une opération.
Une solution d'ordonnancement admissible (qui satisfait l'ensemble des contraintes C1 à C5) peut être trouvé en vérifiant une propriété d'atteignabilité P : à partir de l'état initial, est ce qu'il est possible d'atteindre la localité marquée de tous les automates de produit ? L'exemple qui satisfait la propriété P et qui est retourné par le model-checker est un ordonnancement admissible.
Dans ce papier, nous proposons une modification du modèle de produit afin de le rendre indépendant du type d'atelier. Le point fort de cette modélisation par patrons est de permettre de rendre automatisable l'instanciation des modèles et donc de rendre invisible la couche modélisation à l'utilisateur qui devra uniquement déclarer l'atelier.

4.1.2 Formalisme

Les automates communicants sont une sous-classe de la classe des Automates temporisés définis par (Alur et Dill, 1994) qui partagent des variables et qui sont synchronisés par des transitions étiquetées. Un automate communicant A est un N-tuple $A = (S, V, L, T, S_m;\ q_0, v_0)$, où :

- S est l'ensemble des localités;
- V est l'ensemble de variables entières fini;
- L est l'ensemble des étiquettes de synchronisation qui peut être décomposée en trois ensembles distincts : L_i est l'ensemble des étiquettes de réception qui sont notées *étiquette? ;* L_o est l'ensemble des étiquettes d'émission qui sont notées *étiquette! ;* L_l est l'ensemble des étiquettes locales.
- T est l'ensemble des transitions $(s;\ l;\ g;\ m;\ s') \in S \times L \times M \times S \times G$ où G est l'ensemble des gardes (conditions sur les va-

riables de V) et M est l'ensemble des mises à jour des évaluations de variables; l, m et g sont facultatifs (une transition peut contenir aucune étiquette de synchronisation et aucune mise à jour par exemple), mais une transition doit contenir au moins une étiquette ou une garde;

- $S_m \subseteq S$ est l'ensemble des localités marquées;
- $S_0 \in S$ est la localité initiale;
- $V_0 \in V$ est la valeur initiale des variables.

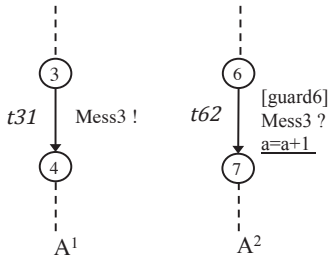


Figure 1 : Exemple d'évolution d'un automate communicant

Les automates peuvent évoluer avec les synchronisations, par l'utilisation des étiquettes. Deux transitions t_k^γ et t_m^β de deux automates A^γ et A^β avec t_k^γ contenant l'étiquette d'émission $l_k^\gamma \in L^\gamma$, notée $l_k^\gamma!$ et t_k^β contenant l'étiquette d'émission $l_k^\beta \in L^\beta$, notée $l_k^\beta?$ sachant que $l_k^\gamma = l_k^\beta$, sont franchies simultanément, à condition que les gardes de ces transitions soient satisfaites. Un état est défini par un couple (s, v) où $s \in S$ et $v \in V$. La Figure 1 montre un exemple d'évolution d'automates temporisés : à l'instant t, l'automate A^1 est dans la localité *3* et l'automate A^2 est dans la localité *6*. Lorsque la transition t_{31} est franchie, le message étiqueté *mess3* est émis et est simultanément reçu par la transition t_{62} qui évolue et incrémente la variable a.
Les automates communicants ne donnent pas une description explicite du temps; le concept d'horloges et de durée sera donc modélisé par des compteurs entiers, afin de pouvoir définir les dates de début et de fin des opérations dans l'ordonnancement. Conventionnellement, les notations sont les suivants: les localités initiales sont indiquées par une transition source, les localités marquées sont indiquées par un double cercle, les noms de localité sont en gras, les noms de message sont en italique et suivis par le symbole! (Resp.?) pour l'émission d'un message (Resp ! pour la réception d'un message), les variables mises à jour sont soulignées, et les gardes sont désignées par des crochets.

4.2 Patron de modélisation de machine

Les machines sont caractérisées par les opérations qu'elles peuvent exécuter, par les durées des opérations, et par une variable d'état permettant de connaître la disponibilité de celle-ci (Ce statut est mémorisé dans la planification des périodes d'indisponibilités). Le modèle de machine interagit avec les modèles de d'opération (décrit dans la partie suivante) par la réception de messages (*requested ?)* et l'émission de messages (*Accepted* ou *Rejeted ou Executed*).

