

Génération automatique de code de simulation via l'information de localisation des produits

ANDRÉS VÉJAR, PATRICK CHARPENTIER

Centre de Recherche en Automatique de Nancy, CRAN (CNRS UMR 7039 - Nancy-Université)
Faculté des Sciences et Techniques, BP 70239, 54506 Vandœuvre-lès-Nancy Cedex, France
andres.vejar@cran.uhp-nancy.fr, patrick.charpentier@cran.uhp-nancy.fr

Résumé - Cette communication propose une méthode originale de génération d'un code de simulation pour des systèmes à événements discrets. Cette méthode n'utilise que l'information de localisation des produits pendant le fonctionnement du système. A partir de ce flux d'information (identifiant produit, localisation, temps) l'algorithme proposé génère un modèle de simulation de type file d'attente.

Abstract - This article proposes an original method for simulation code generation in discrete event systems. This method uses the product location information in the running system. The information flux (product *id*, location, time) is the starting point for the algorithm to generate a queuing network simulation model.

Mots clés - Flux, Localisation, Produit, Simulation, Modélisation.

Keywords - Flux, Location, Product, Simulation, Modeling.

1 INTRODUCTION

Depuis quelques années maintenant, de nouvelles visions du produit manufacturé tendent à lui conférer des capacités de communication et des capacités sensibles dans le cadre du paradigme de produit intelligent [Karkkainen *et al.*, 2003]. La part informationnelle liée à chaque produit est donc alimentée soit par l'environnement matériel direct du produit physique ou soit par le biais de sa propre instrumentation (MEMS, GPS, ...). Les échanges informationnels entre le produit et son environnement peuvent s'effectuer :

1. à certains points de synchronisation (emplacements des lecteurs R/W, technologies RFID, code-barre, ...)
2. de façon quasi-continue (réseaux sans fil de type Wifi, Zigbee, Bluetooth, ...)

Ces technologies de communication peuvent elles même contribuer à la localisation du produit dans son environnement en complément à l'instrumentation embarquée. Elles font l'objet de nombreux travaux de recherches visant à définir et/ou améliorer les architectures, les services, les communications, ..., des systèmes de géolocalisation [Wan *et al.*, 2007; ALRahmawy et Wellings, 2007; Xu et Jacobsen, 2007; Satoh, 2007; Coronato *et al.*, 2006; Goßmann et Specht, 2002].

De nombreux et divers domaines applicatifs et/ou scientifiques utilisent des informations de localisation. Sans être exhaustif, on peut entre autre citer les applications géographiques cherchant à mettre en relation les variables spatiales et temporelles d'un territoire donné, [Church, 2002; Quiroga, 2000; de Oliveira et Ribeiro, 2001; Bonnifait *et al.*, 2007].

On peut également citer les applications en architecture, qui visent à la numérisation de bâtiments existants [Chen *et al.*, 2006; Caron *et al.*, 2007; Song *et al.*, 2007].

Enfin, un des domaines les plus en avance dans l'utilisation des données de localisation est sans doute celui de la robotique mobile visant à la localisation des robots et à la cartographie de

leur environnement [Begum *et al.*, 2008; Gechter *et al.*, 2006; Moreira *et al.*, 2001; Borges et Aidon, 2001; Herianto *et al.*, 2007].

Les applications dans le domaine de la production et logistique de ces technologies sont également nombreuses [Qiu, 2007; Kim *et al.*, 2007; Chow *et al.*, 2007; Kim *et al.*, 2008]. La traçabilité des produits, l'inventaire des stocks produits, la géolocalisation d'une flotte de transport n'en sont que quelques exemples. Il semble qu'à l'heure actuelle le positionnement spatial d'objets physiques se limite à des objets «volumineux» (camions, bateaux,...) ou à des personnes.

Notre travail de recherche consiste à montrer les apports, sur le contrôle et les performances d'un système manufacturier, d'un flux d'information de localisation de produits durant leur élaboration. Ce papier présente ici un cas d'application possible : la génération automatique de code de simulation de flux sur la base des données de localisation des produits, durant leur passage sur le système de production. Ces données constituent un flux d'information pouvant être assimilé à la trace des produits. Si, depuis quelques années, la simulation de flux est devenue incontournable pour l'évaluation de la dynamique des systèmes manufacturiers, [Cassandras et Lafortune, 1999; Park et Lee, 2005], il n'en demeure pas moins que les phases de modélisation, puis de maintenance, de ces modèles restent des opérations délicates et chronophages, [De Vin *et al.*, 2004; Mittal *et al.*, 2005; De Vin *et al.*, 2006]. Ces raisons expliquent, à elles seules, le choix de nous intéresser à cette problématique. L'idée est de remplacer la plus grosse partie des interventions des experts humains lors de la construction du modèle, mais également lors des phases de maintenance ou de reconfiguration, par un générateur automatique. La figure 1 en montre très schématiquement le principe. Le générateur est alimenté par un flux de données en provenance du système réel.

