Génération automatique de modèle de simulation par les données de localisation des produits

Andres Véjar, Fréderic Chaxel, Jean-Yves Bron, Patrick Charpentier

Génération automatique de modèle de simulation par les données de localisation des produits

Andrés Véjar — Frédéric Chaxel — Jean-Yves Bron — Patrick Charpentier

Laboratoire CRAN (CNRS UMR 7039 - Nancy-Université)
Université Henri Poincaré - Campus Sciences
B.P. 70239
54506 Vandœuvre-lès-Nancy Cedex
{prénom.nom}@cran.uhp-nancy.fr

RÉSUMÉ. *Le travail présenté dans ce papier vise, à partir d'un flux de données de localisation de produits dans un processus de production, à construire un modèle de simulation à événements discrets. La construction de ce modèle, via un «générateur» en ligne, s'appuie sur les trajectoires et l'identification des produits. Le modèle ainsi construit est capable de s'autoadapter aux évolutions du système réel.*

ABSTRACT. *This paper aims to build a discrete-event simulation model for production processes using a location data flux of products. The model building is carried out by an on-line* generator *and it is based on the identification of trajectoires and products. The resulting model presents a self-adaptive behavior to changes in the environment.*

MOTS-CLÉS : *Flux, Localisation, Produit, Géolocalisation, Simulation, Modélisation.*

KEYWORDS: *Flux, Location, Product, Geolocation, Simulation, Modeling.*

1. Introduction

La simulation est devenue un outil incontournable lors de l'évaluation des performances des systèmes de production manufacturier. Les étapes du processus conduisant à l'élaboration d'un modèle et à l'identification de ses paramètres (récolte des données, analyse comportementale, ...) sont encore très souvent réalisées par l'homme. Ces étapes constituent la plus grosse partie du temps dévolu à une simulation (Willemain, 1995). Pour tenter de raccourcir ces durées, quelques travaux ont proposé une automatisation de certaines des étapes, comme par exemple (Guru and Savory, 2004; Zhou et al., 2006; Son et al., 2000). Nous proposons pour notre part une approche utilisant les données de localisation des produits en cours de transformation, pour construire automatiquement, et en temps réel, un modèle de simulation de flux. Les produits sont donc ici considérés comme des objets immergés dans un système de production, aptes à fournir en tout lieu, en cas de besoin, des informations diverses, dans notre cas, des informations de localisation et d'identité. On parlera de systèmes ambiants pour le produit.

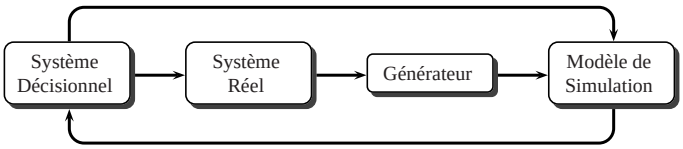

Figure 1. *Le «générateur» dans son environnement*

Ce modèle, de type réseau de file d'attente, est capable de s'auto-adapter aux modifications du système réel. En effet, un flux permanent de données issues du système réel, permet une mise à jour du modèle en cas d'évolution. Le modèle ainsi obtenu peut, de part sa nature, être utilisé pour améliorer le fonctionnement du système (réengineering) ou pour contrôler le système (exploitation). Un générateur de code de simulation, visant à prouver la faisabilité de la démarche a été développé en Python (Rossum, 1995; Downey et al., 2002; Cai, 2005) avec l'aide du package SimPy (Muller and Vignaux, 2003; Muller, 2004; Bahouth et al., 2007). Quelques tests ont été réalisés, et il s'avère que les comportements des systèmes réels et des modèles de simulation obtenus par le biais de ce générateur sont très proches.