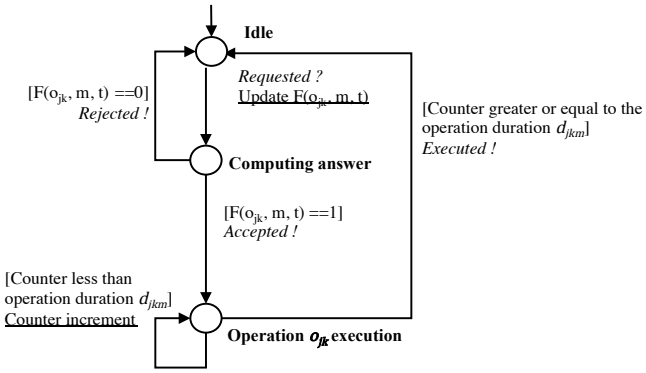


Figure 2 : Patron de modélisation de machine (Marangé et al., 2011)

Ce modèle est composé des localités suivantes (figure 2):

- Dans la localité initiale *Idle*, la machine est inactive et en attente d'une requête du modèle d'opération. Une fois qu'une requête est reçue, le modèle β^j passe dans la localité *Computing answer* où la fonction F sera mise à jour pour déterminer si la machine est disponible et qualifiée pour faire l'opération.
  - La machine ne peut pas accepter la demande si elle est déjà en train d'exécuter une autre opération; cette condition est toujours vraie du fait de la structure du modèle
  - La machine accepte la requête en fonction de la valeur de la fonction *F* vérifiant la qualification de la machine pour l'opération requérante et la disponibilité de la machine pour la durée de l'opération à partir de l'instant t :

$$F(o_{jk}, m, t) = \begin{cases} 1 \; si \; (Q(o_{jk}, m) = 1) \wedge (\forall u_{ml} \in U_m, [t, t + d_{jkm}] \cap u_{ml} = \emptyset) \\ 0 \; sinon \end{cases} \quad (4)$$

- Depuis la localité *Computing answer* si la garde contenant la fonction F est vraie, alors un message *accepted !* est envoyé au modèle d'opération demandeur α^iet le modèle de machine β^j passe dans la localité *Operation o_{jk} execution*. Par contre si la garde n'est pas vraie, un message *rejected !* est envoyé au modèle d'opération demandeur et le modèle de machine revient à la localité initiale.
- Depuis la localité *Operation o_{jk} execution*, un compteur est incrémenté pour faire évoluer le temps. Dès que le temps écoulé depuis le début de l'exécution atteint la valeur de la durée d'exécution d_{jkm}, le modèle de machine β^j retourne à la localité initiale. Une fois, la durée

écoulée, un message *Executed !* prévient le modèle d'opération de la fin de l'exécution

Ce modèle permet de prendre en compte les contraintes : (i) C1 qui est vérifiée par le fait que le modèle de machine doit être dans la localité initiale pour recevoir des requêtes d'exécution d'opération et (ii) C3 et C4 qui sont vérifiées par la garde contenant la fonction F

Ce modèle est générique dans la mesure où les différents types d'atelier imposent des contraintes sur le flux des opérations et non pas sur les machines. Les machines, qu'elles appartiennent à un atelier de type Job-shop, Flow-shop ou Open-shop ont le même comportement et peuvent être modélisées par le même patron.

4.3 Patron de modélisation d'une opération

Le modèle d'opération est sensiblement différent du modèle proposé dans (Marangé et *al.*, 2011) afin qu'il soit générique par rapport au type d'atelier considéré. Nous proposons un patron de modélisation pour chaque opération o_{jk} du produit *j*. Ce patron est donné en (figure 3) :

- Dans la localité *Waiting for opération* o_{jk}*to be performed*, le produit est en attente de l'exécution d'une opération o_{jk}. Si l'opération est exécutable, c'est-à-dire si la garde « *Executable_operation_*o_{jk}» (Marangé et *al.*, 2009) est satisfaite alors l'évolution depuis cette localité correspond à l'émission de la requête « *Requested* » demandant à une machine d'exécuter l'opération o_{jk}. La garde E*xecutable_operation_*o_{jk} permet d'une part de vérifier les contraintes de précédence (notée *Precedence_constraint_*o_{jk}) de la gamme et d'autre part de vérifier l'hypothèse qu'une seule opération n'est exécutée à la fois sur le produit. La contrainte de précédence dépend de l'état des autres opérations de la gamme, et une variable booléenne *Product_available_j* permet de modéliser la disponibilité du produit à subir une opération :

$$\text{Executable\_operation\_o}_{jk} = Product\_available\_j \wedge precedence\_constraint\_o_{jk} \quad (4)$$