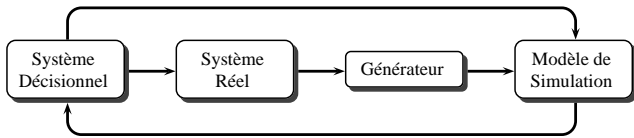


Figure 1. Le «générateur» dans son environnement

2 HYPOTHÈSES & PROBLÉMATIQUE

2.1 Hypothèses de départ

L'hypothèse principale de ce travail est de considérer que tous les objets élémentaires d'un système de production manufacturier peuvent être localisés. On suppose ainsi qu'il existe une technologie capable de fournir un flux de localisation de tous les objets en mouvement dans le système considéré. Nous parlerons dans notre cas de localisation plutôt que de géolocalisation de par la nature et l'échelle du système étudié, l'échelle considérée se limitant à la taille d'un atelier de production. D'autres hypothèses viennent naturellement compléter cette hypothèse principale. En effet, les données observées seront considérées dans ce travail comme fiables et non entachées d'erreur, notre idée étant ici de poser les principes de la méthode proposée le plus simplement possible (la prise en compte des erreurs fera l'objet d'autres travaux et présentations). De plus, l'obtention des données de localisation se fait naturellement par l'intermédiaire de capteurs, embarqués ou non sur les produits, qui via un système de communication alimentent un système de gestion de l'information (Figure 2). Les données de localisation sont, comme d'autres grandeurs physiques, issues de l'environnement des produits en circulation : le système manufacturier. L'accès aux données de localisation doit pouvoir être considéré comme possible en tous les endroits du système et à chaque moment, c'est pourquoi le système doit être un système pervasif. Dans notre cas on peut imaginer que les données sont collectées de manière événementielle, ou de manière discrète (cela dépend de la technologie employée). Dans ce dernier cas, si la fréquence d'acquisition est très élevée, le flux d'information de localisation peut être considéré comme quasi-continu. Enfin, on considérera qu'un identifiant unique est assigné *a priori* à chacun des produits-objets élémentaires manipulés à un moment ou à un autre par le système de production.

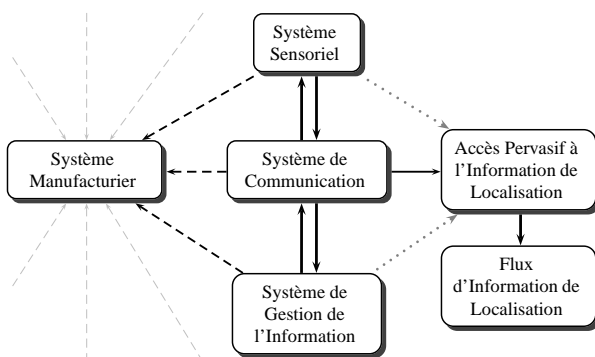


Figure 2. Pour obtenir le flux de localisation...

2.2 Formalisation du problème

L'information de localisation accessible est définie par le 3-tuple (I, R, T) où I est l'ensemble des identifiants des objets. Un identifiant est assigné à chacun des objets de façon unique.

L'ensemble $R \subset \mathbb{R}^2$ est l'ensemble de positions ($r = (x, y)$, $r \in R$) et T représente le temps. Chaque (i, r, t) , est un élément dans le flux f . Cette information est la seule qui sera utilisée pour la génération du code de simulation. Notre problème consiste à concevoir un générateur de modèle de simulation en ligne Φ , capable d'élaborer un modèle m à partir d'un flux de données réelles F .

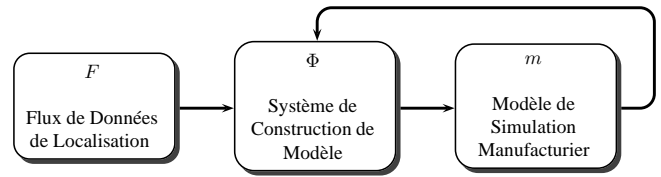


Figure 3. Système de construction du modèle de simulation

Ce générateur Φ doit être capable d'adapter m en fonction de l'évolution du flux F dans le temps (équation 2).

De façon plus formelle nous pouvons noter :

$$F = \{f_1, f_2, f_3, \dots\}, \quad (1)$$

où f_k représente un flux à un instant donné, il est alors l'un des composants du flux F . On considérera que le modèle est initialement vide. L'arrivée d'un nouveau flux au générateur lui permet une mise à jour du modèle m . Le problème consiste à définir Φ pour obtenir m à chaque instant d'arrivée d'un nouveau flux. Cette manière de procéder permet d'adapter le modèle aux modifications émanant du système réel : le modèle en est le reflet instantané. Ainsi nous proposons une forme de présentation récursif du processus d'obtention du modèle m (avec m_0 le modèle initial vide) :

$$\begin{aligned} m_0 &= \emptyset \\ m_k &= \Phi(f_k, m_{k-1}) \end{aligned} \quad (2)$$

