Simulation des processus de production de biens et services

(version du 16 octobre 2015)

Vincent Giard

Professeur émérite à l'Université Paris-Dauphine

Professeur affilié à l'EMINES *School of Industrial Management*

vincent.giard@dauphine.fr

CHAPITRE I

Principaux fondements de la démarche de modélisation/simulation de processus

CHAPITRE II

Les bases de la simulation d'un processus

CHAPITRE III

Caractérisation des Items, des Points d'Entrée et des Points de Sortie

CHAPITRE IV

Caractérisation des files d'attente

CHAPITRE V

Caractérisation des ressources

CHAPITRE VI

Caractérisation des activités

CHAPITRE VII

Compléments sur la modélisation sous Simul8

CHAPITRE VIII

Liste des simulations données en exemple

Chapitre I

PRINCIPAUX FONDEMENTS DE LA DÉMARCHE DE MODÉLISATION/SIMULATION DE PROCESSUS

On commencera par exposer la démarche de Monte Carlo qui fonde la simulation et on en illustrera l'application dans les modèles s'appuyant sur un tableur (§ I). On examinera ensuite les principes de modélisation d'un processus de production de biens ou de services et on verra comment la méthode de Monte Carlo est utilisée par les simulateurs de processus à (page 15).

I LA MÉTHODE DE MONTE CARLO

On commencera par un bref rappel sur les distributions de probabilités (§ I-1), sans doute inutile pour certains d'entre vous sur beaucoup de points; on privilégiera certains aspects dont on aura besoin dans la modélisation/simulation des processus. On examinera ensuite les fondements de la méthode de Monte Carlo (§ I-2, page 9) puis l'application de cette démarche aux modèles construits sur tableur (§ I-3, page 10).

I-1 Distributions de probabilités

Dans la modélisation d'un système productif on modélise la variabilité des temps de traitement d'un item dans un processeur et celle des arrivées par des distributions de probabilités.

Les arrivées dans le système productif peuvent se définir, de manière duale:

- soit par le nombre aléatoire d'items arrivant au cours d'une période de référence (par exemple la minute),
- soit par l'intervalle aléatoire de temps séparant deux arrivées successives dans le système de production.

Cette seconde formulation est celle qui est retenue dans la théorie des files d'attente (voir § II-1, page 12) et dans les approches de modélisation / simulation de processus.

Deux distributions sont privilégiées pour caractériser les **arrivées aléatoires de clients** dans un système productif.

- La **loi de Poisson** correspond à la distribution du nombre d'événements X survenant dans un cadre spatio-temporel donné (exemple: cadre spatial = porte d'entrée d'un bureau de poste; cadre temporel = plage horaire précise; événement = entrée d'un client). La distribution de probabilités d'une loi de Poisson de paramètre λ' est donnée par la formule $P(X = x) = e^{-\lambda'} \cdot \lambda'^{x} / x!$. On montre qu'à cette distribution correspond, de manière duale, celle de l'intervalle de temps T séparant 2 événements successifs qui suit la **loi exponentielle** de paramètre

$\lambda = 1/\lambda'$ qui est telle que $P(T > t) = \int_t^\infty \lambda \cdot e^{-\lambda t} = e^{-\lambda t}$. Cette distribution exponentielle sera préférée à celle de Poisson dans la modélisation / simulation de processus productifs. On montre que la probabilité d'occurrence d'un événement entre *t* et t + ε (ε étant petit) est toujours la même quel que soit le temps écoulé depuis l'occurrence de l'événement précédent[1] ; cette distribution se caractérise donc par l'indépendance d'occurrence de deux événements consécutifs. Pour cette raison, le processus de Poisson est qualifié de «sans mémoire». La feuille «Arrivee de Poisson» du classeur Introduction_-Simulation.xls illustre ce processus ; vous y trouvez :

- la reconstitution de la fonction de répartition $P(T > t)$ d'une loi exponentielle de paramètre $\lambda = 0{,}25$;
- une simulation de 200 arrivées suivant cette loi exponentielle, avec une illustration[2] du passage de la loi exponentielle à la loi de Poisson ;
- une «simulation» de l'utilisation d'un logiciel de reconnaissance d'une distribution statistique (intégré à l'add-in @Risk : option «Ajuster les distributions» de l'onglet @Risk) partant d'une série de données supposées recueillies d'un central téléphonique d'un centre d'appel[3].

- La seconde distribution utilisée pour modéliser les entrées aléatoires d'items dans le système productif est la **distribution d'Erlang** qui est caractérisée par un paramètre de forme α (entier positif) et un paramètre d'échelle $1/\lambda$: $P(T > t) = e^{-t/\lambda} \cdot \sum_{i=1}^{\alpha-1} (\lambda \cdot t)^i / i!$; on montre que $E(T) = \alpha \cdot \lambda$ et $V(T) = \alpha \cdot \lambda^2$. Quand $\alpha = 1$, on retrouve la distribution exponentielle ; pour les valeurs supérieures, la fonction de densité de probabilité est unimodale et «se rapproche» de celle d'une loi Normale dès que α dépasse 5 ou 6 (la loi d'Erlang étant définie, comme l'exponentielle, pour les seules valeurs non négatives, contrairement à la loi Normale). La probabilité d'occurrence d'un événement entre *t* et t + ε (ε étant petit) dépend du temps écoulé depuis l'occurrence de l'événement précédent ainsi que des valeurs des paramètres α et λ. La figure 1 illustre des distributions d'Erlang associées à quelques combinaisons des paramètres α et λ, et la feuille ERLANG du classeur Introduction_Simulation.xls calcule des fonctions de répartition de ces distributions[4].

---

1. Cette probabilité conditionnelle correspond à ce que l'on appelle une fonction de hasard.
2. Les cellules B85:B284 correspondent à la génération aléatoire de 200 intervalles séparant deux arrivées successives suivant la loi exponentielle de paramètre $\lambda = 1/4$ (on verra en page 9 le mécanisme de génération d'occurrences d'une variable aléatoire). Les cellules C85:C284 correspondent aux dates ponctuelles de ces arrivées, calculées à partir des informations de la colonne précédente. Les cellules D85: D284 détermine le numéro de la minute d'arrivée de l'item ; la colonne suivante calcule le nombre d'événements survenant au cours des différentes minutes observées, information correspondant à la génération d'occurrences d'une loi de Poisson de paramètre $\lambda' = 4$.
3. Les cellules I85:I284 correspondent à l'enregistrement de dates d'arrivées d'appel mais en réalité, on a repris ici les valeurs arrondies des cellules C85:C284 (3 chiffres après la virgule) ; les cellules J85:J284 correspondent au calcul de l'intervalle séparant 2 appels successifs ; à partir de ces 200 données, le logiciel d'identification de distribution trouve une loi exponentielle de paramètre 0,255 (au lieu de 0,25 qui caractérise la population mère).

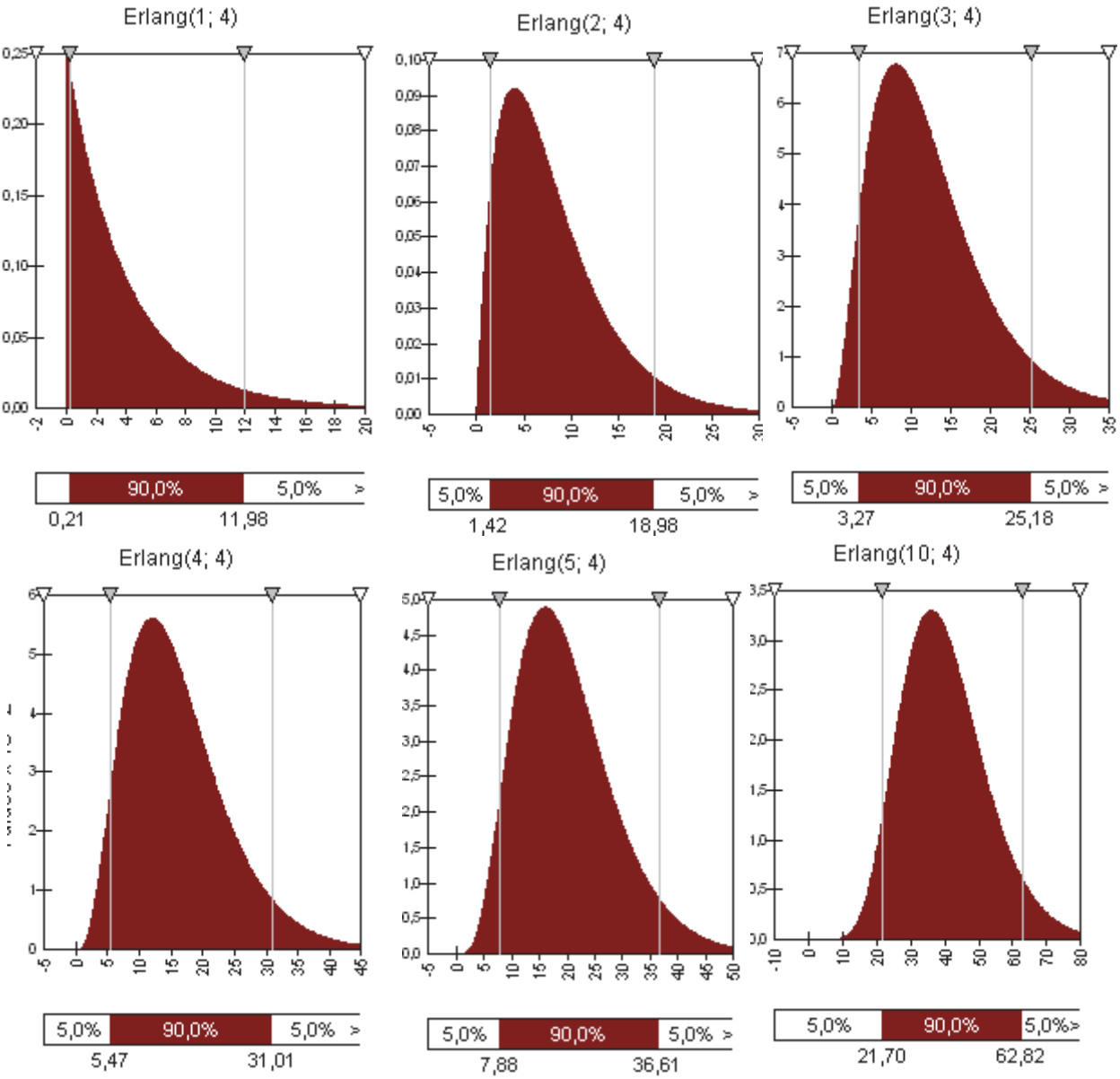


Figure 1
Exemples de distributions d'Erlang

En ce qui concerne les distributions de probabilités du **temps de traitement** d'un client dans un centre de production, on retrouve les distributions théoriques classiques (attention à ce qu'il soit pratiquement impossible d'avoir des valeurs négatives). Les contraintes de recueil de données lorsque l'on veut simuler le fonctionnement d'un système productif d'une certaine complexité conduisent souvent à utiliser la distribution uniforme et la distribution triangulaire.