- Une fois cette requête émise, le modèle est en attente dans la deuxième localité *Waiting for a machine m answer* d'une réponse du modèle de machine qui peut être :
  - Un refus si la machine demandée ne peut pas exécuter l'opération ; dans ce cas le patron revient à la localité de départ du premier dans lequel une nouvelle demande peut être faite,
- Une acceptation si la machine à la qualification et est disponible pour l'opération o_{jk} ; la demande d'allocation (x_{jkm}) est ensuite mis à jour avec la machine utilisée *m* et la date de début de l'opération (y_{jk} = Tcurrent) est enregistrée ; la motif atteint alors la troisième localité *Waiting for the end of opération* o_{jk},
- Dans la troisième localité, le modèle d'opération est en attente de l'achèvement de l'opération indiqué par la réception du message *Executed ?*,
- Dès que cet accusé de réception est reçu, le modèle atteint la dernière localité *End of opération* o_{jk}, la date de fin de l'opération o_{jk} est mis à jour avec $c_{jk} = y_{jk} + d_{jkm}$ et la variable représentant que l'opération est exécutée, est mise à jour.

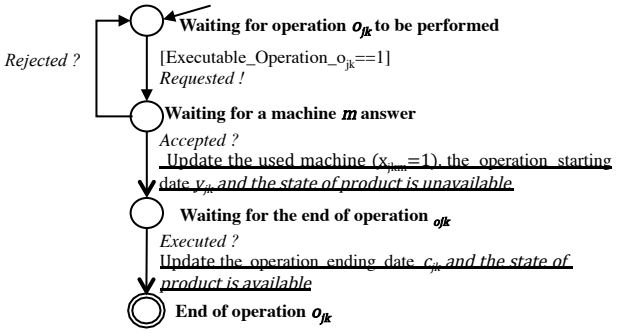


Figure 3 : Patron de modélisation d'une opération o_{jk}

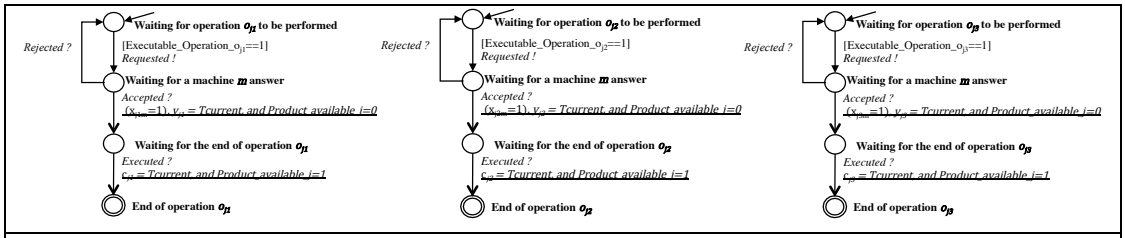
A partir de ce patron, la modélisation des différents types d'atelier se fait par l'instanciation de la garde représentant les contraintes de précédence. Soit une opération o_{jk}, la contrainte sera définie par :

- Pour un atelier Job/Flow-shop, la vérification que l'opération o_{jk-1} a été exécutée
- Pour l'atelier Open-shop, toujours vraie, car il n'y a pas de contrainte de précédence et la vérification qu'une opération ne s'exécute qu'une seulement fois se fait par la structure.

L'utilisation de ce patron pour modéliser les gammes des produits, modifie la propriété utilisée dans le model-checking, c'est à dire qu'on cherche un chemin pour lequel toutes les localités *End of opération* o_{jk} sont vraies. Ce modèle permet de prendre en compte la contrainte C5 par l'utilisation de la garde vérifiant les contraintes de précédence. La figure 4 représente l'instanciation d'une opération en fonction du type d'atelier. Cet exemple montre la possiblité d'instancier des gammes mixant plusieurs types d'atelier.