3 MODÉLISATION

3.1 Objets et produits concernés

Dans ce travail nous considérons le «produit» comme étant un (ou des) objet en cours d'élaboration, depuis sa naissance à l'entrée du système, jusqu'à sa disparition à sa sortie. Des objets, composites ou non, peuvent être assemblés : le résultat de l'assemblage est un agglomérat d'objets nommés produit. Un produit peut être désassemblé : ce processus générant ainsi des objets plus élémentaires. Ce sont ces objets les plus élémentaires à qui sont liés un identifiant unique. Nous allons introduire ici sur la base d'un exemple simple d'un atelier à trois machines M_1 , M_2 et M_3 , et un type de produit P (Fig. 4) cette notion d'objets et de produit. Dans cet exemple, le produit est composé de trois objets a , b , c . L'obtention de P se fait par l'agglomération progressive des objets initiaux (a , b , c) en objets intermédiaires (p_1, p_2) (Fig. 4). On peut représenter ces processus de manière formelle :

$$\begin{aligned} a &\mapsto M_1(a) = p_1 \\ (p_1, b) &\mapsto M_2(p_1, b) = p_2 \\ (p_2, c) &\mapsto M_3(p_2, c) = P, \end{aligned} \quad (3)$$

ou de façon plus synthétique par une composition de fonctions :

$$P = M_3(M_2(M_1(a), b), c) \quad (4)$$

Nous pouvons représenter les equations (3) et (4), comme un graphe $G = (V, E)$ (Fig. 4), ou les sommets sont :

$$V = \{a, b, c, M_1, M_2, M_3, P\} \quad (5)$$

et la matrice d'adjacence des sommets est :

$$Adj(V) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Cette dernière formalisation, sous forme numérique, nous sera utile, lors de la mise en place de notre algorithme, pour établir la composition des produits finaux.

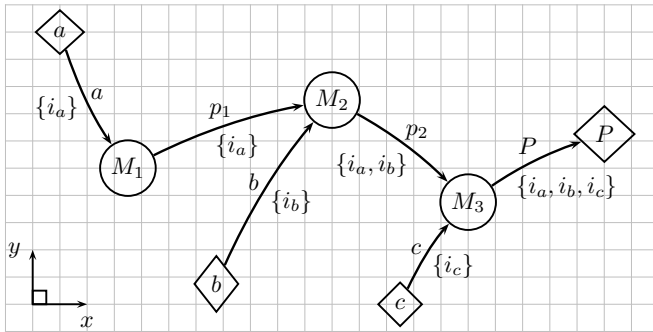


Figure 4. Atelier, trois machines et un produit. Flux d'indicateurs et objets

3.2 Trajectoire et historique des produits

Cette partie reprend l'exemple introduit au paragraphe précédent. Sur la figure (4) sont représentées deux choses différentes :

- la composition (ou nomenclature) d'un type de produit, (cf paragraphe précédent)
- la localisation dans le plan (coordonnées cartésiennes x et y) des ressources nécessaires à l'élaboration de P , ainsi que le parcours des objets, composites ou élémentaires.

Sur ce type de schéma les losanges représentent la naissance ou la disparition des objets et/ou produits (un losange avec une flèche sortante est naturellement un point de naissance, un losange avec une flèche entrante un point de disparition) pour l'étude considérée.

Le suivi de la trajectoire d'un objet initial dans le plan (x, y) dans le temps nous permet d'en déterminer la vitesse v : soit celle-ci est nulle, soit celle-ci est positive (Fig. 5). Le produit est en mouvement ou il est arrêté : cette dernière propriété étant le signe d'une attente... d'où le choix de construire un modèle de simulation basé sur un réseau de files d'attente. L'information $v = 0$ situe un point particulier à la base de la construction de notre modèle. Pour chacun de ces points il est possible d'obtenir un histogramme représentatif du temps de «service» par type de produit.

Cependant, avant la disparition d'un produit, aucune information sur sa composition n'est disponible. Les objets (initiaux et intermédiaires) ne fournissent aucune information quant à leur destination finale. C'est à la disparition du produit final que sa composition est découverte (de par les mouvements joints de chacun de ses composants, Fig. 4). Il est alors possible de retracer l'ensemble des parcours de ses constituants jusqu'à leur

naissance, et ainsi de mettre à jour les histogrammes des temps de service d'un constituant destiné à un produit final sur un point $v = 0$.

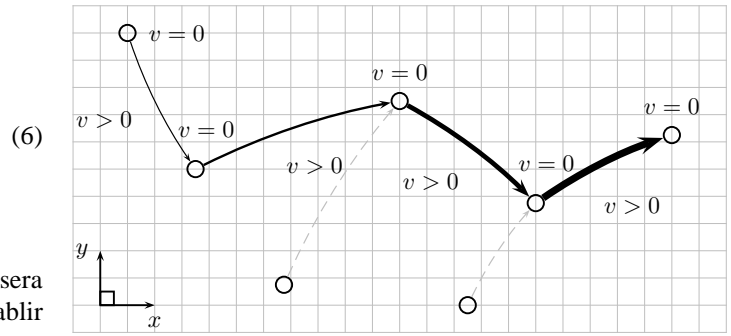


Figure 5. Vitesses composante a

3.3 Files d'attente

A chaque point où $v = 0$, nous allons associer un comportement de type file d'attente (Fig. 6).