2. Contexte

Depuis quelques années maintenant, de nouvelles visions du produit manufacturé tendent à lui conférer des capacités de communication et des capacités sensitives 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 a définir et/ou améliorer les architectures, les services, les communications, …, des systèmes de géolocalisation (Wan et al., 2007; ALRahmawy and Wellings, 2007; Xu and Jacobsen, 2007; Satoh, 2007; Coronato et al., 2006; Goßmann and 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 cherchent à mettre en relation les variables spatiales et temporelles d'un territoire donné, (Church, 2002; Quiroga, 2000; de Oliveira and 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 and 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. Depuis quelques années, la simulation de flux est devenue incontournable pour l'évaluation de la dynamique des systèmes manufacturiers, (Cassandras and Lafortune, 1999; Park and Lee, 2005). En effet, les modèles de simulation de flux sont utilisés pour dimensionner un système en phase de conception, pour améliorer son fonctionnement en phase de ré-engineering, et anticiper un comportement en phase d'exploitation (Mirdamadi et al., 2007). Malgré tout, 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), et ce quel qu'en soit le type d'application visée. Ces raisons expliquent, à elles seules, le choix

de nous intéresser à cette problématique. Le type de modèle auquel nous aboutissons par le générateur proposé doit permettre son utilisation ou en phase de ré-engineering ou en phase d'exploitation. L'intérêt de notre proposition est d'autant plus important que la complexité et la dynamique du système réel à modéliser est grande. 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.

3. Hypothèses & Problématique

3.1. *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 part 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).

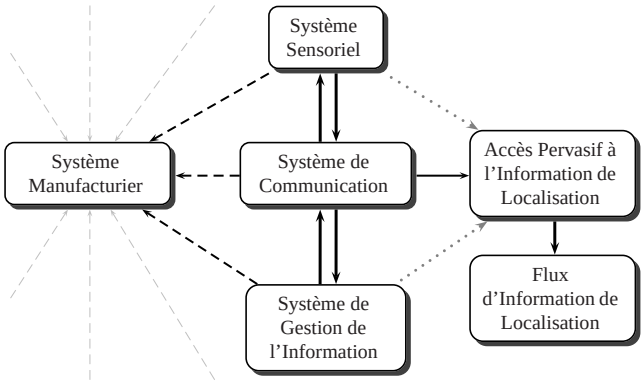

Figure 2. *Obtention du flux de localisation*

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.

3.2. *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[1] de façon unique.

Soit I l'ensemble discret des identifiants de chaque objet composant un produit, R l'ensemble des positions (par exemple $r = (x, y)$, $r \in R \subset \mathbb{R}^2$) , et T l'ensemble des dates. Le passage d'un objet dans l'atelier (ou processus productif) est associé à un parcours défini par un ensemble de paires position-temps pour cet objet. Si l'objet est possesseur d'un identifiant $i \in I$, chaque donnée de localisation est définie comme $(i, r, t), r \in R, t \in T$. Le flux de données de localisation $D \subset I \times R \times T$ est l'information de parcours de tous les objets :

$$D = \{d_k\}_{k=1}^{\infty} = \{(i_k, r_k, t_k) : t_{k+1} \geq t_k, k \in \mathbb{N}\} \tag{1}$$

Chaque (i, r, t), est un élément dans le flux D. 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 D.

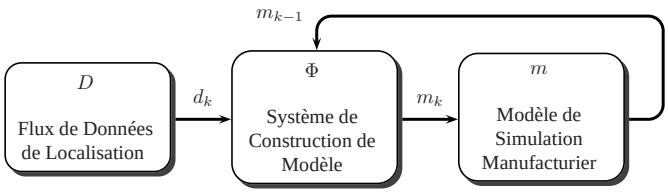

Figure 3. *Système de construction du modèle de simulation*

1. Objet : tout ou partie d'un produit en cours d'élaboration. Il peut subir des transformations élémentaires, être assemblé ou désassemblé. A chaque objet élémentaire est associé un identifiant unique. Le produit est une agglomération d'un certain nombre de ces objets ...

Ce générateur Φ doit être capable d'adapter m en fonction de l'évolution du flux D dans le temps (équation 2). 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 récursive du processus d'obtention du modèle m (avec m_0 le modèle initial vide) :

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

3.3. *Trajectoires et flux de données : modélisation*

Durant leur passage dans le système manufacturier, les objets suivent des trajectoires dans le plan (x, y). L'observation de ces trajectoires dans le temps nous permet de déterminer la vitesse v de l'objet. A chaque point où $v = 0$ est possible associer un comportement de type file d'attente. En chacun de 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 constituants le produit suivent alors forcément la même trajectoire.

