Générateur de code de simulation à partir de l'information de localisation du produit

Andrés Véjar[1], Patrick Charpentier[1]

[1]Centre de Recherche en Automatique de Nancy (CRAN)
Nancy Université, Faculté des Sciences et Techniques, BP 239, 54506 Vandœuvre Cedex, France

andres.vejar@cran.uhp-nancy.fr, patrick.charpentier@cran.uhp-nancy.fr
http://page-perso.cran.uhp-nancy.fr/perso_av/sgc

Résumé— **Cette communication propose une méthode originale de génération d'un code de simulation pour des systèmes à événements discrets. Pour ce faire nous utilisons l'information de localisation des produits pendant le fonctionnement du système. Ce flux d'information sert à l'algorithme proposé à générer un modèle de simulation de type file d'attente.**

Mots-clés— **Flux, Localisation, Produit, Géolocalisation, Simulation, Modélisation.**

I. Introduction

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 [1]. 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. Les applications dans le domaine de la production et logistique de ces technologies sont nombreuses [2], [3], [4], [5]. 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 [6], [7]. 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 [8]. 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, [9], [10], [11], 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.

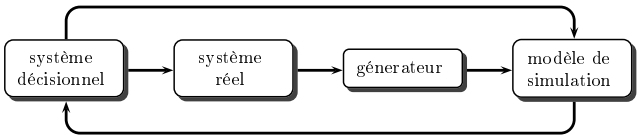

Fig. 1
Le «générateur» dans son environnement

II. Modèle

A. Définition du Produit

Dans ce travail nous considérons le «produit» comme étant un ou plusieurs objets en cours d'élaboration. Des objets peuvent être assemblés : le résultat de l'assemblage est un agglomérat d'objets que nous nommerons produit. Un produit peut être désassemblé : ce processus générant ainsi des objets plus élémentaires. A chaque objet est associé un identifiant unique qui nous permettra de suivre cet objet pendant toute son évolution dans le système. Un

agglomérat d'objets est donc perçu comme un agglomérat d'identifiants. La seconde hypothèse de ce travail est de considérer que tous les objets élémentaires 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.

Les données observées seront considérées dans ce travail comme fiables et non entachées d'erreur, notre idée étant d'ici de poser les principes de la méthode proposée le plus simplement possible. Cette hypothèse implique que les trajectoires de produits du même type seront en tout points identiques.

B. Flux d'information de localisation

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 (Fig. 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 est considéré comme pouvant être effectué en tous les endroits du système et à chaque moment, c'est pourquoi nous qualifions le système de pervasif.

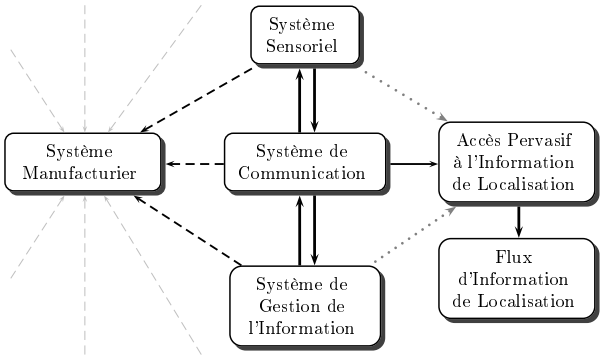

Fig. 2
Pour obtenir le flux de localisation...

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.

Concrètement l'information de localisation est définie par le 3-tuple (I, R, T) où I est l'ensemble des identifiants des objets. Un identifiant est assigné a priori à chacun des objets de façon unique. $R \subset \mathbb{R}^2$ 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.

C. Le problème

Notre problème consiste à obtenir grâce à un générateur de modèle de simulation en ligne Φ, un modèle m à partir d'un flux de donnés réelles F.

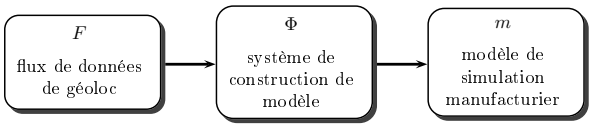

Fig. 3
Système de construction du modèle de simulation

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

De façon un peu plus formelle nous pouvons noter :

$$F = \{f_1, f_2, f_3, \ldots\}, \tag{1}$$

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 récursive du processus d'obtention du modèle m (avec m_0 le modèle initial vide) :

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

D. Réflexions

Nous allons introduire la solution que nous proposons par le biais 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). Celui-ci 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. 6). On peut représenter ces processus de manière formelle :

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

ou de façon plus synthétique par une composition de fonctions :

$$P = M_3\left(M_2\left(M_1(a), b\right), c\right) \tag{4}$$

Nous pouvons représenter les équations (3) et (4), comme un graphe $G = (V, E)$ (Fig. 4), où les vertex sont :

$$V = \{a, b, c, M_1, M_2, M_3, P\} \tag{5}$$

et la matrice d'adjacence des vertex est :

$$Adj(V) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6}$$

La figure (4) représente deux choses différentes :

1. la composition (ou nomenclature) d'un type de produit,

2. la localisation dans le plan 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.