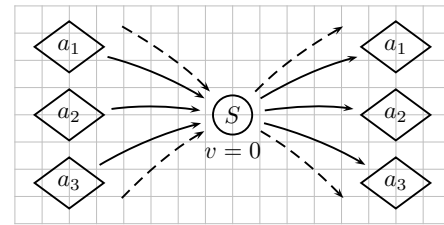


Figure 6. Point de service, $v = 0$

En ces points deux types d'événements peuvent se produire : l'arrivée ou le départ d'objet(s). Comme la composition d'un produit ne peut être connue qu'à la sortie de cette file d'attente, les objets constituant le produit possèdent alors forcément suivent alors la même trajectoire. Nous définissons alors (Fig. 7) :

- e_1 l'instant d'arrivée du premier objet constituant le produit.
- e_2 l'instant d'arrivée du dernier objet constituant le produit.
- s est l'instant commun de sortie de tout les objets constituant le produit.

Sur ces bases nous pouvons alors obtenir :

- O , la durée entre l'arrivée du premier objet et l'arrivée du dernier objet nécessaire à la constitution du produit.
- A , le temps d'attente du service : durée définie entre le moment où toutes les pièces nécessaires à la constitution du produit sont arrivées et le moment réel de passage.
- T , le temps de service au point $v = 0$.
- T_{Tot} , le temps total d'arrêt au point $v = 0$.

On peut lier ces différentes variables par l'équation :

$$T_{Tot} = O + A + T \quad (7)$$

Pour calculer A il est nécessaire de connaître le temps de sortie du produit précédent s^{k-1} où $k - 1$ représente l'ordre de sortie du produit. La règle de calcul de A est : si $e_2^k < s^{k-1}$ alors le temps d'attente est $s^{k-1} - e_2^k$, sinon le temps d'attente est 0.

De manière plus formelle et en utilisant la fonction d'Heaviside :

$$\theta(y) = \begin{cases} 1 & \text{si } y \geq 0 \\ 0 & \text{si } y < 0, \end{cases} \quad (8)$$

nous pouvons maintenant définir à chaque $k^{\text{ième}}$ sortie d'un produit les variables introduites jusqu'à présent :

$$\begin{aligned} y^k &= e_2^k - s^{k-1} \\ \alpha^k &= \theta(y^k) \\ \beta^k &= 1 - \alpha^k \\ T^k &= \alpha^k(s^k - e_2^k) + \beta^k(s^k - s^{k-1}) \\ A^k &= \beta^k(s^{k-1} - e_2^k) \\ O^k &= e_2^k - e_1^k \end{aligned} \quad (9)$$

à la condition où $s^0 = 0$.

Le 3-tuple (T, A, O) possède toute l'information nécessaire à la caractérisation du point $v = 0$ comme point de service.

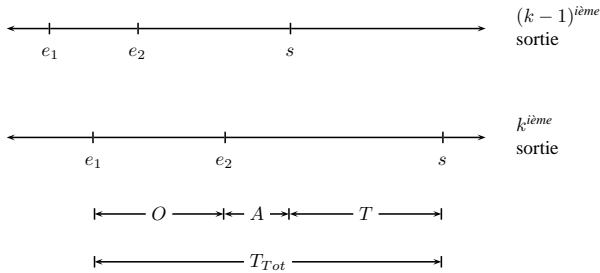


Figure 7. Temps dans un point $v = 0$ avec congestion

Il nous faut noter que l'analyse présentée ici n'est valide qu'à la condition théorique où la file d'attente se réduit à un seul point (tout les objets attendent au même point).

4 ALGORITHME

L'algorithme proposé ici correspond naturellement au générateur de code de simulation. Il est composé de trois parties principales.

La première partie génère les positions des machines et les chemins suivis par les produits, sur la base des flux de données temps réel qui l'alimentent. La seconde partie traite la problématique de sortie des produits du système. La sortie des produits est en effet le moment où il est possible de savoir si oui ou non ce produit existait ou pas. S'il existait on met à jour ses données, sinon on le créer. La troisième partie sert à modéliser les lois de comportement pour le modèle ainsi généré. Des lois statistiques sont proposées pour représenter les temps d'inter-arrivées et de services pour chaque type de produit et chaque machine. L'algorithme lié à cette première partie est déclenché à chaque modification du flux. A chaque nouvelle donnée (i, r, t) un nouveau produit est créé si son identifiant est nouveau. A chaque arrêt de ce produit est créé un «point d'arrêt» (point de localisation d'une file d'attente et/ou point de service), et un lien entre produit et «point d'arrêt». Ce «point d'arrêt» est validé si la position du produit i est la même entre deux instant d'observations t et t_s , le point correspond à une réelle attente du produit. Dans ce cas les temps d'entrée et de sortie du produit i sur le point d'arrêt localisé en r sont conservés. Ils servent ensuite à la modélisation du comportement dynamique du type de produit en ce point.