Pour calculer un ordonnancement admissible d'un atelier donné, il faut :

- Instancier l'ensemble d'automates $NA = \{\alpha^1, \dots, \alpha^O, \beta^1, \dots \beta^M\}$ incluant :
  - O instance du patron d'opération α^i
  - M instances du patron de machine β^m
- Vérifier la propriété d'atteignabilité de l'état fin de chaque opération, par un model-checker (dans notre cas, nous utilisons UppAal (Behrmann et *al.*, 2004))



Atelier en Job-shop ou Flow-shop (cas avec precedence entre operations)			
o_{j1} → o_{j2} → o_{j3}	Precedence constraint o_{j1}	Precedence constraint o_{j2}	Precedence constraint o_{j3}
	∅	[oj1_executed==1]	[oj2_executed==1]
Atelier en Open-shop (cas sans précédence entre opérations)			
o_{j2}, o_{j1}, o_{j3}	Precedence constraint o_{j1}	Precedence constraint o_{j2}	Precedence constraint o_{j3}
	∅	∅	∅
Atelier hybride 1			
o_{j1} → (o_{j2}, o_{j3})	Precedence constraint o_{j1}	Precedence constraint o_{j2}	Precedence constraint o_{j3}
	∅	[oj1_executed==1]	[oj1_executed==1]
Atelier hybride 2			
(o_{j2}, o_{j3}) → o_{j1}	Precedence constraint o_{j1}	Precedence constraint o_{j2}	Precedence constraint o_{j3}
	[oj2_executed==1 ∧ oj3_executed==1]	∅	∅

Figure 4 : Règles d'instanciation du patron d'opération

5 INTERFACE LOGICIELLE D'INSTANCIATION ET DE RESOLUTION

Nous avons montré dans la section précédente que les modèles étaient suffisamment génériques pour modéliser tout type d'atelier. Cependant, il est important pour que cette approche soit pleinement opérationnelle, qu'un décideur puisse l'utiliser sans prérequis particulier sur les outils SED. Nous avons donc proposé un premier prototype logiciel dont le but est de fournir une interface à l'utilisateur qui rende transparent les modèles automates.

Ce prototype est supporté par i) une IHM qui permet à l'utilisateur de rentrer les informations de l'atelier, d'appeler le model-checker pour calculer un ordonnancement admissible et de visualiser cet ordonnancement dans un diagramme de Gantt, ii) une base de données qui permet de sauvegarder les informations de l'atelier rentrées par l'utilisateur, iii) le model-checker (pour nous UppAal).

Le modèle de la base de données et une instanciation simple sont présentés respectivement dans la figure 5 et la figure 6. Les capacités d'un atelier sont représentées par un ensemble d'opérations réalisables ((1) classe Operation) et un ensemble de machines ((2) classe Machine). Les machines possèdent une ou plusieurs qualifications ((3) classe Qualification) pour des opérations ((4) association entre Qualification et Operation). De plus chaque machine peut subir plusieurs périodes d'indisponibilité ((5) classe PeriodeIndisponibilite).

L'atelier doit fabriquer plusieurs produits ((6) classe Produit). Chaque produit peut être fabriqué selon une ou plusieurs gammes ((7) classe Gamme). Seule une gamme est active : ceci est précisé grâce à l'attribut Active de la classe gamme. Une gamme est composée d'une ou plusieurs opérations ((8) classe OperationGamme) choisies dans la liste des opérations disponibles ((9) association entre OperationGamme et Operation). Les opérations de la gamme peuvent avoir des contraintes de précédence ((10) association de la classe OperationGamme vers elle-même).