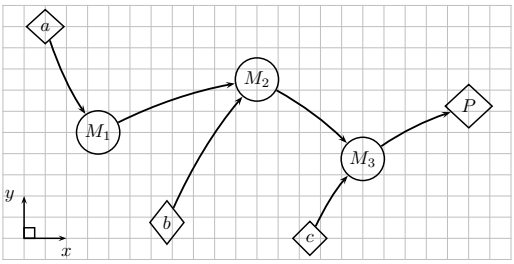

Fig. 4
Atelier, trois machines et un produit

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, assimilables à des «monoserveurs», il est possible d'obtenir un histogramme représentatif du temps de «service». Il existe bien sûr un histogramme par point où $v = 0$ pour chacun des types de produit. Cependant, avant la disparition d'un produit, aucune information sur sa composition n'est disponible. Les objets initiaux et intermédiaires ne sont en effet porteurs d'aucune information quant à leur destination finale. C'est à la sa disparition du produit final que sa composition est découverte (de par les mouvements joints de chacun de ses composants, Fig. 6). Il est alors possible de retracer l'ensemble des parcours de ses constituants jusqu'à leur naissance, et ainsi mettre à jour les histogrammes des temps de services d'un constituant destiné à un produit final sur un point $v = 0$.

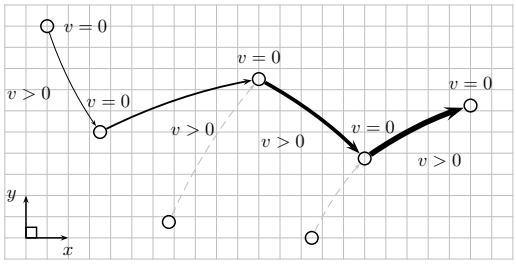

Fig. 5
Vitesses composante a

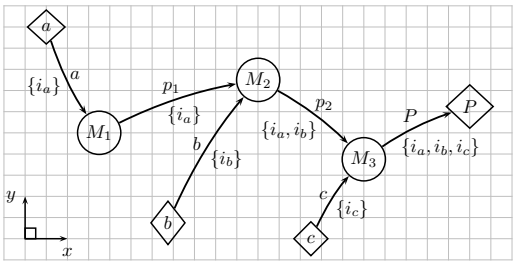

Fig. 6
Atelier, flux d'indicateurs et objets

E. Files d'attente

A chaque point où $v = 0$, nous allons associer un comportement de type file d'attente (Fig. 7).

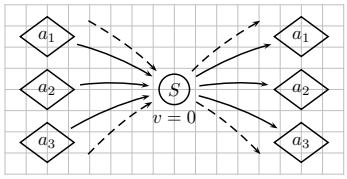

Fig. 7
Point de service, $v = 0$

En ces points deux types d'événements peuvent se produire : l'arrivée ou le départ d'objet(s). La composition d'un produit ne peut être connue qu'à la sortie de cette file d'attente : les objets constituants le produit possèdent alors forcément le même mouvement. Nous définissons alors (Fig. 8) :

- 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.

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 défini à partir du moment où toutes les pièces nécessaires à la constitution du produit sont déjà sont arrivées.
- 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 :

$$T_{Tot} = O + A + T \tag{7}$$

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.

Il nous est possible d'obtenir, par le biais d'informations issues d'un observateur placé à la sortie de la file d'attente, une loi de répartition de la longueur de la file d'attente L.

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

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

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

$$\begin{array}{lcl} y^k & = & e_2^k - s^{k-1} \\ \alpha^k & = & \theta(y^k) \\ \beta^k & = & 1 - \alpha^k \\ T^k & = & \alpha^k(s^k - e_2^k) + \beta^k(s^k - s^{k-1}) \\ A^k & = & \beta^k(s^{k-1} - e_2^k) \\ L^k & = & \beta^k(1 + L^{k-1}) \\ O^k & = & e_2^k - e_1^k \\ T_{Tot}^k & = & s^k - e_1^k, \end{array} \tag{9}$$

à la condition où $(s^0, L^0) = (0, 0)$.

Le tuple (T, A, L, O, T_{Tot}) possède toute l'information nécessaire à la caractérisation du point $v = 0$ comme point de service.

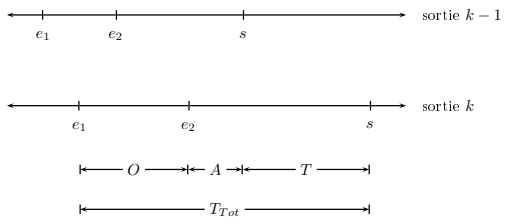

Fig. 8
TEMPS DANS UN POINT $v = 0$ AVEC CONGESTION

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). 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.

III. ALGORITHME

L'algorithme proposé 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'alimente. 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 ses informations sur ses points de service (ou d'arrêt) obtenus pendant le parcours de ses différents composants dans l'atelier.