L'algorithme lié à cette deuxième partie est déclenché à chaque sortie de produit du système. Savoir si un produit sort du système peut être en soi un problème difficile à résoudre. Nous avons fait le choix de considérer un produit comme sorti du système si les données avec l'identifiant i n'apparaissent plus

dans le flux pendant une durée considérée (choisie arbitrairement mais suffisamment importante). Comme nous l'avons déjà expliqué dans l'équation (3) ou sur la figure (4) il faut être capable de retrouver la composition du produit final à la sortie (ses composés). A ce niveau nous utilisons les matrices d'adjacence liées à chaque produit élémentaire représenté par son identifiant. La somme des matrices d'adjacence de chacun des produits élémentaires sortant au même lieu et eu même moment, permet l'obtention de la composition du produit. L'algorithme compare cette somme des matrices avec celles obtenues précédemment. Si cette matrice n'existe pas encore, cela signifie qu'un nouveau produit est sorti du système. On lui relie alors les informations sur ses points de service (ou d'arrêt) obtenus pendant le parcours de ses différents composants dans l'atelier.

for d **do**

if $\exists p \in P: p.i = d.i$ **then**

if $p.m.s = 1$ **then**

if $p.m.r = d.r$ **then**

$p.m.a \leftarrow p.m.a + d.t - p.m.t$

$p.m.t \leftarrow d.t$

else

$m \leftarrow m' \in M: m'.r = p.m.r$

if $p.m.t - p.m.a < m.l$ **then**

$p.m.T \leftarrow p.m.t - m.l$

else

$p.m.T \leftarrow p.m.a$

end if

$m.l \leftarrow p.m.t$

$p.M \leftarrow p.M \cup \{p.m\}$

$p.m \leftarrow \check{m}(d.r, d.t)$

end if

else

if $p.m.r = d.r$ **then**

$p.m.s \leftarrow 1$

if $\nexists m \in M: m.r = d.r$ **then**

$M \leftarrow M \cup \{p.m\}$

end if

else

$p.m.r \leftarrow d.r$

end if

$p.m.t \leftarrow d.t$

end if

else

$m \leftarrow \check{m}(d.r, d.t)$

$p \leftarrow \check{p}(d.i, m)$

$P \leftarrow P \cup \{p\}$

end if

end for

Figure 8. Algorithme simplifiée

La troisième partie consiste à caractériser le comportement stochastique des différents points d'arrêt. Cette troisième et dernière phase est une phase de post-traitement de l'ensemble des informations collectées en phase 1 et 2. Une estimation de la fonction de densité de probabilité pour les temps d'inter-arrivées et les temps de services est menée. Elle est ensuite validée par un test de Wilcoxon [Wilcoxon et Co, 1997].

Nous présentons (Fig. 8) un fragment de l'algorithme développé dans sa version simplifiée. Ce fragment permet de collecter les positions des points d'arrêts pour chaque produit, leurs

temps d'inter-arrivées et leurs temps de service à chaque point d'arrêt.

Dans l'algorithme présenté une donnée (i, r, t) est définie par la structure d , où $d.i$ est l'identifiant, $d.r$ la position (x, y) et $d.t$ l'instant. Les autres structures de données utilisées sont m et p . La structure m , est définie dans la table (1), et la fonction pour la créer est $\check{m}(r, t)$.

Tableau 1. structure de données m

var	description	défaut
$m.r$	position (x, y)	none
$m.t$	instant courant	none
$m.s$	binaire pour désigner l'arrêt	0
$m.T$	temps de service	0.0
$m.l$	instant de sortie	0.0
$m.a$	temps d'arrêt	0.0

La structure p est définie dans la table (2). Elle est créée par la fonction $\check{p}(i, m)$.

Tableau 2. structure de données p

var	description	défaut
$p.i$	identifiant	none
$p.m$	structure de type m	none
$p.M$	ensemble ordonné de structures m	\emptyset

En plus des structures d, m, p , il existe deux ensembles globaux, M , et P , vides par défaut.

5 APPLICATION

Cette partie a vocation à valider le principe de faisabilité de la génération automatique d'un code de simulation sur la base d'informations de localisation.

Nous avons donc dans un premier temps généré un module de simulation avec SimPy (Simulation in Python) [Muller, 2004; Muller et Vignaux, 2003; Bahouth *et al.*, 2007]. Celui-ci nous génère le flux de localisation des produits. Il fait office d'artefact du système réel. Nous nous limitons ici volontairement à un cas simple pour limiter le volume imparti à sa présentation et à l'analyse des résultats dans le cadre de ce papier. Le flux de localisation est ensuite récupéré par le module générateur de modèle de simulation qui met en œuvre les phases 1 et 2 vues précédemment. Enfin, un module d'analyse des données (phase 3) permet de vérifier les résultats obtenus. Tout les modules sont programmés avec le langage Python [Rossum, 1995; Downey *et al.*, 2002; Cai, 2005].

5.1 Présentation du cas d'application choisi

L'atelier modélisé ici est composé de différentes machines entre lesquelles les produits évoluent, transportés par des AGV's.