- La **distribution uniforme** est utilisée lorsque les informations dont on dispose laissent penser que la variable à laquelle on s'intéresse ne peut pas prendre de valeur en dessous d'une borne inférieure (x_{Min}) ni au-dessus d'une borne supérieure (x_{Max}) et peut prendre n'importe quelle valeur comprise entre ces bornes avec la même probabilité. Cette distribution peut être

4. La fonction d'Excel utilisée est celle de la loi Gamma qui conduit à la distribution de Erlang lorsque α est entier.

définie :

- pour des valeurs ***discrètes*** de X ($\Rightarrow P(X = x) = 1/(x_{Max} - x_{Min} + 1)$, $\forall x$ $\Rightarrow P(X \leq x) = (x - x_{Min} + 1)/(x_{Max} - x_{Min} + 1)$)
- ou pour des valeurs ***réelles*** ($\Rightarrow P(X \leq x) = (x - x_{Min})/(x_{Max} - x_{Min})$)

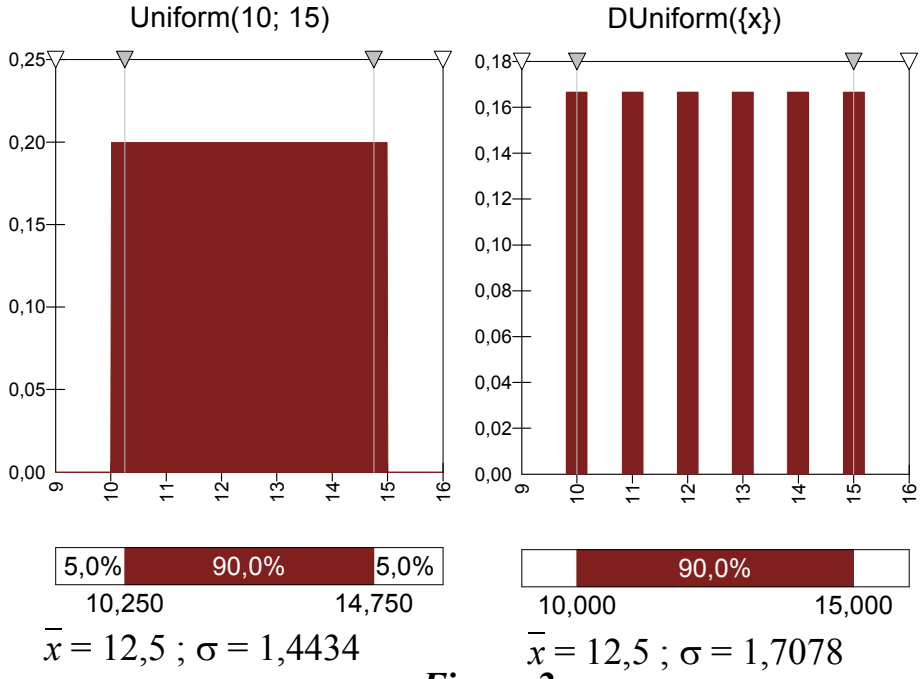


Figure 2

Exemple de distribution Uniforme continue et discrète $\mathscr{U}(10; 15)$ *&* $\mathscr{DU}(10; 15)$

- La **distribution triangulaire** est peu ou pas enseignée mais son intérêt opérationnel la rend incontournable dans nombreuses situations. Comme pour la loi uniforme, le spécialiste consulté estime que les valeurs de X sont bornées (x_{Min} et x_{Max}) mais il considère pouvoir s'engager sur l'information qu'il considère comme la plus probable (x_{Mode}) ; en outre, il estime que la probabilité d'observer x est d'autant plus forte que cette valeur est proche du mode. Cette distribution triangulaire[1] est donc définie pour $x_{Min} \leq x \leq x_{Max}$. Sa densité de probabilité est croissante linéairement jusqu'au mode x_{Mode}, puis décroissante linéairement. On a donc :
  - $f(x) = 2(x - x_{Min})/\{(x_{Max} - x_{Min})(x_{Mode} - x_{Min})\}$, pour $x \leq x_{Mode}$ et $f(x) = 2(x_{Max} - x)/\{(x_{Max} - x_{Min})(x_{Mode} - x_{Min})\}$ pour $x \geq x_{Mode}$.
  - En outre, $P(X \leq x) = (x - x_{Min})^2/\{(x_{Max} - x_{Min})(x_{Mode} - x_{Min})\}$, pour $x \leq x_{Mode}$ et $P(X \leq x) = 1 - 1 - (x_{Max} - x)^2/\{(x_{Max} - x_{Min})(x_{Mode} - x_{Min})\}$, pour $x \geq x_{Mode}$.
  - $E(X) = (x_{Min} + x_{Max} + x_{Mode})/3$;

1. Voir Giard, *Statistique appliquée à la gestion* (8e édition, 2003, Economica), p. 163.

- $V(X) = \{x_{Min}^2 + x_{Mode}^2 + x_{Max}^2 - x_{Min} \cdot x_{Max} - x_{Min} \cdot x_{Mode} - x_{Max} \cdot x_{Mode}\} / 18$

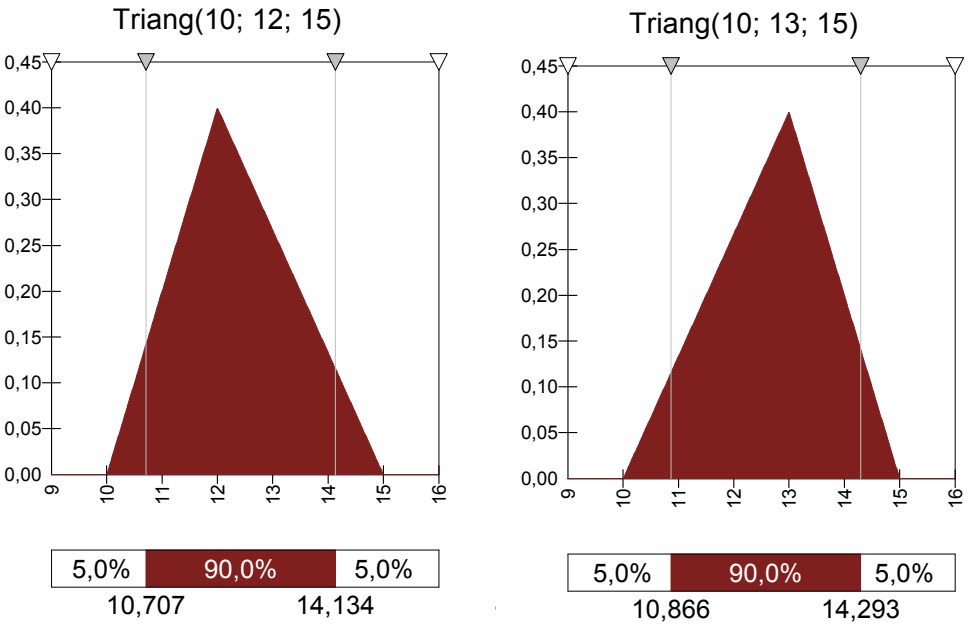


Figure 3
Exemple de distribution triangulaire : $\mathcal{T}(10\,;\,12\,;\,15)$ et $\mathcal{T}(10\,;\,13\,;\,15)$

I-2 Fondements de la méthode de Monte Carlo

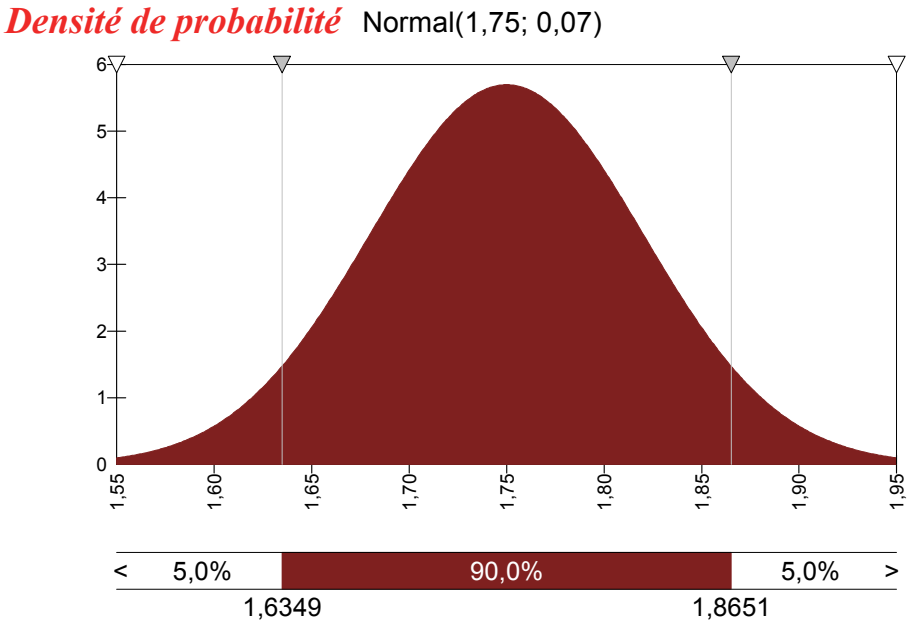
Par **méthode de Monte Carlo** on désigne un mécanisme de génération de réalisations indépendantes d'une variable aléatoire[1]. Ce mécanisme repose sur la combinaison de 2 concepts.

- La fonction de répartition de la variable aléatoire. La figure 4, page 10, illustre la fonction de densité de probabilité de la loi $\mathcal{N}(1{,}75\ ;\ 0{,}07)$ et la fonction de répartition de cette loi. À partir de cette dernière, il est évident que la connaissance de x entraîne celle de $P(X<x)$ et réciproquement.
- La génération d'un nombre aléatoire réel dont la valeur est comprise entre 0 et 1. La fonction Alea() d'Excel remplit cette fonction avec indépendance en probabilité des réalisations obtenues. Ce nombre aléatoire est considéré, dans la méthode de Monte Carlo comme une réalisation d'une probabilité cumulée.

À chaque réalisation aléatoire indépendante d'une probabilité cumulée, la fonction de répartition fait correspondre une occurrence x de la variable aléatoire X, indépendante des occurrences antérieurement obtenues. La feuille «Monte Carlo» du classeur Introduction_Simulation.xls illustre ce mécanisme.

Cette méthode de génération d'occurrences indépendantes d'une variable aléatoire fonde de nouvelles approches de gestion du risque, notamment en manage-

1. Bibliographie : principes de base dans V. Giard, *Statistique appliquée à la gestion* (8e édition, chapitre III, Economica, 2003) ; techniques avancées : D. Vose, *Risk Analysis : A Quantitative Guide* (3e édition, Wiley, 2008).



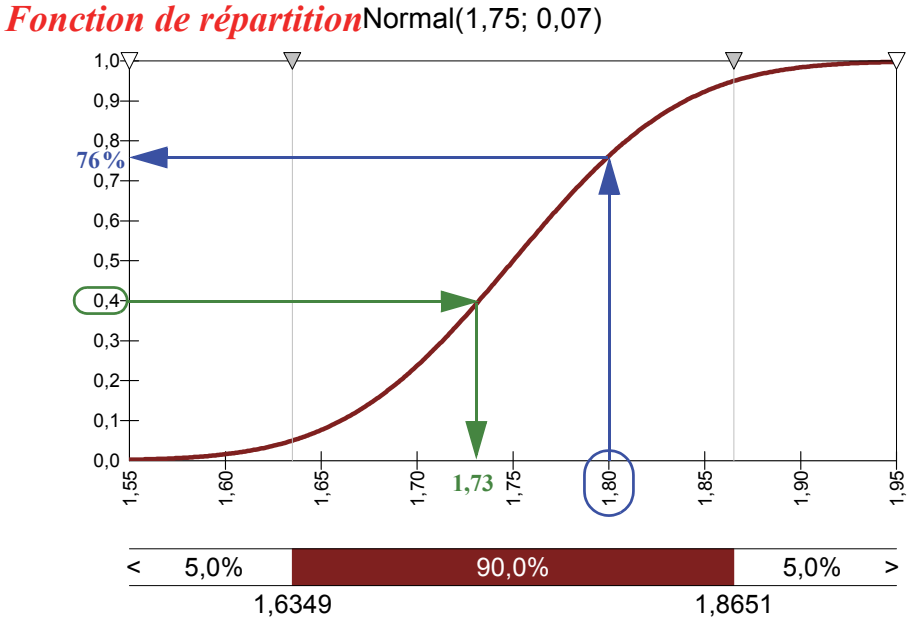


Figure 4

Fonction de répartition : $x \Rightarrow P(X<x)$ ou $P(X<x) \Rightarrow x$

ment de projet ou en aide à la décision à partir de modèles réalisés sur tableur. Elle est également au cœur des simulateurs de processus de production.

I-3 Usage de la méthode de Monte Carlo dans les tableurs

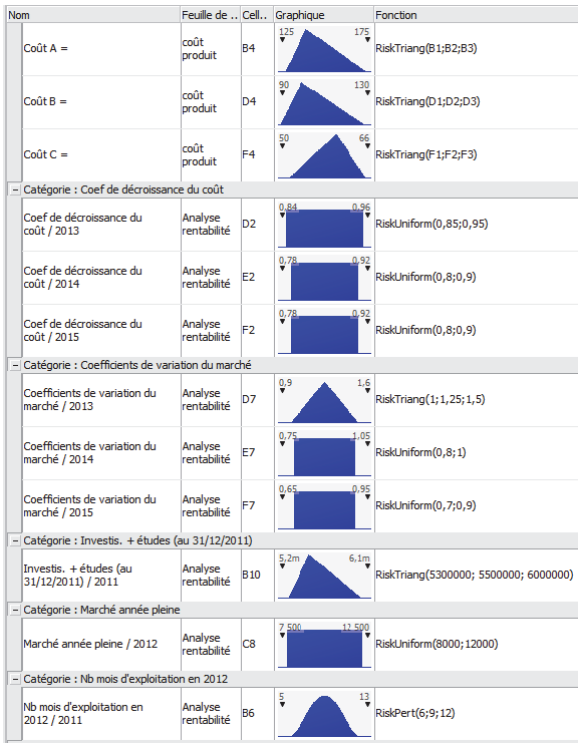
L'intérêt de cette démarche peut être montré à travers 2 exemples utilisant @Risk (normalement déjà téléchargé sur votre portable[1]).