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 [12].

Nous présentons (Fig. 9) 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 (I), et la fonction pour la créer est $\breve{m}(r, t)$.

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

TABLE I
STRUCTURE DE DONNÉES m

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

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

TABLE II
STRUCTURE DE DONNÉES p

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

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
        $m.l \leftarrow p.m.t$
        $p.M \leftarrow p.M \cup \{p.m\}$
        $p.m \leftarrow \breve{m}(d.r, d.t)$
      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

Fig. 9
ALGORITHME SIMPLIFIÉE

IV. 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) [13], [14], [15]. 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. Le flux de localisation est ensuite récupéré par le module générateur de modèle de simulation qui met en oeuvre 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 un langage de très haut niveau : Python [16], [17], [18].

A. Présentation du modèle

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 :

$$mach(s, l) = (s - l) \mod M \tag{10}$$

- 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 [19], et 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'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.

B. Mise en oeuvre avec SimPy

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

TABLE III
CONDITIONS EXPÉRIMENTALES

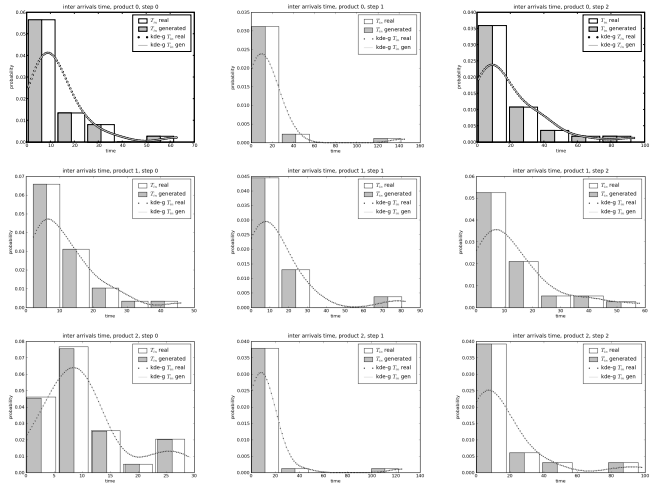

Fig. 10

Distributions des temps inter-arrivées dans le cas d'étude

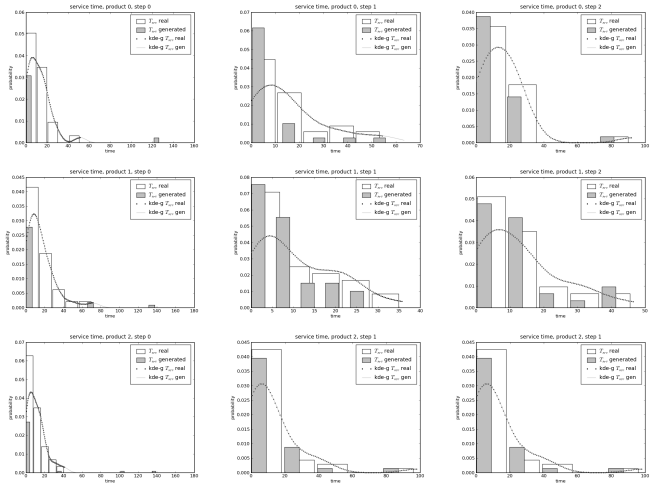

Fig. 11

Distributions des temps de service dans le cas d'étude

C. *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.

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).

V. Conclusions

Les résultats obtenus ici sont encourageants. Ils valident la démarche mise en œuvre ainsi que l'outil que nous avons développé. Une des prochaines étapes de notre travail consistera à prendre en compte les imprécisions de mesure de localisation, ainsi que les variations de la période d'échantillonnage et leurs impacts sur le générateur. Enfin, le générateur dans sa version ainsi établie pourra alors être confronté à une validation en aveugle sur un système réel.

Références

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

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

[3] Moon-Chan Kim, Chang Ouk Kim, Seong Rok Hong et 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.

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

[5] Jindae Kim, Kaizhi Tang, Soundar Kumara, Shang-Tae Yee et Jeffrey Tew : Value analysis of location-enabled radio-frequency identification information on delivery chain performance. *International Journal of Production Economics*, 112(1):403–415, Mar 2008.

[6] C. G. Cassandras et S. Lafortune : *Introduction to Discrete Events Systems*. Kluwer Academic Publishers, Dordrecht, 1999.

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

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

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

[10] S. Mittal, E. Mak et 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.

[11] L. J. De Vin, A. H. C. Ng, J. Oscarsson et 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.

[12] F. Wilcoxon et A. C. Co : Individual Comparisons by Ranking Methods. *Breakthroughs in Statistics*, 1997.

[13] K. Muller : Advanced systems simulation capabilities in SimPy. *Europython 2004*, 2004.

[14] K. Muller et T. Vignaux : SimPy : Simulating Systems in Python. *ONLamp. com Python Devcenter*, 2003.

[15] Alex Bahouth, Steven Crites, Norman Matloff et 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.

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

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

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

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