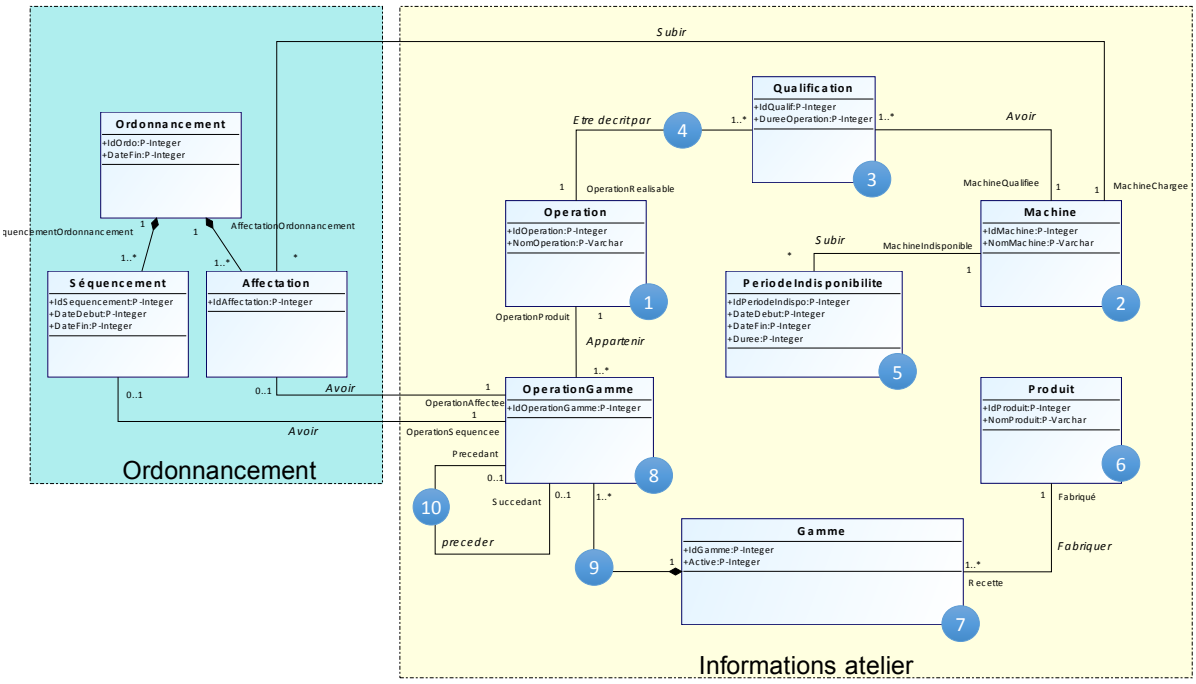


Figure 5 : Modèle sémantique de la base de données

Les données liées à l'ordonnancement résultat sont sauvegardées dans Ordonnancement, Sequencement et Affectation. Chaque OperationGamme est affectée à une Machine via la classe Affectation et a une DateDebut et une DateFin sauvegardées dans Sequencement. Un Ordonnancement est composé de plusieurs Sequencement (un par OperationGamme) et Affectation (une par OperationGamme).

La génération des automates temporisés, peut ensuite se déduire directement de la base de données sans que l'utilisateur n'ait besoin de participer :

- Une instance du patron de modélisation de machine est construite pour chaque instance de la classe Machine et les paramètres de sa fonction **F** sont construits à partir des instances de la classe Qualification et de la classe PeriodeIndisponibilite associées à la machine (cf. section 0).
- Une instance du patron de modélisation d'opération est construite pour chaque instance de la classe OperationGamme et la garde définissant la contrainte de précédence (Precedence constraint) est définie à partir des contraintes de précédence (attribut IdOperationGammePrec) en suivant les règles définies dans la figure 4 de la section 4.3.

La figure 6 permet également de visualiser les automates générés à partir de l'instanciation présentée.