A chaque point où $v = 0$, nous allons associer un comportement de type file d'attente. Nous définissons alors (figure 4) :

– 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 constituants le produit.

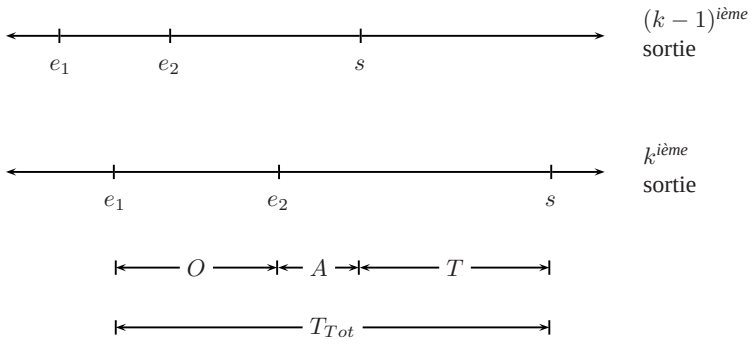

Figure 4. *Temps dans un point $v = 0$ avec congestion*

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 :

$$T_{Tot} = O + A + T \tag{3}$$

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 :

$$\theta(y) = \begin{cases} 1 & \text{si } y \geq 0 \\ 0 & \text{si } y < 0, \end{cases} \tag{4}$$

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

$$\begin{array}{lcl} y^k & = & e_2^k - s^{k-1} \\ \alpha^k & = & \theta(y^k) \\ \beta^k & = & 1 - \alpha^k \\ T^k & = & \alpha^k(s^k - e_2^k) + \beta^k(s^k - s^{k-1}) \\ A^k & = & \beta^k(s^{k-1} - e_2^k) \\ O^k & = & e_2^k - e_1^k \end{array} \tag{5}$$

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. 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). De plus, il nous est impossible en l'état, sur la base des informations récoltées, de déterminer un comportement détaillé au niveau des files d'attentes (type de règles de gestion, préemption, ...), ou de discriminer la panne d'un «serveur» d'un autre état d'attente. Le modèle n'est que le reflet des informations qui ont servies à le construire, avec le point de vue du produit : celui-ci ne sait si la machine est en panne ou dans un autre état.

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.

Algorithme 1 Génération du Layout

procedure LAYOUT(D)
  $S \leftarrow$ SERVEURS(D) ▷ Points de vitesse nulle avec récurrence
  $A \leftarrow$ PARCOURS(D) ▷ Les arcs entre points
  $P \leftarrow$ PRODUITS(A, S) ▷ C'est un graphe $p_j = (\text{arcs, points})$
  $L \leftarrow$ CREERLAYOUT(A, S, P) ▷ C'est l'union de tous les
  ▷ graphes de produits $L = \cup_j p_j$
  return L ▷ Le layout
end procedure

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.

Algorithme 2 Génération des Distributions

procedure DISTRIBUTIONS(D, L)
  $T \leftarrow$ DISTRTYPES(D, L) ▷ Identifier la distribution d'entrée de
  ▷ chaque type de produit pour chaque serveur
  $(t_i, t_s) \leftarrow$ DISTRTEMPS(D, L) ▷ Identifier la distribution de temps
  ▷ d'inter-arrivées, et de temps de service
  ▷ pour chaque serveur et pour chaque produit
  return (T, t_i, t_s) ▷ Les distributions
end procedure

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). 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.

Algorithme 3 Algorithme simplifié

for all 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
        $m.l \leftarrow p.m.t$
        $p.M \leftarrow p.M \cup \{p.m\}$
        $p.m \leftarrow \breve{m}(d.r, d.t)$
      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
    $m \leftarrow \breve{m}(d.r, d.t)$
    $p \leftarrow \breve{p}(d.i, m)$
    $P \leftarrow P \cup \{p\}$
  end if
end for

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 and Co, 1997). Nous présentons (*cf.* algorithme 3) 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 (*cf.* tableau 1), et la fonction pour la créer est $\breve{m}(r, t)$.

Tableau 1. *Structures de données m et p*

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

La structure p est définie dans (*cf.* tableau 1). Elle est créée par la fonction $\breve{p}(i, m)$. En plus des structures d, m, p, il existe deux ensembles globaux, M, et P, vides par défaut.