Les différents types de produits sont générés pour la formule :

$$mach(s, l) = (s - l) \bmod M \quad (10)$$

– l : le type de produit, $l = 0, \dots, L - 1$,

- s : l'opérations dans la gamme, $s = 0, \dots, S - 1$,
- M : nombre de machines ou poste de travail,

basée sur la formulation initiale proposée par Thiesse et Fleisch [2008], que nous avons légèrement modifiée. La formule ainsi proposée permet de générer des séquences de transformation (les gammes) pour chacun des produits.

- Les temps d'interarrivées des produits dans l'atelier sont fournis par une fonction de distribution exponentielle. Une proportion identique de chaque type de produit est générée.
- Les temps de service de chaque machine sont également fournis par une fonction de distribution exponentielle.
- La disposition des machines dans le système est réalisée de manière aléatoire en début de simulation, dans un cadre également figé (dimensions du système).
- Le système de localisation permet d'obtenir l'information de localisation du produit (i, r, t) avec une fréquence fixe.

5.2 Mise en œuvre avec SimPy

Les paramètres utilisés dans la simulation sont les suivantes :

Tableau 3. conditions expérimentales

paramètres	valeurs
nombre de types de produits	3
nombre de machines	3
nombre d'opérations ou phases	3
quantité de produits	100
temps d'inter-arrivées	exponentiel, $\mu = 1/3$
temps de service sur machines	exponentiel, $\mu = 13$
nombre d'AGV's	10
vitesse des AGV's	0.44
dimensions de l'atelier	$(-10, -10), (10, 10)$
période d'échantillonnage pour la localisation	0.5

5.3 Résultats

Les résultats fournis par le modèle issu du générateur de code sont tout a fait cohérents avec ceux issus de l'artefact du système réel. En effet le test de Wilcoxon, montre que les données, concernant les temps d'inter-arrivées et les temps de service générées par les deux modèles sont distribuées selon la même loi statistique (Tables 4 et 5).

Tableau 4. Test de Wilcoxon temps inter-arrivées

Produit	Opération		
	0	1	2
0	0	1	2
t -statistic	168.5	138.0	131.0
two-tailed p -value	0.7	0.0	0.0
1	0	1	2
t -statistic	96.0	5.0	124.0
two-tailed p -value	0.49	0.0	0.0
2	0	1	2
t -statistic	240.5	195.5	192.5
two-tailed p -value	0.73	0.0	0.0