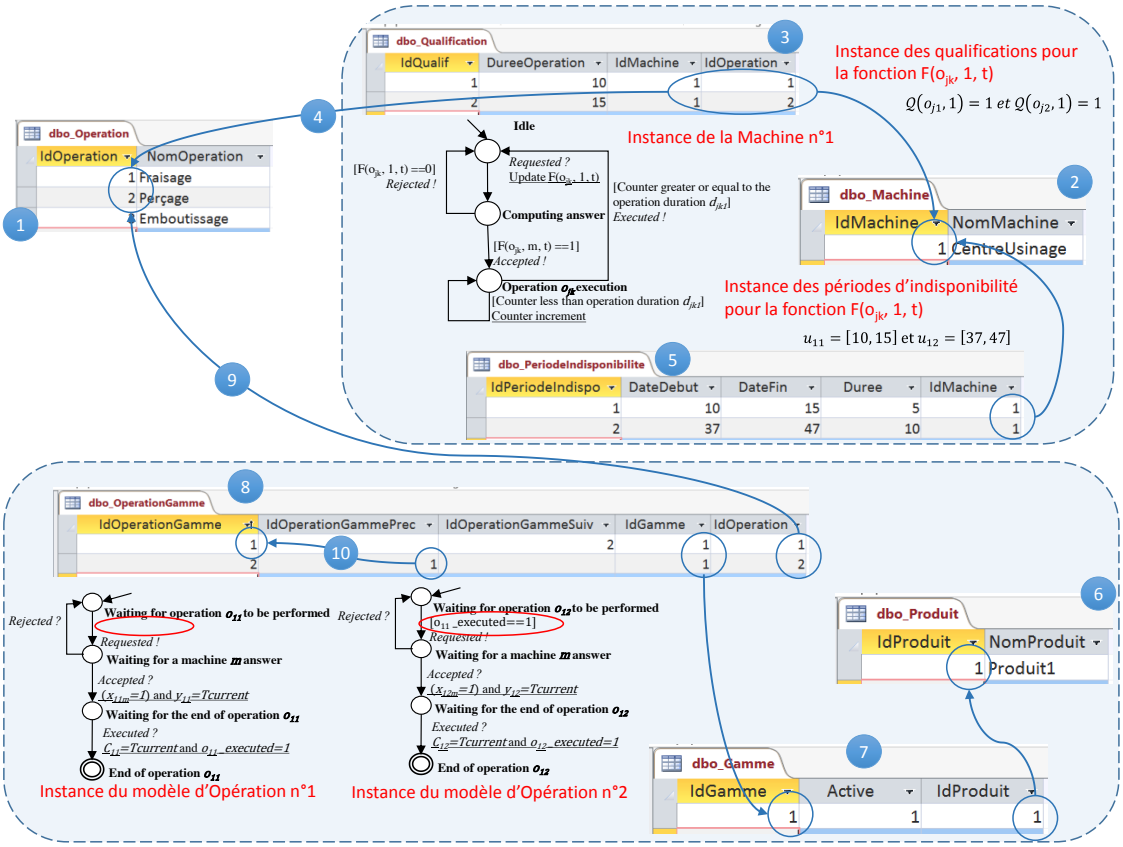


Figure 6 : Exemple basique d'instanciation d'un atelier dans la base de données

6 DISCUSSION

La modélisation proposée dans ce papier permet de :

- Augmenter la généricité : Le modèle de machine est directement instanciable et le modèle d'opération utilise une garde pour définir les contraintes de précédence de la gamme. Ceci est possible pour tous les types d'atelier. Ces modèles sont suffisamment génériques pour que l'instanciation ait pu être automatisée dans un logiciel. De plus les informations demandées à l'utilisateur ne sont aucunement liées au type d'atelier et restent à un niveau général de modélisation de l'atelier (machines, opérations, gammes, produits).
- Diminuer la complexité de modélisation : les patrons proposés dans ce papier sont suffisamment génériques pour qu'une interface graphique ait pu être proposée aux utilisateurs afin de rendre transparent l'utilisation des automates communicants. Cela ne nécessite aucunement de prérequis sur les outils SED. Seules des informations sur l'atelier sont demandées à l'utilisateur et les modèles automates sont générés automatiquement.
- Augmenter l'évolutivité : le prototype logiciel proposé permet à un utilisateur d'ajouter/modifier/supprimer des contraintes (précédence entre opérations, qualifications…) ou des machines, des produits, des opérations disponibles sans avoir à se soucier des modèles sous-jacents puisque ceux-ci peuvent être regénérés à besoin de manière transparente.

De plus, d'autres travaux ont montré que le temps de calcul d'un ordonnancement admissible est raisonnable par model checking sur des automates communicants. Ce qui fait de cette approche une approche efficiente. Le tableau 2 permet de repositionner notre approche dans l'ensemble des approches d'ordonnancement.

Tableau 2 : Positionnement de notre approche par rapport aux approches classiques

Approche de résolution	Obtention de la solution		Modélisation du problème		
	Optimalité	Efficience	Complexité de modélisation	Généricité	Evolutivité
MILP	+	-	-	-	-
Méta-heuristique	-	+	-	-	-
Heuristique dédiée	-	+	-	-	-
Automates Communicants (Marangé et al. MOSIM 2016)	-	+	+	+	+

7 CONCLUSION

Dans ce papier, nous avons proposé une modélisation générique par automates communicants pour résoudre des problèmes d'ordonnancement d'atelier de type Open-shop/Job-shop/Flow-shop. En effet, nous avons montré qu'il était possible de modéliser les machines et les produits d'un atelier sans tenir compte de l'atelier étudié. Cette généricité des modèles nous a permis de concevoir une IHM demandant à l'utilisateur des informations d'un niveau général.