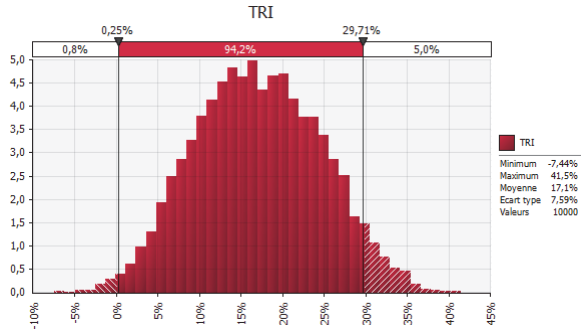
1. C'est l'une des composantes de *Decision Tools* pour les étudiants qui ont obtenu la licence de cette suite d'add-ins d'Excel. Pour les autres, une version d'évaluation est disponible à l'URL : http://www.palisade.com/risk/fr/.

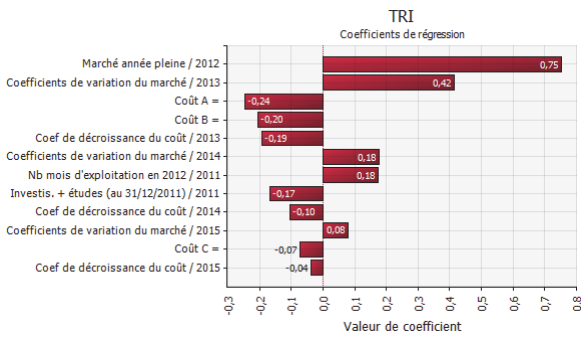
- **Exemple 1**. Ouvrez la feuille «Exemple 1 - Simulation» du classeur Introduction_Simulation.xls.On y traite le cas d'une politique calendaire de réapprovisionnement avec contrainte de capacité de transport. Dans cet exemple fictif on s'intéresse à une ligne d'assemblage de véhicules commune à 2 gammes A (25%) et B (75%) et plus particulièrement à l'approvisionnement des pare-brise de la gamme A, dont la production quotidienne X_A suit donc la loi $\mathscr{B}(400; 0{,}25)$. La commande est passée le soir pour une livraison le lendemain matin. Le camion utilisé est d'une capacité de 110 unités. La politique calendaire retenue est telle que risque accepté de rupture de stock de 5% (risque élevé retenu pour faciliter l'exemple). La détermination du niveau de recomplètement R à partir de la $\mathscr{B}(400; 0{,}25)$ conduit à vouloir recompléter le stock des pare-brise de la gamme A à 114 unités (le risque exact est de 4,86%). On effectue une simulation en supposant que le stock initial est de 20 et que la première livraison attendue est de 100. La simulation montre que le risque encouru en raison de la contrainte de capacité passe de 4,86% à 6,58%.
- **Exemple 2**: Ouvrez le classeur MECANOR-I.xls. Il s'agit d'une analyse de rentabilité d'un dossier d'investissement hors fiscalité et en univers certain.

Année	2012	2013	2014	2015	2016
Coef de décroissance du coût			0,9	0,85	0,85
Coût unitaire		312,00	280,80	238,68	202,88
Prix de vente		500,00	500,00	450,00	400,00
Marge unitaire		188,00	219,20	211,32	197,12
Nb mois d'exploitation en 2013	9				
Coefficient de variation du marché			1,25	0,7	0,5
Marché année pleine		10 000	12 500	8 750	4 375
Investis. + études (au 31/12/2012)	-5 600 000				
Année	2012	2013	2014	2015	2016
FNT (au 31/12/année)	-5 600 000	1 410 000	2 740 000	1 849 050	862 409
Taux d'actualisation	5%	10%	15%	20%	
FNTA	534 903	-75 464	-593 214	-1 036 271	
TRI	9,33%				

- Le remplacement des données certaines par des dires d'experts (probabilités subjectives sur les estimations fournies) permettent, par le biais de @Risk, d'établir une distribution de probabilités du TRI et à déterminer les variables ayant le plus fort impact sur ce résultat.







II MODÉLISATION DES PROCESSUS DE PRODUCTION

Fondamentalement, un système productif est caractérisé

- par un ou plusieurs points d'entrée[1] par lesquels arrivent des «clients» du système productif,
- par un ou plusieurs points de sortie[2] par lesquels sortent des «clients» du système productif, après traitement,
- un ensemble de files d'attente[3] et de processeurs[4] (où les clients sont traités) reliés entre eux et aux points d'entrée et de sortie par des arcs orientés de telle sorte qu'aucun élément du système n'est déconnecté des autres et qu'un client entré dans le système puisse en sortir.

L'analyse de tels systèmes intéresse les spécialistes de la recherche opérationnelle depuis plus d'un demi-siècle, dans des systèmes simplifiés dans lesquels les arrivées sont aléatoires, de même que les temps de traitement, et où le fonctionnement du système est caractérisé par des règles simples (Premier Arrivé - Premier Servi, presque toujours). Ce courant scientifique est celui de la théorie des files d'attente (§ II-1). Les limites de cette approche pour le traitement de système réel sont évidentes mais il a fallu attendre le milieu des années soixante-dix, avec la montée en puissance des ordinateurs, pour voir apparaître des simulateurs de processus exploitant la méthode de Monte Carlo (§ II-2, page 14).

II-1 Théorie des files d'attente

La théorie des files d'attente s'intéresse à des systèmes productifs «très simples». Elle vise principalement à déterminer des conditions de stationnarité, c'est-à-dire des conditions garantissant au système une non-croissance exponentielle d'items dans un stock, et des formules de variables caractérisant le régime de croisière: distribution de probabilités du nombre de clients présents dans le système productif, espérance mathématique du nombre de clients dans le système

1. On reviendra en détail sur les caractéristiques de ces points d'entrée aux pages 25 et suivantes et aux pages 55 et suivantes.
2. On reviendra en détail sur les caractéristiques de ces points de sortie à la page 31 et aux pages 60 et suivantes.
3. On reviendra en détail sur les caractéristiques des files d'attente aux pages 30 et suivantes et aux pages 62 et suivantes.
4. On reviendra en détail sur les caractéristiques des processeurs (que l'on désignera par la suite comme des *activités*) aux pages 30 et suivantes et aux pages 72 et suivantes.

productif E(N) ou en attente dans un stock E(N_q), temps moyen de présence dans le système E(T) ou dans un stock E(T_q)…). Des résultats analytiques ont donc été établis pour différents problèmes caractérisés par une loi d'arrivée, une configuration simple de système productif (processeur unique, processeurs identiques en parallèle, processeurs en série…), lois du temps de traitement d'un item par un processeur (on parle de loi de service dans cette approche théorique) et des disciplines de file d'attente (principalement FIFO).

La figure 5 de la page 14 illustre quelques configuration «simplistes» prises en charge par la théorie des files d'attente et le tableau 1 fournit quelques exemples de résultats analytiques obtenus dans cette approche. Malgré le simplisme des hypothèses, ces résultats présentent plusieurs intérêts.

Tableau 1

Quelques résultats analytiques de la théorie des files d'attente

Nombre **s** de processeurs	Limite du stock	Espérance mathématique du nombre d'items présents dans		Espérance mathématique du temps passé dans		Probabilité d'avoir n items présents dans le système productif $\pi_n = P(N = n)$
		le système productif E(N)	le stock E(N_q)	le système productif E(T)†	le stock E(T_q)	
1	∞	$\frac{\rho}{1-\rho}$	$\frac{\rho^2}{1-\rho}$	$\lambda \cdot E(N)$	$\lambda \cdot E(N_q)$	$[1-\rho] \cdot \rho^n$
	c	$\frac{\rho}{1-\rho} - \frac{(c+1)\rho^{c+1}}{1-\rho^{c+1}}$	$E(N) - (1-\pi_0)$	$\frac{\lambda}{(1-\pi_c)} \cdot E(N)$	$\frac{\lambda}{(1-\pi_c)} \cdot E(N_q)$	$\frac{1-\rho}{1-\rho^{c+1}} \cdot \rho^n$
s processeurs en parallèle	∞	$\frac{\mu}{\lambda} + \frac{\rho\left[1 - \sum_{n=0}^{s-1} \pi_n\right]}{1-\rho}$	$E(N) - \frac{\mu}{\lambda}$	$\lambda \cdot E(N)$	$\lambda \cdot E(N_q)$	$\frac{1}{\sum_{i=0}^{s-1} \frac{(s\rho)^i}{i!} + \frac{(s\rho)^s}{s!(1-\rho)}} \cdot \frac{(s\rho)^n}{n!}$

†. En l'absence de contrainte de capacité sur les stocks, la relation $E(T) = \lambda \cdot E(N)$ (connue sous le nom de **loi de Little**) est valable pour le système productif ou le stock alimentant le ou les processeurs. En cas de contrainte sur le stock, le correctif $(1-\pi_c)$ permet de passer de la demande exprimée à la demande satisfaite.

- Tout d'abord, ces relations permettent de comprendre le danger des raisonnements en moyenne. Avec des arrivées de Poisson caractérisées par un intervalle moyen de 10' entre deux arrivées successives, et une loi de service exponentielle caractérisé par un temps moyen de 6', l'utilisation de ces relations montre que le temps moyen passé dans le système est de 15' et que le processeur a 40% de chances d'être inoccupé (voir la feuille «File d'attente» du classeur Introduction_Simulation.xls);
- Ces relations se sont avérées très utiles pour traiter quelques problèmes importants de systèmes productifs simples, comme ceux liés à la conception et du pilotage de traitements informatiques de systèmes d'exploitation.

Cette approche de la théorie des files d'attente reste malgré tout d'un intérêt opérationnel très limité tout d'abord parce que ces résultats analytiques sont valables en régime de croisière, celui-ci étant long à atteindre. Mais, plus encore, on ne dispose de résultats que pour des hypothèses / combinaison d'hypothèses simplistes. Les problèmes que l'on rencontre sur le terrain en particulier dans les processus de production de services ne s'accommodent pas des hypothèses de