5. Application

Cette partie à 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 and 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 modélise 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 :

$$mach(s, l) = (s - l) \mod M \tag{6}$$

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

– s : l'opérations dans la gamme, $s = 0, \ldots, S - 1$,

– M : nombre de machines ou poste de travail,

basée sur la formulation initiale proposée par Thiesse and Fleisch (2008). La formule ainsi proposée permet de générer des séquences de transformation (les gammes) pour chacun des produits.

– Les temps d'inter-arrivées 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.

Les paramètres utilisées dans la simulation sont présentés dans (*cf.* tableau 2).

Tableau 2. *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

5.2. *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 (*cf.* tableau 3). Les produits et leurs gammes sont intégralement identifiés par le générateur de code.

Tableau 3. *Test de Wilcoxon temps inter-arrivées et de service*

Produit		T inter-arrivées			T de service		
	opération	0	1	2	0	1	2
0	t-statistic	168.5	138.0	131.0	220.0	187.0	185.5
	two-tailed p-value	0.7	0.0	0.0	0.8	0.35	0.3
	opération	0	1	2	0	1	2
1	t-statistic	96.0	5.0	124.0	154.0	175.0	166.0
	two-tailed p-value	0.49	0.0	0.0	0.07	0.15	0.17
	opération	0	1	2	0	1	2
2	t-statistic	240.5	195.5	192.5	261.0	270.0	309.0
	two-tailed p-value	0.73	0.0	0.0	0.07	0.09	0.37

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

Les résultats obtenus valident la démarche mise en œuvre ainsi que l'outil que nous avons développé. Le «générateur» tel qu'il a été conçu possède des avantages liés à la réalisation du modèle de manière automatique et en ligne :

– réduction globale du temps de modélisation par la réduction du temps de collecte de données, de génération du modéle et de sa maintenance.

– fiabilisation du modèle obtenu par la réduction de l'intervention humaine dans le processus.

Nous développons actuellement une autre version du «générateur» qui tienne compte des imprécisions de mesure de localisation, ainsi que des variations de la période d'échantillonnage. Une fois développé, cette version du «générateur» pourra alors être confrontée à une validation en aveugle sur un système réel. De manière encore plus large, nous réflechissons actuellement à l'ouverture à d'autres types d'application (hors manufacturing), du même concept.

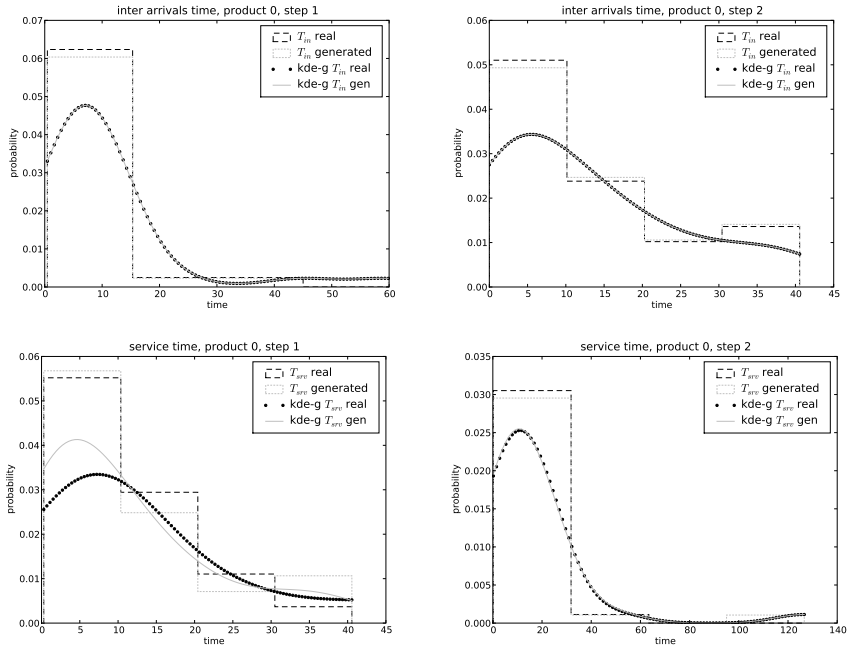

Figure 5. *Distributions des temps inter-arrivées (première file) et des temps de service (deuxième file)*

Références

Mohammed ALRahmawy and Andy Wellings. 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, 2007. ACM Press. ISBN 978-59593-813-8.