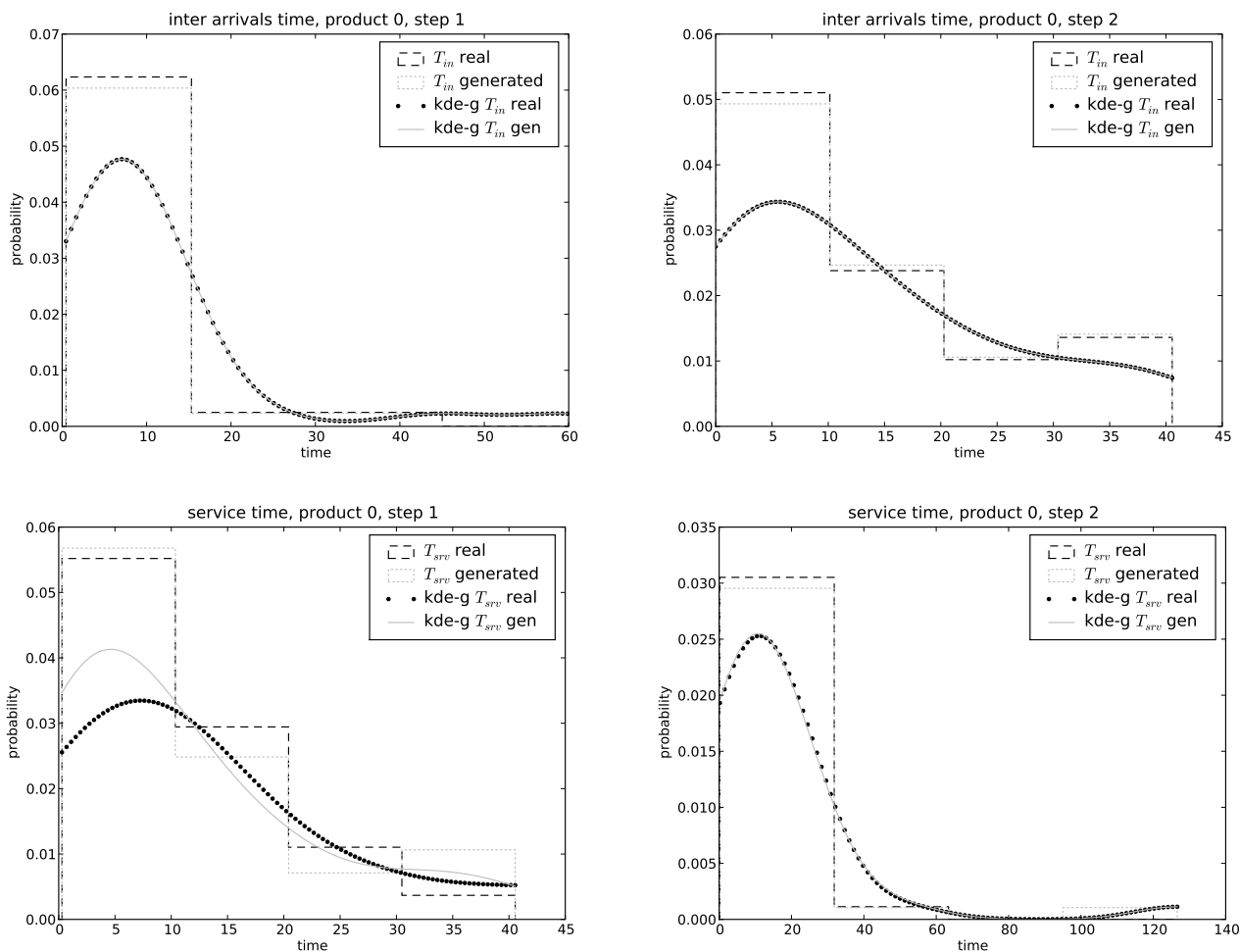


Figure 9. Distributions des temps inter-arrivées (première file) et des temps de service (deuxième file)

Tableau 5. Test de Wilcoxon temps de service

Produit	Opération		
0	0	1	2
<i>t</i> -statistic	220.0	187.0	185.5
two-tailed <i>p</i> -value	0.8	0.35	0.3
1	0	1	2
<i>t</i> -statistic	154.0	175.0	166.0
two-tailed <i>p</i> -value	0.07	0.15	0.17
2	0	1	2
<i>t</i> -statistic	261.0	270.0	309.0
two-tailed <i>p</i> -value	0.07	0.09	0.37

Les produits et leurs gammes sont intégralement identifiés par le générateur de code. Celui-ci positionne également de manière précise les différentes ressources mises en œuvre pour l'élaboration des produits.

Ces premiers résultats montrent la faisabilité de la méthode proposée. Ils ont été confirmés par des tests effectués sur des systèmes de taille plus importante (jusqu'à 15 machines, 15 opérations ou phases, 15 types de produits).

6 CONCLUSIONS

La démarche et le «générateur» présenté dans ce papier ont été validés par l'exemple et les résultats encourageants auxquels il

nous a conduit. Actuellement cette première version du "générateur" est en cours de modification afin d'y intégrer les imprécisions de mesure des trajectoires produits. C'est là une étape nécessaire avant la confrontation avec un système réel, dont les données de localisation produit extraites seront forcément bruitées. Cette confrontation incontournable permettra de valider dans des conditions réelles et en grandeur réelle le modèle et donc le générateur développé. Dans le futur, nous souhaitons également examiner les possibilités d'enrichissement du modèle par l'extraction d'autres règles ou paramètres sur la base des données de géolocalisation des produits (comme par exemple les règles de priorités devant chacun des serveurs). Nous réfléchissons également au transfert de cette méthodologie à d'autres types d'application hors champs manufacturier.

7 RÉFÉRENCES

- ALRahmawy, M. et Wellings, A. (2007). A model for real time mobility based on the RTSJ. In *JTRES '07 : Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems*, pages 155–164, New York, NY, USA. ACM Press.
- Bahouth, A., Crites, S., Matloff, N. et Williamson, T. (2007). Revisiting the Issue of Performance Enhancement of Discrete Event Simulation Software. In *40th Annual Simulation Symposium (ANSS'07)*, volume 0, pages 114–122, Los Alamitos, CA, USA. IEEE Computer Society.