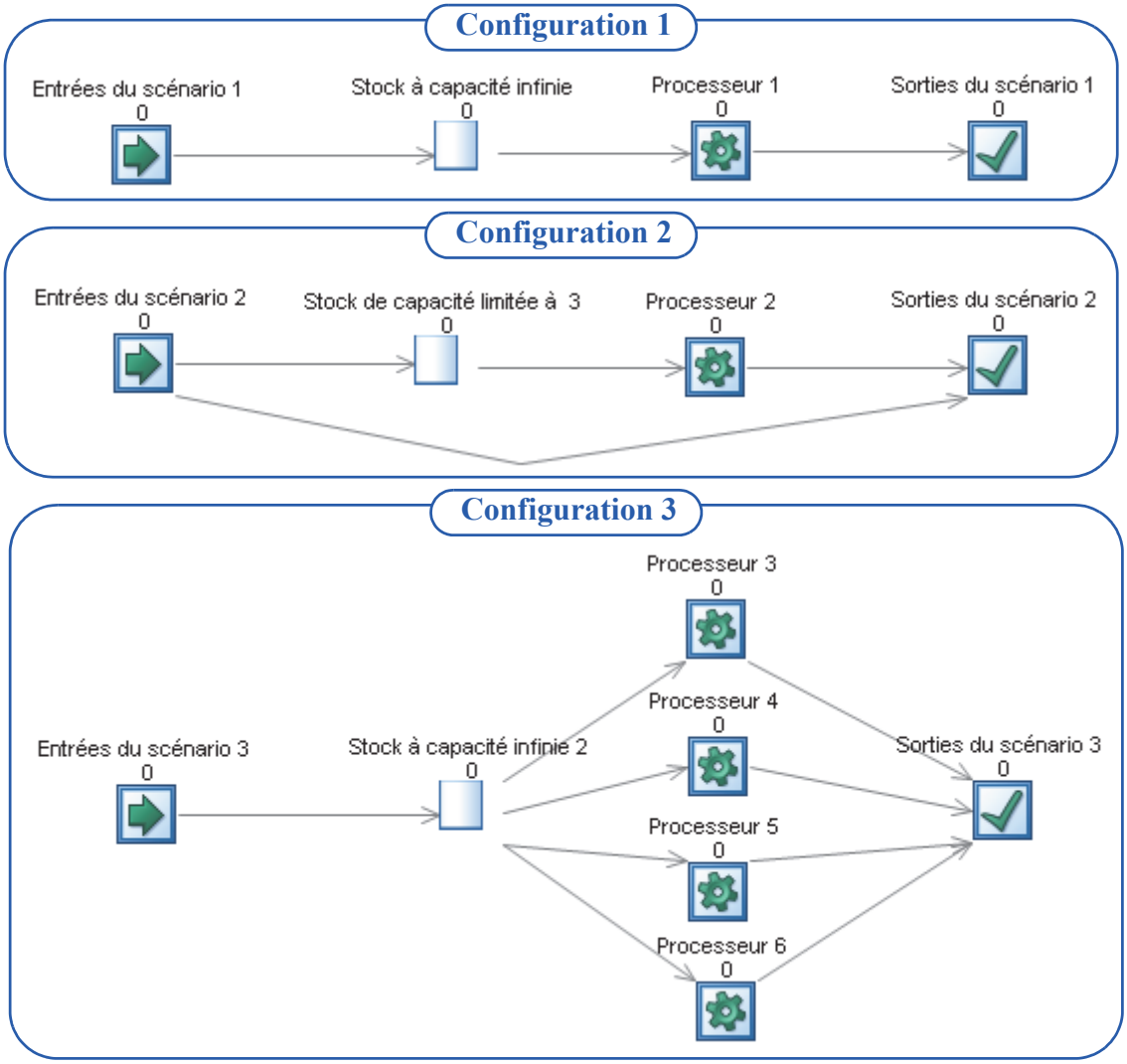


Figure 5

Exemples de configurations simples de systèmes productifs

travail de la théorie des files d'attente (exemple: configuration 1 avec limite de 3 items en stock). La prédictibilité de l'impact de la variation des paramètres du système productif est tout aussi peu intuitive. Dans l'exemple Impredictibilite_processus_simple.S8, si l'on passe de 6 à 8 clients par heure (augmentation de 25% des arrivées, l'intervalle moyen passant de 10' à 7,5'), ces relations analytiques montrent que le temps moyen passé dans le système double tandis que la probabilité que le processeur soit inoccupé tombe à 20%, à condition que certaines contraintes ne pèsent sur le stock; si tel n'est pas le cas (temps maximal d'attente, par exemple), aucune prévision du comportement du système ne peut être établie sans faire appel à la simulation.

II-2 Usage de la méthode de Monte Carlo dans les simulateurs de processus

Dans la modélisation d'un processus de production, l'utilisation de variables aléatoires pour décrire le fonctionnement d'un système est incontournable. On les retrouve principalement pour décrire les arrivées dans le système, les temps de traitement des processeurs et, dans certains aiguillages en sortie d'un processeur

(par exemple, un contrôle de qualité, opération effectuée par un processeur, envoie un produit venant d'être contrôlé soit dans un stock de produits corrects, soit dans un stock de produits non conformes, la probabilité de chaque direction étant prédéterminée). Les constantes sont décrites dans les simulateurs de processus comme des variables aléatoires suivant une loi certaine.

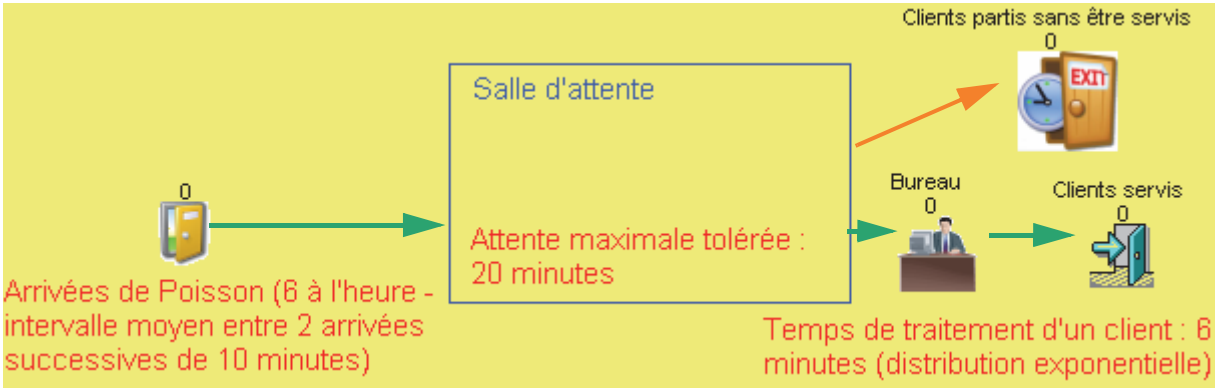
Le fonctionnement d'un **simulateur de processus à événements discrets** se base sur le modèle du système productif et les réalisations des variables aléatoires pour faire avancer l'horloge. Ces réalisations de variables aléatoires permettent de définir des événements à venir. Illustrons ce principe à partir de la première configuration de la figure 5 de la page 14; les valeurs numériques utilisées ici doivent toutes être considérées comme des réalisations de variables aléatoires.

- Initialisation du système: un item 1 arrive en $t = 2'$; immédiatement l'arrivée suivante est générée, elle se produit dans 3' ; en t = 2', la liste des dates des événements à venir est donc {5}.
- L'item 1 arrive dans le stock en $t = 2'$ (on suppose, pour simplifier ici, les temps de transport nuls) et le quitte immédiatement pour arriver dans le processeur qui est libre, un temps opératoire de 1' est calculé; la liste des dates des événements à venir est donc {3; 5}.
- L'horloge du simulateur passe à la date de l'événement à venir le plus précoce $t = 3$, associée à la sortie de l'item 1 du processeur qui devient libre → la liste des dates des événements à venir est alors {5} ; le simulateur calcule le temps de traitement, 4' → la liste des dates des événements à venir est alors {5; 7}
- L'horloge du simulateur passe à la date de l'événement à venir le plus précoce associé à l'arrivée de l'item 2 → $t = 5$ → la liste des dates des événements à venir est alors {7} ; le simulateur calcule le temps séparant la prochaine arrivée, 1' → la liste des dates des événements à venir est alors {6; 7}.
- En $t = 5$ toujours, l'item 2 va dans le stock (vide) puis arrive dans le processeur; le temps opératoire est calculé, il est de 4'; la liste des dates des événements à venir est donc {6; 7; 9}.
- L'horloge du simulateur passe à la date de l'événement à venir le plus précoce associé à l'arrivée de l'item 3 → $t = 6$ → la liste des dates des événements à venir est alors { 7; 9} ; le simulateur calcule le temps séparant la prochaine arrivée, 5' → la liste des dates des événements à venir est alors {7; 9; 11} ;
- En $t = 6$ toujours, l'item 2 va dans le stock (vide) puis il y séjourne puisque le processeur ne se libérera qu'en t = 9; la liste des dates des événements à venir est donc toujours {7; 9; 11}.
- L'horloge du simulateur passe à la date de l'événement à venir le plus précoce associé à la fin du traitement de l'item 2 par le processeur et donc la libération du processeur → $t = 7$; l'item 2 sort et l'item 3 quitte le stock pour rentrer dans le processeur → détermination du temps de traitement de l'item 3, 2' → la liste des dates des événements à venir est donc toujours {9; 9; 11}
- etc.

II-3 Imprédictibilité du comportement d'un système productif

Une cartographie de flux visualise, sur une représentation simplifiée d'un système productif existant (plan avec visualisation des principaux postes de production et lieux de stockage), les chemins qu'empruntent différents flux homogènes d'objets ou de personnes passant par différents postes de production pour y subir des traitements, avant de quitter le système. Cette représentation est classique en

analyse de processus, fournit une vision statique ne permettant pas, même avec des temps moyens, d'anticiper le comportement d'un système productif en raison des interactions se produisant dans le système. Illustrons ce point avec l'exemple suivant pour lequel la théorie des files d'attente ne peut apporter de réponse.



- t = VA du temps de séjour d'un client servi; t_i du client i est **corrélé** à t_{i+1} du client $i+1$; $\bar{d}$ est le nombre moyen de client / heure.
- Temps Opératoire d du bureau est une VA; simulation sur 5 jours ⇒
  - $\bar{d} = 6 \Rightarrow \bar{t} = 10{,}7$, $\sigma_t = 9{,}8$, $t_{max} = 47{,}5$, P $(t < 10) = 56\%$, % DNS $= 10{,}3\%$
  - $\bar{d} = 9 \Rightarrow \bar{t} = 16{,}0$, $\sigma_t = 13{,}1$, $t_{max} = 73{,}2$, P $(t < 10) = 40\%$, % DNS $= 23{,}5\%$

D'une manière générale, la simulation est indispensable pour améliorer **efficacité/efficience** des processus de production de biens ou de services.

II-4 Exemples d'utilisation de ces concepts par Simul8

Pour commencer, vous pouvez regarder rapidement les deux simulations suivantes, de la bibliothèque d'exemples de Simul8. Ce cours s'appuiera sur sa propre bibliothèque d'exemples. Vous devez mettre ce fichier pdf de cours dans un dossier contenant tous ces exemples pour pouvoir utiliser tous les liens hypertextuels vous permettant d'ouvrir ces exemples.

- Simulation d'un fastfood (fastfood.S8 appartenant à la base d'exemples de Simul8)
- Simulation d'un processus industriel (bottling.S8 appartenant à la base d'exemples de Simul8)

Chapitre II

Les bases de la simulation d'un processus

On adoptera ici une démarche inductive en s'appuyant sur un exemple pour montrer progressivement comment on arrive à modéliser le fonctionnement d'un système productif, puis à le simuler. Les chapitres suivants seront consacrés à l'approfondissement des «objets, au sens large» mobilisés dans cette modélisation. On partira de l'analyse du fonctionnement d'un cabinet médical spécialisé en imagerie médical, comportant deux bureaux de médecin, une secrétaire et une salle de prise de clichés radio exécutés par un opérateur spécialisé. Cette analyse a pour objectif d'examiner la possibilité de l'arrivée d'un troisième médecin dans ce cabinet, le problème ne se posant pas en termes de place (le réaménagement est techniquement réalisable sans trop de dépenses) mais en termes de capacité des ressources partagées (secrétaire et opérateur) à absorber un surcroît de charge de travail mais aussi en termes de conséquences sur le temps passé par les patients dans ce cabinet.

I Un préalable : la cartographie «papier» du processus

I-1 Exemple introductif (version statique)

I-1.1 Énoncé

Cette version statique du problème ne comporte aucune information temporelle (relatives aux arrivées de patient ou de temps de traitement) parce que ces informations sont sans utilité dans le travail de représentation graphique du processus.

Le cabinet médical étudié appartient à deux radiologues, les docteurs Dupond et Martin qui reçoivent sur rendez-vous des patients dont certains arrivent avec une radio. Certains patients arrivent avec des radios ; pour simplifier, on supposera qu'il ne s'agit alors que de fournir un diagnostic et qu'aucune radio complémentaire n'est nécessaire. Le patient est accueilli par une secrétaire médicale qui prépare son dossier pour le praticien concerné et envoie le patient dans la salle d'attente. C'est également elle qui s'occupe de percevoir le règlement. Enfin, elle filtre les appels téléphoniques des clients souhaitant parler aux docteurs et elle s'occupe de la prise de rendez-vous téléphonique. Le patient en salle d'attente est appelé par le médecin avec lequel il a rendez-vous, dès que celui-ci est libre. S'il a une radio, le médecin fait sa consultation puis envoie le patient à la secrétaire. Dans le cas contraire, après un premier examen, il envoie le patient au cabinet de radiologie. L'opérateur, s'il est libre, fait immédiatement une radio puis envoie le patient dans la salle d'attente. Dans le cas contraire, le patient attend dans une petite salle d'attente attenante au cabinet de radiologie. Une fois muni des radios

qui viennent d'être faites, le patient qui attend dans la salle d'attente est repris en priorité par le médecin qui a demandé ces radios. On est alors ramené au cas de la prise en charge par le praticien d'un patient avec radio.

- Cabinet médical avec docteurs Dupond et Martin, secrétaire médicale (accueil, prise de rendez-vous téléphoniques, encaissement)
- Accueil, salle d'attente consultation médicale, 2 cabinets de consultation, salle d'attente radio, cabinet de radiologie
- RDV toutes les 20'pour chaque médecin ; patients avec ou sans radio ; arrivée aléatoire des appels téléphoniques
- patient passe d'abord par accueil, va en salle d'attente, est appelé par le médecin avec lequel il a pris RDV; s'il a une radio, diagnostic puis envoi à secrétaire, sinon envoyé en radio, puis en salle d'attente avec la radio et est ensuite repris par le médecin dans les conditions d'un patient avec radio.