Alex Bahouth, Steven Crites, Norman Matloff, and Todd Williamson. 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, 2007. IEEE Computer Society.

Momotaz Begum, George K.I. Mann, and Raymond G. Gosin. Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots. *Applied Soft Computing*, 2008.

Philippe Bonnifait, Maged Jabbour, and Gérald Dherbomez. Real-Time Implementation of a GIS-Based Localization System for Intelligent Vehicles. *EURASIP Journal on Embedded Systems*, 2007, 2007.

G. A. Borges and M. J. Aidon. Design of a Robust Real-Time Dynamic Localization System for Mobile Robots. In Michel Devy and Lerasle Frédéric, editors, *SIRS 2001 : proceedings of the 9th international symposium on intelligent robotic systems*, pages 425–434. LAAS-CNRS, 2001.

X. Cai. On the performance of the Python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1) :31–56, 2005.

Francois Caron, Saiedeh Navabzadeh Razavi, Jongchul Song, Philippe Vanheeghe, Emmanuel Duflos, Carlos Caldas, and Carl Haas. Locating sensor nodes on construction projects. *Auton Robot*, 2007.

C. G. Cassandras and S. Lafortune. *Introduction to Discrete Events Systems.* Kluwer Academic Publishers, Dordrecht, 1999.

Lun-Chi Chen, Ruey-Kai Sheu, Hui-Chieh Lu, Win-Tsung Lo, and Yen-Ping Chu. Object Finding System Based on RFID Technology. In Heng Tao Shen, Jinbao Li, Minglu Li, Jun Ni, and Wei Wang, editors, *APWeb Workshops : XRA, IWSN, MEGA, and ICSE, Harbin, China, January 16-18, 2006, Proceedings*, volume 3842, pages 383–396. Springer, 2006. ISBN 3-540-31158-0.

Harry K. H. Chow, K. L. Choy, and W. B. Lee. A dynamic logistics process knowledge-based system - An RFID multi-agent approach. *Know.-Based Syst.*, 20 (4) :357–372, 2007. ISSN 0950-7051.

R. L. Church. Geographical information systems and location science. *Computers and Operations Research*, 29(6) :541–562, 2002.

Antonio Coronato, Giuseppe De Pietro, and Massimo Esposito. A Semantic Location Service for Pervasive Grids. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006 : OTM 2006 Workshops*, volume 4278, pages 1274–1284. Springer, 2006. ISBN 3-540-48273-3.

M. G. S. de Oliveira and P. C. M. Ribeiro. Production and analysis of coordination plans using a geographic information system. *Transportation Research Part C*, 9 (1) :53–68, 2001.

L. J. De Vin, A. H. C. Ng, and J. Oscarsson. Simulation-Based Decision Support for Maunufacturing System Life Cycle Management. *Journal of Advanced Manufacturing Systems*, 3 :115–128, 2004.

L. J. De Vin, A. H. C. Ng, J. Oscarsson, and S. F. Andler. Information Fusion for Simulation Based Decision Support in Manufacturing. *FAIM 2005 Special Issue of Robotics and Computer Integrated Manufacture*, 22 :429–436, 2006.

A. Downey, J. Elkner, and C. Meyers. *How to Think Like a Computer Scientist : Learning with Python*. Green Tea Press, 2002.

Franck Gechter, Vincent Chevrier, and François Charpillet. A reactive agent-based problem-solving model : Application to localization and tracking. *ACM Trans. Auton. Adapt. Syst.*, 1(2) :189–222, 2006. ISSN 1556-4665.

Joachim Goßmann and Marcus Specht. Location Models for Augmented Environments. *Personal Ubiquitous Comput.*, 6(5-6) :334–340, 2002. ISSN 1617-4909.

Ashu Guru and Paul Savory. A Template-Based Conceptual Modeling Infrastructure for Simulation of Physical Security Systems. In R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *WSC'04 : Proceedings of the 36th conference on Winter simulation*, pages 866–873, 2004.