Actuellement, cette approche et l'IHM ont été testées sur des exemples académiques et doivent être éprouvées dans un cadre industriel. L'utilisation dans le domaine industriel nous demandera d'être intégrable avec des ERP et de prendre en compte des ordres de fabrication (OFs), ainsi que des règles de lotissement … Enfin, il serait intéressant d'étendre l'approche de modélisation et de résolution à d'autres problèmes classiques d'ordonnancement ou de recherche opérationnelle.

REFERENCES

Abdeddaim F., Maler O (2001) Job-shop scheduling using timed automata In: Computer aided verification, Springer, pp 478-492

Alur R, Dill D (1994) A theory of timed automata. Theoretical Computer Science 126(2):183-235

Aubry A. Optimisation pour la configuration robuste de systèmes de production de biens et de services, thèse de l' Institut National Polytechnique de Grenoble - INPG, 2007. French

Behrmann G, Brinksma E, M MH, Mader A (2005) Production scheduling by reachability analysis - a case study. In: International Parallel and Distributed Processing Symposium, 3, pp 140-147

Behrmann G, David A, Larsen K (2004) A tutorial on uppaal. In: Proc. Formal Methods for the Design of Real-Time Systems, Springer, no. 3185 in Lecture Notes in Computer Science, pp 200-236

Clarke E, Emerson E (1981) Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proc. Logics of Programs Workshop, Springer, no. 131 in Lecture Notes in Computer Science, pp 52-71

Gao J, Gen M, Sun L (2006) Scheduling jobs and maintenances in flexible Job-shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing 17(4):493-507

Garey M, Johmson D, Sethi R (1976) The complexity of flow shop and Job-shop scheduling. Mathematics of Operational 1(2):117-129

Kobetski A, Fabian M (2009) Time-optimal coordination of flexible manufacturing systems using deterministic finite automata and mixed integer linear programming. Journal of Discrete-Event Dynamic Systems: Theory and Applications 19(3):287-315

Lemattre T, Denis B, Faure J, Pétin J, Salaun P (2011) Designing operational control architectures of critical systems by reachability analysis. In: IEEE 7th International Conference on Automation Science and Engineering

Marangé P., Gouyon D., Pétin J-F., Riera B., Verification of functional constraints for safe product driven control 2nd IFAC Workshop on Dependable Control of Discrete System, DCDS'09, Jun 2009, Bari, Italy. pp.315-320, 2009

Marangé P., Pétin J-F., Manceaux A., Gouyon D., Contribution à la reconfiguration des systèmes de production : ordonnancement par recherche d'atteignabilité Journal Européen des Systèmes Automatisés, Hermès, 2011, 45 (1/3), pp.45-60.

Panek S, Engell S, Stursberg O (2006) Scheduling and planning with timed automata. In: 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, Elsevier, pp 1973-1978

Pétin J, Gouyon D, Morel G (2007) Supervisory synthesis for product-driven automation and its application to a flexible assembly cell. Control Engineering Practice 15(5):595-614

Rabiee M, Zandieh M, Ramezani P (2012) Bi-objective partial flexible Job-shop scheduling problem: Nsga-ii, nrga, moga and paes approaches. International Journal of Production Research 50(24):7327-7342

Rajkumar M, Asokan P, Vamsikrishna V (2010) A grasp algorithm for flexible Job-shop scheduling with maintenance constraints. International Journal of Production Research 48(22):6821-6836

Rasmussen JI, Larsen KG, Subramani K (2004) Resource-optimal scheduling using priced timed automata. In: Tools and Algorithms for the Construction and Analysis of Systems, Springer, pp 220-235

Subbiah S, Engell S (2010) Short-term scheduling of multi-product batch plants with sequence-dependent changeovers using timed automata models. In: 20th European Symposium on Computer Aided Process Engineering, Elsevier, 28, pp 1201-1206

Vilcot G, Billaut J (2011) A tabu search algorithm for solving a multicriteria flexible Job-shop scheduling problem. International Journal of Production Research 49(23):6963-6980

Zribi N, Kamel AE, Borne P (2008) Minimizing the makespan for the mpm Job-shop with availability constraints. International Journal of Production Economics 112(1):151-160