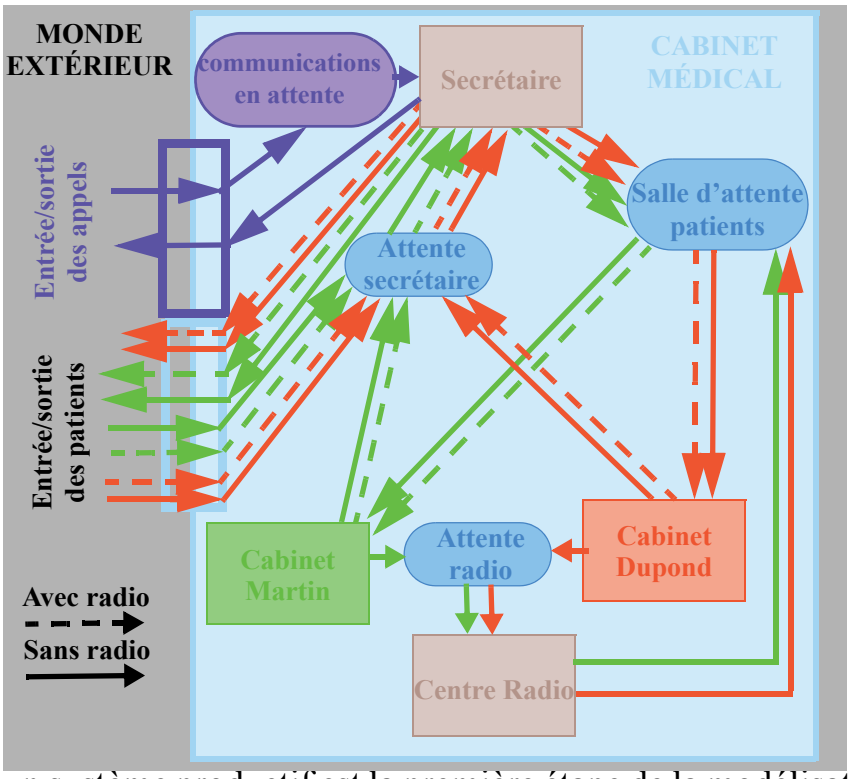
I-1.2 Cartographie des flux

On va commencer par une cartographie des flux avant de procéder à une cartographie des processus. Une cartographie de flux visualise, sur une représentation simplifiée d'un système productif existant (plan avec visualisation des principaux postes de production et lieux de stockage), le chemin qu'emprunte un flux d'objets passant par différents postes de production pour y subir des opérations, avant de quitter le système. Ici les postes de production sont le poste de travail de la secrétaire et ceux des médecins et de l'opérateur radio ; les stocks sont les lieux où attendent les patients. Il est utile d'utiliser une convention graphique (couleur…) pour visualiser les trajets effectués par les quatre flux de clients se rendant au cabinet. Vous noterez que la cartographie des flux représentée à la page suivante comporte un premier stock devant le poste de travail de la secrétaire, où les patients attendent que la secrétaire puisse s'occuper d'eux (en pratique, cela peut être une chaise[1], et un stock correspondant à la mise en attente d'appels téléphonique. En effet, si un poste de travail est occupé, il faut bien que les «clients du poste de travail» en attente de traitement aient un lieu où attendre.

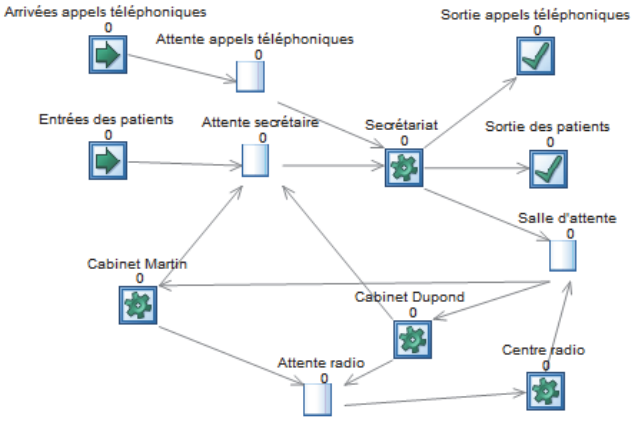
I-2 Création de la cartographie des processus sous Simul8

La cartographie des flux devient rapidement inexploitable dès que le nombre de routes possibles devient grand. On passe à la représentation de la cartographie des processus qui se caractérise d'une part par la fusion de tous les flux allant directement d'un point à un autre et d'autre part par le fait que cette représentation est utilisée par un logiciel capable de piloter la circulation d'objets à traiter par le système productif considéré (par la suite on retiendra le terme d'**item**) en fonction, notamment, de ses caractéristiques. L'établissement de la cartographie des

1. Dans de nombreux cabinets, le patient attend en salle d'attente que la secrétaire se libère. La circulation des flux est alors différente de celle représentée ici.

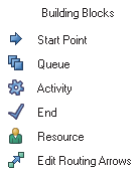


processus d'un système productif est la première étape de la modélisation du fonctionnement de ce système pour aboutir à sa simulation. La représentation suivante, obtenue avec Simul8 correspond à la cartographie des processus qui fusionne les flux de la cartographie des flux montrée ci-dessus.



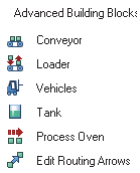
Nous allons expliquer ici comment créer cette représentation statique du système de production de service. On reviendra en section II, page 24, avec plus de détail, sur ce processus de création du modèle du cabinet médical, après avoir introduit la version complète du problème.

Il vous est vivement conseillé d'effectuer de votre côté le travail de création décrit ici pour pouvoir effectuer sans difficulté le travail demandé après. Ouvrez Simul8 et cliquez sur l'option **Create New** pour créer une nouvelle simulation. Commencez par sauver cette simulation vierge et pensez à sauver périodiquement votre travail pour éviter de mauvaises surprises (pas de sauvegarde automatique).



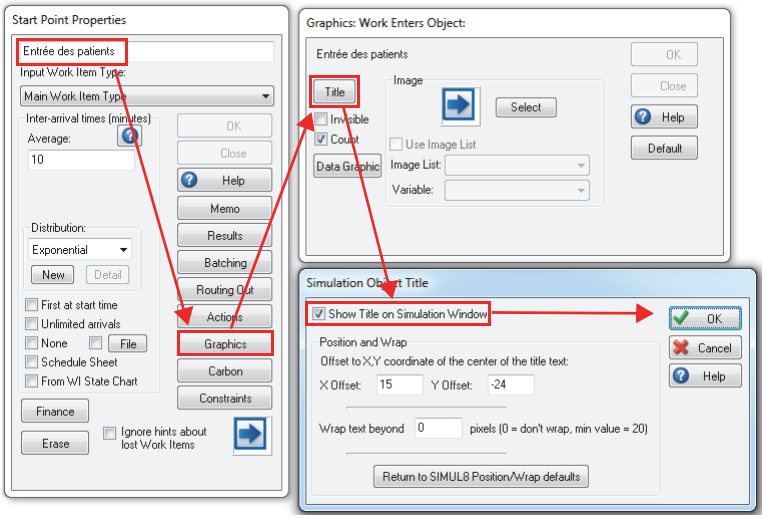
Vous êtes sur l'onglet Home. La partie gauche de l'écran regroupe les outils de création du système productif. Les quatre premiers outils permettent de créer les **points de passage** par lesquels peut passer un item, la circulation d'un point à un autre n'étant possible que si une flèche relie ces deux endroits.

- L'outil Start Point permet de créer un **point d'entrée**.
- L'outil Queue permet de créer une **file d'attente** (un stock peut toujours être considéré comme un type particulier de file d'attente).
- L'outil Activity permet de créer une **activité** ce qui correspond généralement un poste de travail dans lequel un traitement est effectué sur l'item en transit dans ce poste. Une activité peut aussi être un point de passage virtuel permettant de modifier certaines caractéristiques de l'objet en transit, en particulier son orientation.
- L'outil End permet de créer un **point de sortie** des items.
- L'outil Resource permet de créer des **ressources partagées** (en général du personnel) entre plusieurs *activités*, le travail d'une activité étant conditionné par la présence de la ressource (ou des ressources, dans certains cas). Dans l'exemple retenu, aucune ressource n'est partagée: la secrétaire est «attachée» en permanence au poste de travail (activité) secrétariat, au même titre que la chaise, le téléphone, l'ordinateur, etc. Aucune ressource n'est alors à déclarer, le poste de travail étant une entité de travail autonome. Le sens de partage est donc différent de celui rencontré dans le langage courant («les deux médecins se partagent la secrétaire»).
- Signalons, pour terminer que le bouton Advanced Building Blocks situé en bas à gauche de la fenêtre permet d'utiliser d'autres composants spécifiques (convoyeur, véhicules…) qui ne sont pas tous accessibles dans la version étudiante.



Par défaut le mode de visualisation des itinéraires possibles (flèches) est activé Le bouton Edit Routing Arrows est à enfoncer lorsque l'on souhaite créer de nouveaux chemins entre deux des quatre points de passage.

On va commencer par décrire les points de passage et itinéraires que peuvent emprunter les items «patients». On complétera ensuite la représentation par la description des points de passage et chemins que peuvent emprunter les items «appels téléphonique».



Les clients arrivent au cabinet (les items arrivent dans le système productif étudié) en franchissant la porte d'entrée du cabinet (... par un point d'entrée). Utilisez l'outil Start Point pour créer ce point d'entrée. En cliquant sur ce bouton, le pointeur de la souris (ou du trackpad) se transforme en passant de l'icône d'une flèche à celle du signe +. Déplacez le pointeur là où vous voulez créer le point d'entrée et collez l'icône du point d'entrée par un clic-gauche. Un double clic-gauche sur le point d'entrée ouvre la fenêtre ci-contre. Changez le nom par défaut (*Start Point 1*) par «Entrée des patients». Par défaut, le nom donné à un point d'entrée (ou une file d'attente) n'est pas affiché. Pour forcer l'affichage, cliquez sur le bouton Graphics qui ouvre une fenêtre dans laquelle vous cliquez sur le bouton Title qui ouvre une nouvelle fenêtre dans laquelle vous cochez l'option Show Title on Simulation Window.

Refermez successivement toutes les fenêtres, ce qui vous conduit à observer maintenant sur la feuille du modèle, le point d'entrée suivant 

.

Vous devez poursuivre la construction de votre modèle en créant une file d'attente des patients devant le secrétariat, pour permettre à un patient qui arrive puisse se «mettre quelque part» si la secrétaire est occupée au moment de son arrivée (dans de nombreuses situations réelles, ce patient prend un ticket et s'installe dans la salle d'attente, ce qui conduit à une modélisation légèrement différente). Pour ce faire, utilisez l'outil Queue en suivant les mêmes principes que précédemment: double-click sur l'icône de la file d'attente, changement du nom par défaut, visualisation du nom en passant par la fenêtre ouverte par le bouton Graphics, dans laquelle vous cliquez sur le bouton Title qui ouvre une nouvelle fenêtre dans laquelle vous cochez l'option Show Title on Simulation Window. On obtient alors 

.

Pour créer le chemin possible entre ce point d'entrée et cette file d'attente, vous devez cliquer sur le bouton Edit Routing Arrows. Le pointeur se transforme alors en une flèche pleine horizontale. Il faut alors la positionner sur le point d'origine (ici le point d'entrée) et effectuer un clic-gauche. Un point d'interrogation se place alors au-dessus de la flèche horizontale du pointeur que vous déplacez sur le point d'arrivée choisi (ici la file d'attente); l'icône du pointeur se transforme alors en une flèche épaisse. Il suffit alors d'un clic-gauche pour achever ce processus de création. Pour annuler le mode de création, vous devez cliquer sur le bouton

qui reprend son aspect initial (

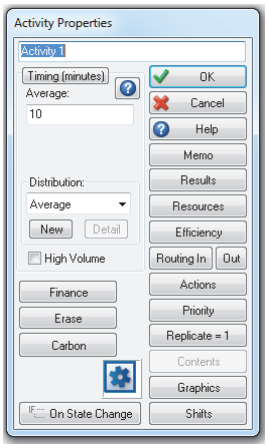
). Vous pouvez, sans problème, effectuer plus tard cette création d'itinéraire possible, en la groupant avec d'autres.

Il faut ensuite créer le poste de travail Secrétariat, avec l'outil 

. Un double clic-gauche ouvre la fenêtre ci-contre. Il suffit seulement de modifier le titre par défaut en le remplaçant par «Secrétariat». Après validation en cliquant sur le bouton 

, il faut créer la flèche reliant la file d'attente à cette activité, en suivant la même procédure que précédemment, ce qui conduit à la représentation suivante 

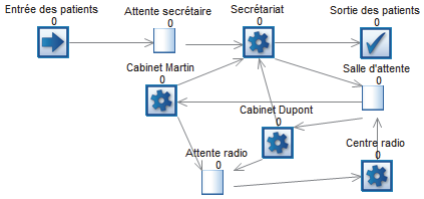
.