- Begum, M., Mann, G. K. et Gosin, R. G. (2008). Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots. *Applied Soft Computing*.
- Bonnifait, P., Jabbour, M. et Dherbomez, G. (2007). Real-Time Implementation of a GIS-Based Localization System for Intelligent Vehicles. *EURASIP Journal on Embedded Systems*, 2007.
- Borges, G. A. et Aidon, M. J. (2001). Design of a Robust Real-Time Dynamic Localization System for Mobile Robots. In Devy, M. et Frédéric, L., éditeurs. *SIRS 2001 : proceedings of the 9th international symposium on intelligent robotic systems*, pages 425–434. LAAS-CNRS.
- Cai, X. (2005). On the performance of the Python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1), pp. 31–56.
- Caron, F., Razavi, S. N., Song, J., Vanheeghe, P., Duflos, E., Caldas, C. et Haas, C. (2007). Locating sensor nodes on construction projects. *Auton Robot*.
- Cassandras, C. G. et Lafortune, S. (1999). *Introduction to Discrete Events Systems*. Kluwer Academic Publishers, Dordrecht.
- Chen, L.-C., Sheu, R.-K., Lu, H.-C., Lo, W.-T. et Chu, Y.-P. (2006). Object Finding System Based on RFID Technology. In Shen, H. T., Li, J., Li, M., Ni, J. et Wang, W., éditeurs. *APWeb Workshops : XRA, IWSN, MEGA, and ICSE, Harbin, China, January 16-18, 2006, Proceedings*, volume 3842, pages 383–396. Springer.
- Chow, H. K. H., Choy, K. L. et Lee, W. B. (2007). A dynamic logistics process knowledge-based system - An RFID multi-agent approach. *Know.-Based Syst.*, 20(4), pp. 357–372.
- Church, R. L. (2002). Geographical information systems and location science. *Computers and Operations Research*, 29(6), pp. 541–562.
- Coronato, A., Pietro, G. D. et Esposito, M. (2006). A Semantic Location Service for Pervasive Grids. In Meersman, R., Tari, Z. et Herrero, P., éditeurs. *On the Move to Meaningful Internet Systems 2006 : OTM 2006 Workshops*, volume 4278, pages 1274–1284. Springer.
- de Oliveira, M. G. S. et Ribeiro, P. C. M. (2001). Production and analysis of coordination plans using a geographic information system. *Transportation Research Part C*, 9(1), pp. 53–68.
- De Vin, L. J., Ng, A. H. C. et Oscarsson, J. (2004). Simulation-Based Decision Support for Manufacturing System Life Cycle Management. *Journal of Advanced Manufacturing Systems*, 3, pp. 115–128.
- De Vin, L. J., Ng, A. H. C., Oscarsson, J. et Andler, S. F. (2006). Information Fusion for Simulation Based Decision Support in Manufacturing. *FAIM 2005 Special Issue of Robotics and Computer Integrated Manufacture*, 22, pp. 429–436.
- Downey, A., Elkner, J. et Meyers, C. (2002). *How to Think Like a Computer Scientist : Learning with Python*. Green Tea Press.
- Gechter, F., Chevrier, V. et Charpillat, F. (2006). A reactive agent-based problem-solving model : Application to localization and tracking. *ACM Trans. Auton. Adapt. Syst.*, 1(2), pp. 189–222.
- Goßmann, J. et Specht, M. (2002). Location Models for Augmented Environments. *Personal Ubiquitous Comput.*, 6(5-6), pp. 334–340.
- Herianto, Sakakibara, T. et Kurabayashi, D. (2007). Artificial Pheromone System Using RFID for Navigation of Autonomous Robots. *Journal of Bionic Engineering*, 4(4), pp. 245–253.
- Karkkainen, M., Holmstrom, J., Framling, K. et Artto, K. (2003). Intelligent products - a step towards a more effective project delivery chain. *Computers in Industry*, 50, pp. 141–151.
- Kim, J., Tang, K., Kumara, S., Yee, S. T. et Tew, J. (2008). Value analysis of location-enabled radio-frequency identification information on delivery chain performance. *International Journal of Production Economics*, 112(1), pp. 403–415.
- Kim, M.-C., Kim, C. O., Hong, S. R. et Kwon, I.-H. (2007). Forward-backward analysis of RFID-enabled supply chain using fuzzy cognitive map and genetic algorithm. *Expert Systems with Applications*, In Press, Corrected Proof.
- Mittal, S., Mak, E. et Nutaro, J. J. (2005). DEVS-Based Dynamic Model Reconfiguration and Simulation Control in the Enhanced DoDAF Design Process. *submitted to Journal of Defense Modeling and Simulation*.
- Moreira, A. P., Sousa, A. et Costa, P. (2001). Vision Based Real-Time Localization of Multiple Mobile Robots. In *3rd Int. Conf. on Field and Service Robotics*, pages 103–106.
- Muller, K. (2004). Advanced systems simulation capabilities in SimPy. *Europython 2004*.
- Muller, K. et Vignaux, T. (2003). SimPy : Simulating Systems in Python. *ONLamp.com Python Devcenter*.
- Park, K.-J. et Lee, Y.-H. (2005). A On-line Simulation Approach to Search Efficient Values of Decision Variables in Stochastic Systems. *Int J Adv Manuf Technol*.
- Qiu, R. G. (2007). RFID-enabled automation in support of factory integration. *Robot. Comput.-Integr. Manuf.*, 23(6), pp. 677–683.
- Quiroga, C. A. (2000). Performance measures and data requirements for congestion management systems. *Transportation Research Part C*, 8, pp. 287–306.
- Rossum, G. (1995). Python reference manual. CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands.
- Satoh, I. (2007). A location model for smart environments. *Pervasive Mob. Comput.*, 3(2), pp. 158–179.
- Song, J., Haas, C. T. et Caldas, C. H. (2007). A proximity-based method for locating RFID tagged objects. *Advanced Engineering Informatics*, 21(4), pp. 367–376.
- Thiesse, F. et Fleisch, E. (2008). On the value of location information to lot scheduling in complex manufacturing processes. *International Journal of Production Economics*, 112(2), pp. 532–547.
- Wan, T., Zeitouni, K. et Meng, X. (2007). An OLAP system for network-constrained moving objects. In *SAC '07 : Proceedings of the 2007 ACM symposium on Applied computing*, pages 13–18, New York, NY, USA. ACM Press.
- Wilcoxon, F. et Co, A. C. (1997). Individual Comparisons by Ranking Methods. *Breakthroughs in Statistics*.
- Xu, Z. et Jacobsen, A. (2007). Adaptive location constraint processing. In *SIGMOD '07 : Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 581–592, New York, NY, USA. ACM.