Herianto, Toshiki Sakakibara, and Daisuke Kurabayashi. Artificial Pheromone System Using RFID for Navigation of Autonomous Robots. *Journal of Bionic Engineering*, 4(4) :245–253, 2007.

M. Karkkainen, J. Holmstrom, K Framling, and K. Artto. Intelligent products - a step towards a more effective project delivery chain. *Computers in Industry*, 50 : 141–151, 2003.

J. Kim, K. Tang, S. Kumara, S. T. Yee, and J. Tew. Value analysis of location-enabled radio-frequency identification information on delivery chain performance. *International Journal of Production Economics*, 112(1) :403–415, 2008.

Moon-Chan Kim, Chang Ouk Kim, Seong Rok Hong, and Ick-Hyun Kwon. Forward-backward analysis of RFID-enabled supply chain using fuzzy cognitive map and genetic algorithm. *Expert Systems with Applications*, In Press, Corrected Proof, 2007.

Samieh Mirdamadi, Franck Fontanili, and Lionel Dupont. Discrete Event Simulation-Based Real-Time Shop Floor Control. In Ivan Zelinka, Zuzana Oplatková, and Alessandra Orsoni, editors, *21st European Conference on Modelling and Simulation ECMS 2007*, pages 572–577, 2007.

S. Mittal, E. Mak, and J. J. Nutaro. DEVS-Based Dynamic Model Reconfiguration and Simulation Control in the Enhanced DoDAF Design Process. *submitted to Journal of Defense Modeling and Simulation*, 2005.

Antonio Paulo Moreira, Armando Sousa, and Paulo Costa. Vision Based Real-Time Localization of Multiple Mobile Robots. In *3rd Int. Conf. on Field and Service Robotics*, pages 103–106, jun 2001.

K. Muller. Advanced systems simulation capabilities in SimPy. *Europython 2004*, 2004.

K. Muller and T. Vignaux. SimPy : Simulating Systems in Python. *ONLamp.com Python Devcenter*, 2003.

K.-J. Park and Y.-H. Lee. A On-line Simulation Approach to Search Efficient Values of Decision Variables in Stochastic Systems. *Int J Adv Manuf Technol*, 2005.

Robin G. Qiu. RFID-enabled automation in support of factory integration. *Robot. Comput.-Integr. Manuf.*, 23(6) :677–683, 2007. ISSN 0736-5845.

Cesar A. Quiroga. Performance measures and data requirements for congestion management systems. *Transportation Research Part C*, 8 :287–306, 2000.

G. Rossum. Python reference manual. CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 1995.

Ichiro Satoh. A location model for smart environments. *Pervasive Mob. Comput.*, 3 (2) :158–179, 2007. ISSN 1574-1192.

Young Jun Son, Albert T. Jones, and Richard A. Wysk. Automatic Generation of Simulation Models From Neutral Libraries : An Example. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *WSC'00 : Proceedings of the 32th conference on Winter simulation*, pages 1558–1567, 2000.

Jongchul Song, Carl T. Haas, and Carlos H. Caldas. A proximity-based method for locating RFID tagged objects. *Advanced Engineering Informatics*, 21(4) :367–376, 2007.

Frédéric Thiesse and Elgar Fleisch. On the value of location information to lot scheduling in complex manufacturing processes. *International Journal of Production Economics*, 112(2) :532–547, 2008.

Tao Wan, Karine Zeitouni, and Xiaofeng Meng. 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, 2007. ACM Press. ISBN 1-59593-480-4.

F. Wilcoxon and A. C. Co. Individual Comparisons by Ranking Methods. *Breakthroughs in Statistics*, 1997.

T. R. Willemain. Model Formulation : What Experts Think About and When. *Operations Research*, 43(6) :916–932, 1995.

Zhengdao Xu and Arno Jacobsen. 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, 2007. ACM. ISBN 978-1-59593-686-8.

Ming Zhou, Qun Zhang, and Zhimin Chen. What Can Be Done to Automate Conceptual Simulation Modeling ? In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *WSC '06 : Proceedings of the 38th conference on Winter simulation*, pages 809–814, 2006.