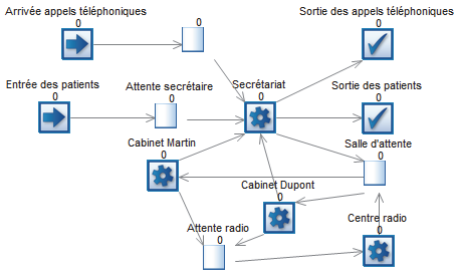
On crée ensuite, de la même façon, les autres activités (Cabinet Martin, Cabinet Durand, Centre Radio), les autres files d'attente (Salle d'attente, Attente radio) et les flèches visualisant les chemins que peuvent emprunter les patients.

Reste enfin à créer le point de sortie des patients. Dans la cartographie des flux, les patients entrent et sortent par une même porte. Dans la modélisation du processus il convient de dissocier ces deux rôles de la porte qui joue maintenant le rôle d'un point de sortie, ce que l'on va créer avec outil 

.



Pour terminer, il faut créer un point d'entrée spécifique pour les appels téléphoniques, une file d'attente permettant de mettre ces appels en attente si nécessaire, et un point de sortie spécifique pour ces items.



I-3 Version complète de l'exemple du Cabinet médical (version dynamique)

Repartons de la modélisation statique de ce cas pour y introduire tous les éléments qui vont permettre une reconstitution dynamique du processus. Dans cette perspective, l'énoncé initial est modifié[1]. Nous allons utiliser ce mini-cas pour introduire progressivement les fondements de la modélisation dans la section II, page 24.

Le cabinet médical étudié ici a été créé par deux radiologues, les docteurs Dupond et Martin qui reçoivent sur rendez-vous des patients dont certains arrivent avec une radio. Les rendez-vous sont pris sur la base d'un rendez-vous toutes les vingt minutes. La réputation de ces praticiens fait que tous les rendez-vous sont pris, ce qui fait 24 x 5 = 120 patients par semaine pour chaque docteur. Le premier patient est pris à 9 heures, le dernier de la matinée a un rendez-vous fixé à 11h40; le premier patient de l'après-midi arrive à 14 heures et le dernier a un rendez-vous fixé à 18h40. On supposera que cabinet est ouvert de 9 heures à 19h30 (pour avoir le temps de traiter le dernier patient) et que le personnel du cabinet est en pause après traitement du dernier patient du matin (cette pause pouvant être considérée comme étant de 2 heures). Les patients arrivent en moyenne 5 minutes en avance avec une dispersion autour de cette avance de 2 minutes; la loi normale $\mathcal{N}(-5; 2)$ peut être utilisée sans problème pour résumer l'aléa sur l'avance ou le retard d'un patient par rapport à sa date de rendez-vous. Certains de ces patients (20% en moyenne) arrivent avec des radios; pour simplifier, on supposera qu'il ne s'agit alors que de fournir un diagnostic et qu'aucune radio complémentaire n'est nécessaire (mais une autre hypothèse aurait pu être retenue).

Le patient est accueilli par une secrétaire médicale qui prépare son dossier pour le praticien concerné et envoie le patient dans la salle d'attente. C'est également elle qui s'occupe de percevoir le règlement. Enfin, elle s'occupe de la prise de rendez-vous téléphonique[2]. La distribution de probabilité de la durée de chacune de ces trois activités est donnée dans le tableau 2.

Le patient en salle d'attente est appelé par le médecin avec lequel il a rendez-vous, dès que celui-ci est libre. S'il a une radio, le médecin fait sa consultation puis envoie le patient à l'accueil. Dans le cas contraire, après un premier examen, il envoie le patient au cabinet de radiologie. L'opérateur, s'il est libre, fait immédiatement une radio puis envoie le patient dans la salle d'attente. Dans le cas contraire, le patient attend dans une petite salle d'attente attenante au cabinet de radiologie. Une fois muni des radios qui viennent d'être faites, le patient qui attend dans la salle d'attente est repris en priorité par le médecin qui a demandé ces radios. Pour simplifier, on considérera que la durée de la consultation qui s'en suit est similaire à celle d'un patient arrivé au cabinet avec ses radios. Les lois de probabilité de ces activités sont également consignées dans le tableau 2

1. Les données de ce cas ne prétendent pas refléter la réalité.

2. Pour simplifier la modélisation, on supposera que l'appel téléphonique est prioritaire sur les autres activités mais que la secrétaire achève d'abord l'activité en cours.

Tableau 2 : Distributions de probabilité†

Variable		Loi
Arrivées des patients	Décalage par rapport à l'heure de rendez-vous	$\mathcal{N}(-5\,;\,2)$
	Arrivées des patients du docteur Martin et du docteur Dupond	Arrivées‡ toutes les 20 minutes à partir de 9 heures jusqu'à 11h40 et de 14 heures jusqu'à 18h40
	Possession d'une radio par un patient arrivant dans le cabinet	$\mathcal{B}$ (n = 1 ; p = 0,2)
Travail de la secrétaire médicale	Accueil d'un nouveau patient	Exponentiel (1)
	Règlement du patient	Triangulaire (Min = 1 ; M_o = 4 ; Max = 6)
	Réponse à un appel téléphonique	Exponentiel (1)
Consultation & radio	Durée d'une consultation avec radio	Triangulaire (Min = 8 ; M_o = 10 ; Max = 13)
	Durée d'une consultation sans radio	Triangulaire (Min = 5 ; M_o = 7 ; Max = 14)
	Radiographie	Triangulaire (Min = 5 ; M_o = 8 ; Max = 10)
Arrivées des communications téléphoniques*		Exponentiel (10)

†. Les paramètres de temps ont pour unité la minute.
‡. Le décalage est supposé ne pas jouer pour le premier rendez-vous de chaque demi-journée.
*. Une communication en attente pendant plus d'une minute est perdue

Il s'agit d'étudier ici l'impact de l'adjonction possible d'un troisième médecin dans ce cabinet (en termes d'efficacité et d'efficience).

II Modélisation du processus de l'exemple du cabinet médical

Avant de commencer cette modélisation, il faut réfléchir, en partant de la cartographie des flux et de l'énoncé sur les classes d'objets, au sens large, nécessaires à la modélisation d'un processus. Certains sont liés à la circulation des «clients» du système productif, que l'on désignera ici sous le terme générique d'**items** (quels sont-ils?); d'autres sont liés à la caractérisation de ces items (lesquels?); enfin les derniers sont liés à la prise en compte du temps dans la mise en œuvre du processus (lesquels?).

D'une manière générale, il est vivement conseillé de procéder à une modélisation en deux temps, tant que votre niveau d'expertise dans ce domaine n'est pas suffisant. Dans un premier temps, ne vous préoccupez pas de la gestion du temps (lois d'arrivées aux points d'entrée, temps de traitement des items aux activités), pour vous concentrer exclusivement sur les cheminements possibles des items dans ce système de production de services. Cette description doit être exhaustive à la fin de cette première étape. C'est ce que nous avons fait en section I-2, page 18. Le paramétrage par défaut permet une circulation des items dans le système productif. Dans ce contexte, la simulation de votre modèle, indispensable pour

vérifier que les itinéraires empruntés par les items en fonction de leurs caractéristiques sont corrects, ne peut que conduire à l'observation d'un comportement aberrant (accumulation dans certaines files d'attente). Donc, pas de panique ! Dans un second temps, il faudra remplacer le paramétrage par défaut des lois d'arrivées aux points d'entrée et des temps opératoires des activités, en tenant compte des données et hypothèses formulées.

II-1 Modélisation de la circulation des flux dans la simulation

Les principes de base de création de la modélisation d'un processus de production d'un bien ou d'un service ont été présentés à la page 18. Détaillons maintenant la création et la caractérisation des différents points de passage d'un item dans le système productif étudié.

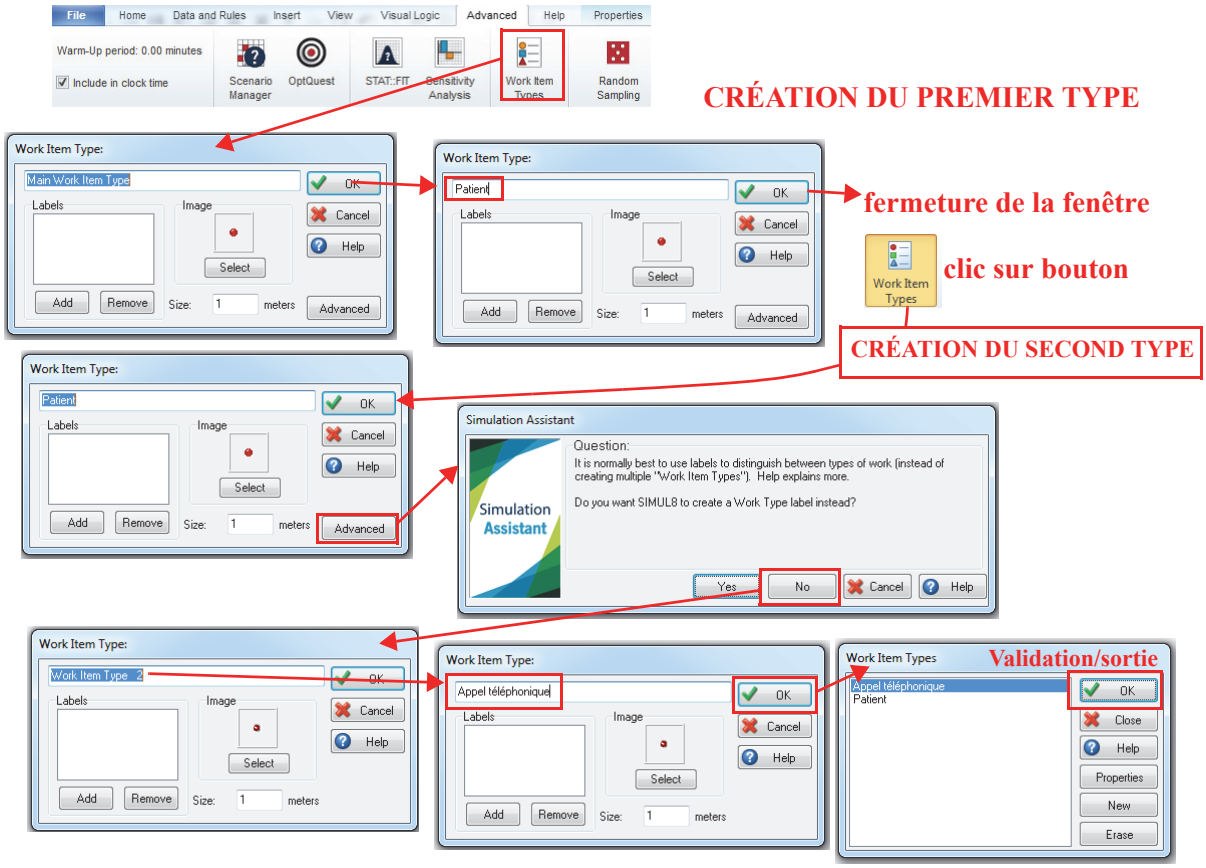
II-1.1 Création des points d'entrée

La création du point d'entrée a été expliquée en page 21. Pour des raisons que l'on évoquera ultérieurement, on crée un point d'entrée par docteur. On commence par créer le point d'entrée des patients pour Dupond; après caractérisation par des labels, on créera le point d'entrée des patients pour Martin par copier-coller du point d'entrée Dupond, puis on adaptera les caractéristiques non communes. L'item par défaut est du type 

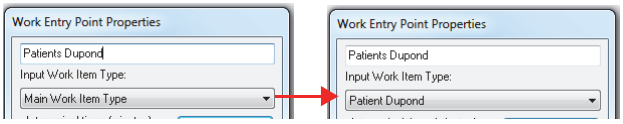
. Nous allons maintenant fermer la fenêtre du point d'entrée et créer plusieurs types d'item

II-1.2 Création de plusieurs types d'items.

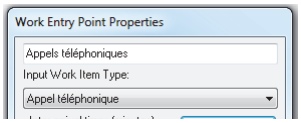
Simul8 propose un seul **Type d'items** par défaut. Pour en créer plusieurs (ici, deux, le «patient» et l'«appel téléphonique»), on procède comme suit.



On crée ensuite, de la même façon, le type d'item «Appel téléphonique». Ces trois types d'items étant créés, on va maintenant modifier l'item par défaut du point d'entrée qui vient d'être créé (Patients Dupond).



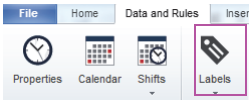
Maintenant on peut créer le point d'entrée «Appels téléphoniques» et lui affecter «Appel téléphonique» comme type d'item



II-1.3 Caractérisation des items entrants

Il existe deux possibilités pour créer des labels:

- utiliser l'option Labels de l'onglet Data and Rules de la barre de menu située en haut de l'écran, ce que nous commencerons par illustrer;

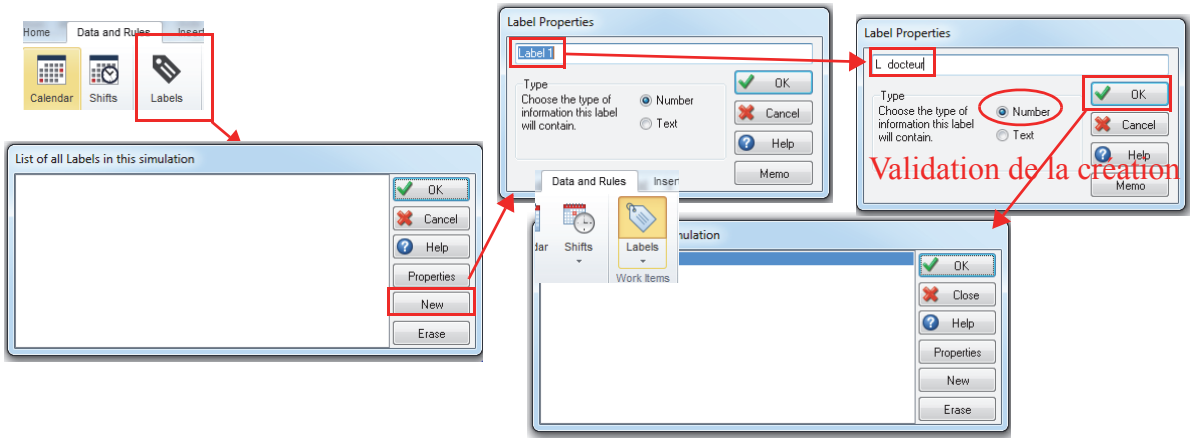


- utiliser l'option de création de nouveaux labels dans la fenêtre ouverte par le bouton Add a Label to Change de la fenêtre ouverte par le bouton Actions de la fenêtre d'un point d'entrée ou d'une activité, ce que nous illustrerons à la page 27).

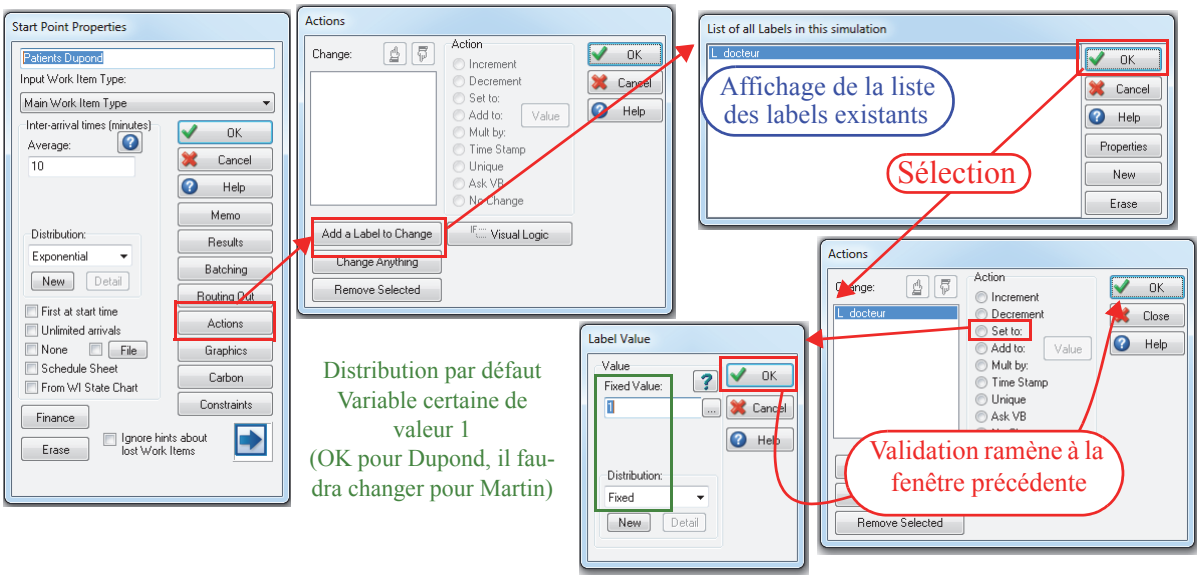
On va commencer par caractériser les patients par les *labels* «L docteur»,

«L Possession radio» et «L Sortie Secrétariat», à des fins d'aiguillage des items (*labels* «L docteur», «L Possession radio» et «L Sortie Secrétariat») et permettre la définition de la durée d'une opération (*labels* «L Possession radio» et «L Sortie Secrétariat»). Il est conseillé d'adopter une convention pour distinguer facilement la nature des objets créés. Dans cette modélisation, on a retenu le préfixe «L» pour les labels et le préfixe «D» pour les distributions. La procédure est la suivante.

On va commencer par créer directement le *label* «L docteur» qui va permettre de distinguer les patients selon les praticiens avec lequel ils ont rendez-vous. On créera les labels suivants directement dans les points d'entrée et les activités, ce qui revient au même.,



Il faut ensuite affecter le *label* «L docteur» aux patients du point d'entrée «Patients Dupond». Il est préférable, comme on le verra, d'utiliser un numéro (1 pour Dupond et 2 pour Martin), plutôt qu'un libellé.

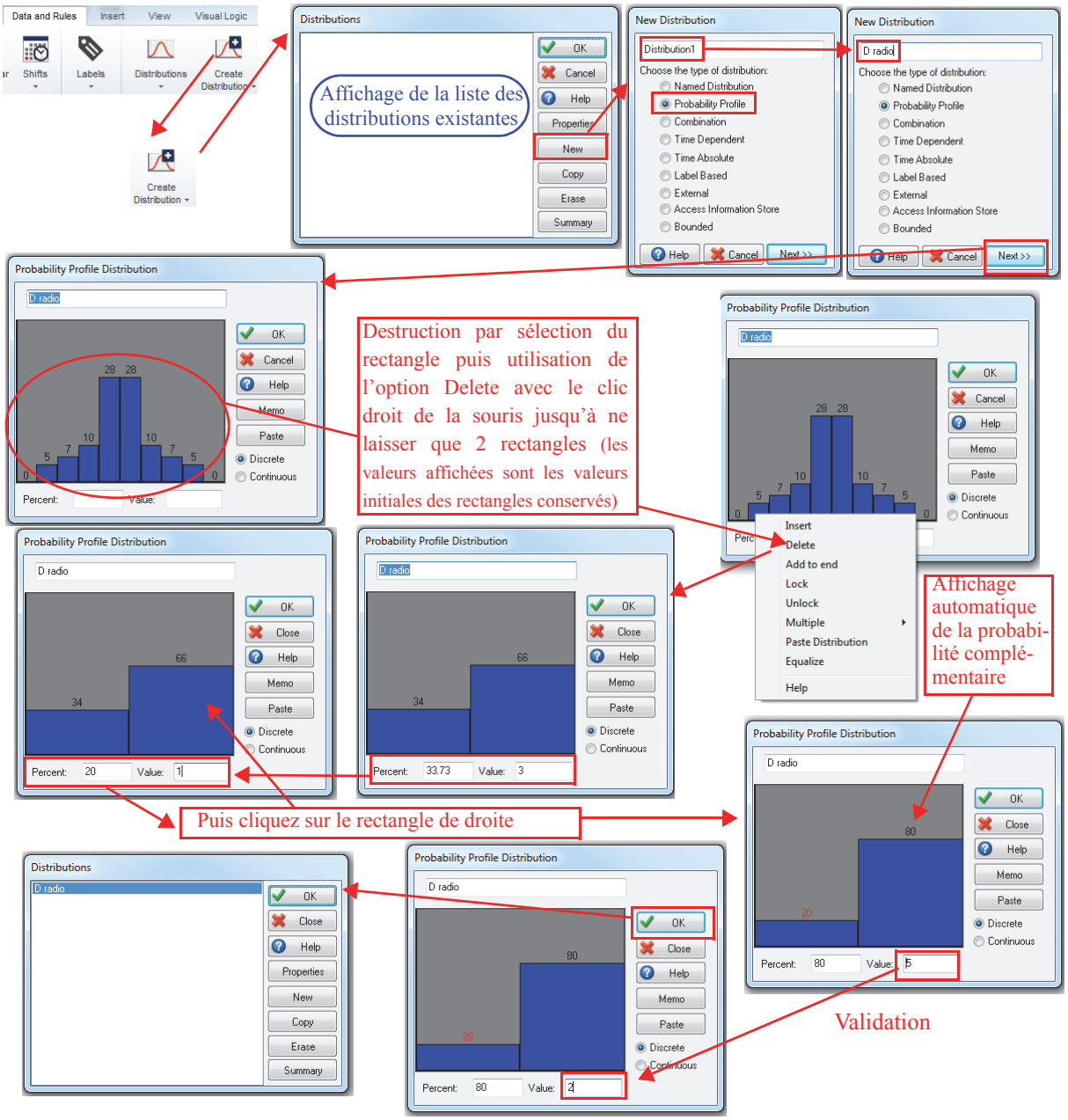


Après la recopie du point d'entrée du Docteur Dupond pour créer celle du Docteur Martin, il faudra changer la valeur de ce *label* pour les items correspondant aux patients ayant rendez-vous avec le Docteur Martin.

Le patient est également caractérisé par le fait qu'il possède ou non une radio prise avant son rendez-vous ce qui sera défini par le *label* «L Possession radio»

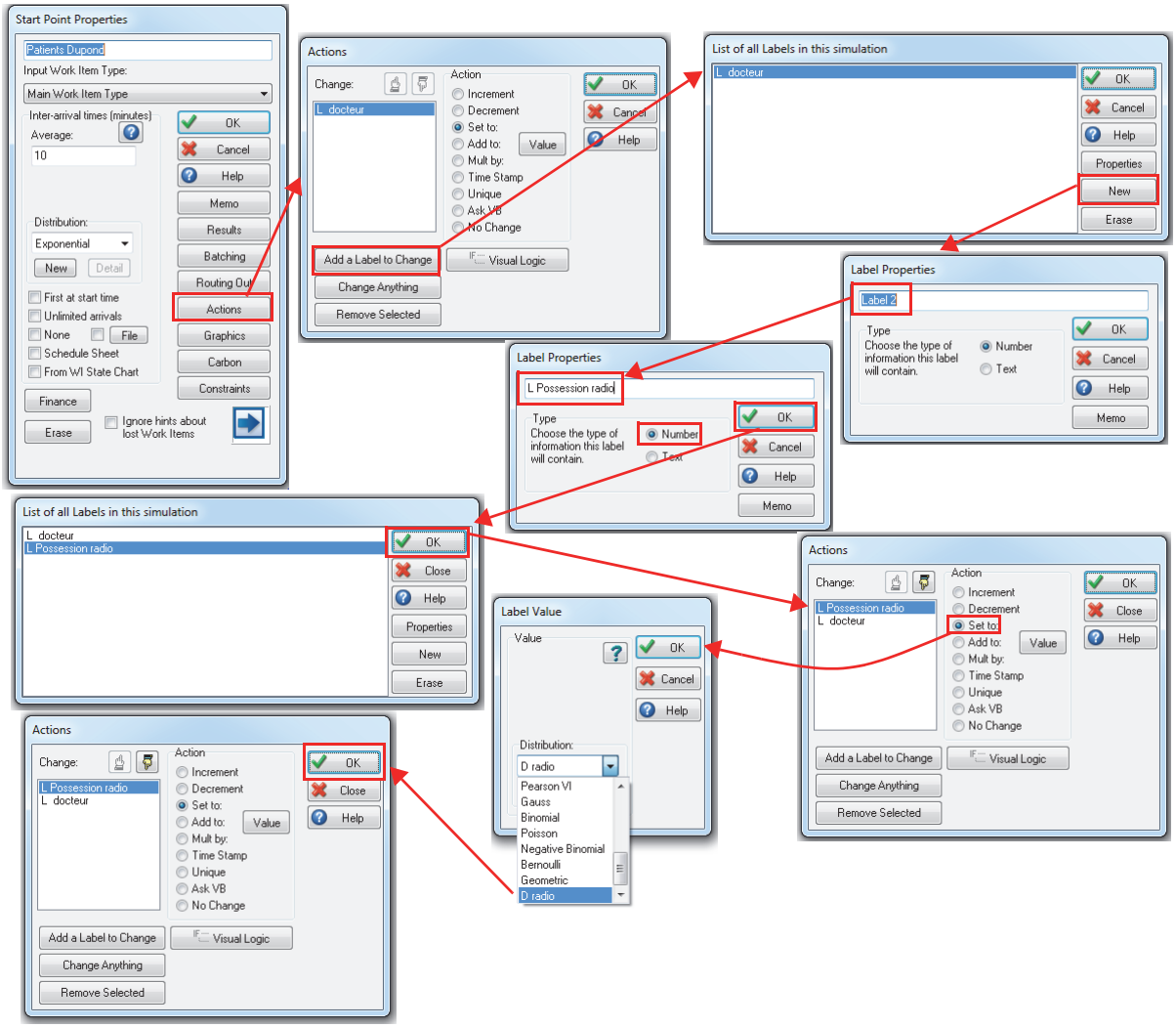
attaché à chaque item entrant. La valeur de ce *label* est définie aléatoirement, chaque patient ayant 20% de chances d'arriver avec une radio. La valeur de ce *label* sera donc l'occurrence d'une variable aléatoire quantitative, correspondant à un numéro d'ordre. Compte tenu de l'usage qui sera fait de ce *label* dans l'orientation d'un item à la sortie d'une activité, la variable aléatoire sera une variable discrète positive qui prendra la valeur 1 si le patient possède une radio et 2, dans le cas contraire. Il faut commencer par créer cette variable aléatoire. Le menu de création de ces distributions de probabilités offre plusieurs possibilités, on doit choisir ici l'option ***Probability Profile*** qui permet de définir une variable aléatoire discrète. Les autres distributions seront étudiées au § II-2, page 35.

Cette création s'effectue comme suit.

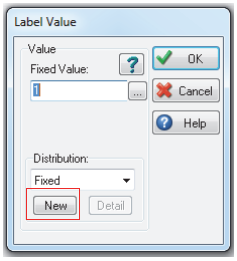


Nous allons créer le *label* «L Possession radio», cette fois-ci en partant du point d'entrée, sachant que la procédure est la même en partant d'une activité et l'affecter au patient entrant en lui attribuant la valeur prise par une occurrence de

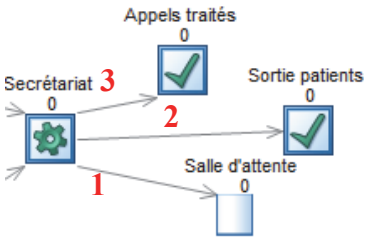
la variable aléatoire «D radio».



Cette création de la distribution «D Radio»est passée par l'onglet Data and Rules mais il est possible de créer directement cette distribution en cochant le bouton radio Set to permettant de définir la valeur d'un label, puis en sélectionnant le bouton New.



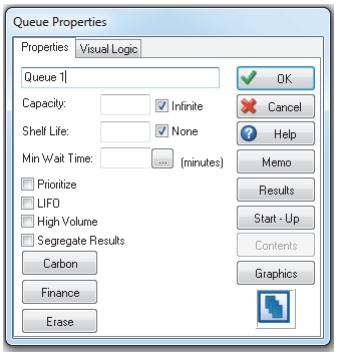
On aura besoin d'un *label* pour définir la direction empruntée par un item à la sortie du secrétariat. On appellera ce *label* «L Sortie Secrétariat» en raison de son usage. On aurait pu aussi appeler ce label «L Etape» car, pour le patient, il correspond à une étape de son processus de traitement dans le système (on reprendra cette convention à la page 99). On affectera à l'item entrant ce *label* en lui assignant la valeur 1. Ultérieurement, on l'affectera également aux appels téléphoniques entrant en lui assignant la valeur 3.



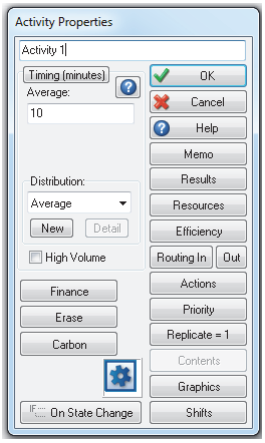
II-1.4 Création des files d'attente, activités, ressources et points de sortie

Il s'agit maintenant de créer le graphe du processus de production étudié. Pour ce faire, il faut partir, dans un premier temps, de la cartographie des flux établie page 18, en fusionnant les flux partant d'une même origine pour se rendre à une même destination. Ce graphe est constitué de files d'attente, d'activités et de points de sortie qui correspondent, avec les points d'entrée, à des points de passage possibles des items dans le système productif étudié. Les chemins que peuvent potentiellement emprunter les items sont représentés par des flèches. On va examiner la création de ces différents points de passage. On examinera en outre la création d'une ressource, même si la modélisation de notre exemple n'en mobilise pas.

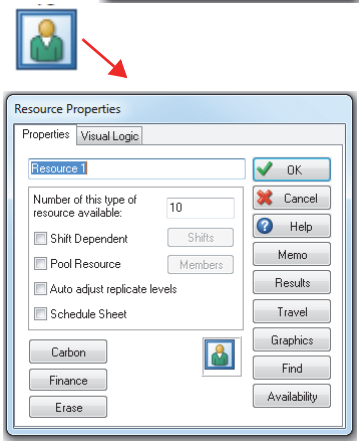
- **Création d'une file d'attente**, un stock pouvant toujours être considéré comme une file d'attente. Une file d'attente est créée avec l'outil Queue ; le pointeur d'écran se transforme en une croix ; une fois la localisation désirée atteinte, un clic-gauche permet de créer la file d'attente et de la visualiser dans le modèle graphique. Un double-clic gauche sur l'icône provoque l'affichage de la fenêtre des propriétés de la file d'attente, toutes initialisées par défaut (voir ci-dessous). On analysera en détail ces propriétés au chapitre IV, page 62.



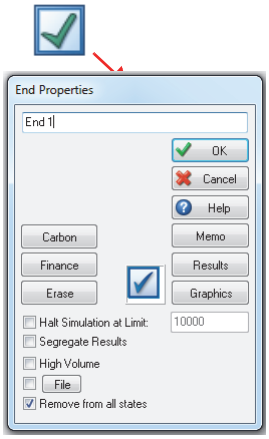
- **Création d'une activité**. L'activité est créée avec l'outil Activity, de la même façon qu'une file d'attente. Un double-clic gauche sur son icône provoque l'affichage de la fenêtre des propriétés de l'activité. Ces propriétés sont initialisées par défaut (voir ci-contre). On analysera en détail ces propriétés au chapitre VI, page 71.



- **Création d'une ressource**. Une ressource est créée avec l'outil Resource, de la même façon qu'une activité. Nous n'avons pas besoin ici d'expliciter les ressources en matériel et personnel mobilisées par une activité pour exécuter une opération. Un double-clic gauche sur son icône provoque l'affichage de la fenêtre des propriétés de la ressource, initialisées par défaut (voir ci-dessous). On analysera en détail ces propriétés au chapitre V, page 68.



- **Création d'un point de sortie**. Le point de sortie est créé avec l'outil End, de la même façon qu'une file d'attente. Un double-clic gauche sur son icône provoque l'affichage de la fenêtre des propriétés d'un point de sortie, initialisées par défaut (voir ci-dessus). On reviendra au § III-2.1, page 51 sur les modalités d'exploitation des informations recueillies sur les items quittant le système productif étudié.



- **Création des flèches**. Une flèche matérialise un itinéraire partiel qu'au moins un item est susceptible d'emprunter. Si cet itinéraire partiel a été créé par erreur, il suffit, pour le détruire, de cliquer sur la flèche puis d'utiliser la touche de suppression de votre ordinateur.

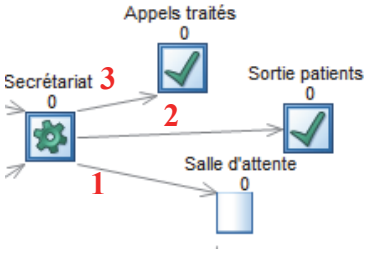
  Pour relier deux points de passage, il faut cliquer sur le bouton Edit Routing Arrows qui se transforme en , pour marquer que la fonctionnalité de création de flèches est active. Le pointeur d'écran se transforme alors en flèche horizontale . Il faut se mettre ensuite sur le point de départ et effectuer un clic-gauche; un point d'interrogation est alors ajouté au-dessus de la flèche du pointeur d'écran ( ); lorsque ce pointeur se trouve au-dessus d'un point de destination possible, l'image du pointeur se transforme et flèche épaisse ( ); un clic-gauche valide alors cette destination. Pour désactiver l'outil de création de flèche, il suffit de cliquer sur le bouton qui reprend sa forme initiale (Edit Routing Arrows).

  Certaines modélisations de processus impliquent qu'une activité donnée et une file d'attente donnée puissent être à la fois point de départ et point d'arrivée de chacun de ces deux points de passage. L'existence d'un tel cycle dans le graphe de modélisation du processus productif ne pose aucun problème.

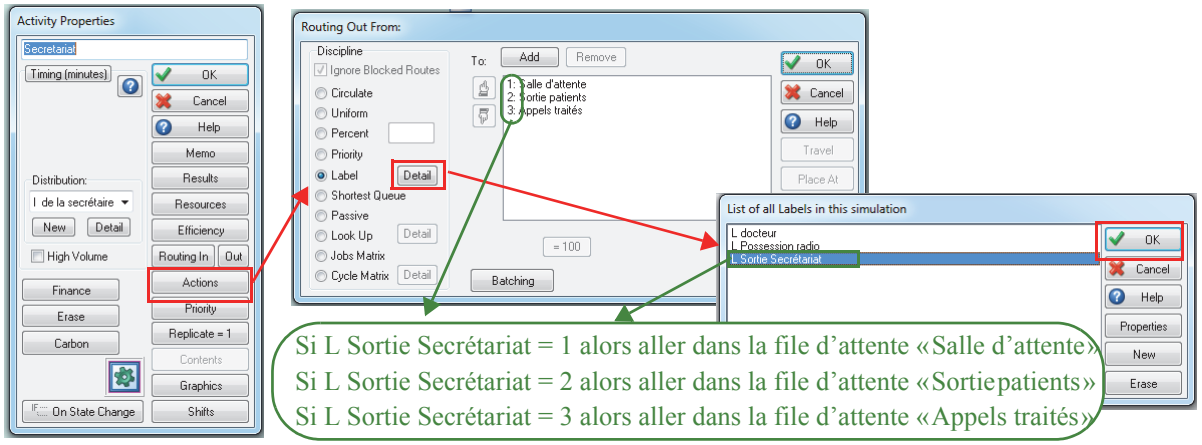
  Ajoutons enfin que la déclaration de l'utilisation d'une ressource par une activité s'effectue dans la définition des propriétés de cette activité. Aucune flèche ne peut relier une ressource à une activité pour matérialiser l'appel de cette ressource, la flèche n'ayant pour vocation que de matérialiser un trajet possible par un item traité par le système productif modélisé.

II-1.5 Contrôle du cheminement d'un item

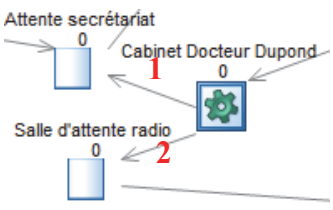
Seules les activités font «bouger» les items, en les «tirant» d'une file d'attente puis, après traitement, en les poussant vers une file d'attente ou un point de sortie. Les files d'attentes sont, par nature, passives. Les problèmes de contrôle du cheminement d'un item se posent essentiellement lorsqu'une file d'attente est alimentée par plusieurs activités ou lorsqu'une activité alimente plusieurs files d'attente.



Commençons par examiner le dernier cas, qui se pose pour les patients en sortie du secrétariat ou des cabinets de consultation mais aussi pour les appels téléphoniques. Trois directions sont possibles à la sortie du secrétariat. Le patient qui arrive doit emprunter la première direction; lorsqu'il repasse au secrétariat, il doit prendre la seconde direction pour sortir du cabinet médical. Un appel téléphonique, après traitement par la secrétaire[1], ne peut qu'emprunter la troisième direction. Le principe retenu est d'utiliser la valeur d'un *label* («L Sortie Secrétariat», en l'occurrence) de l'item par l'activité «Secrétariat» dans la définition de la direction de sortie à suivre. La procédure à suivre est la suivante, le *label* «L Sortie Secrétariat» ayant déjà été créé.

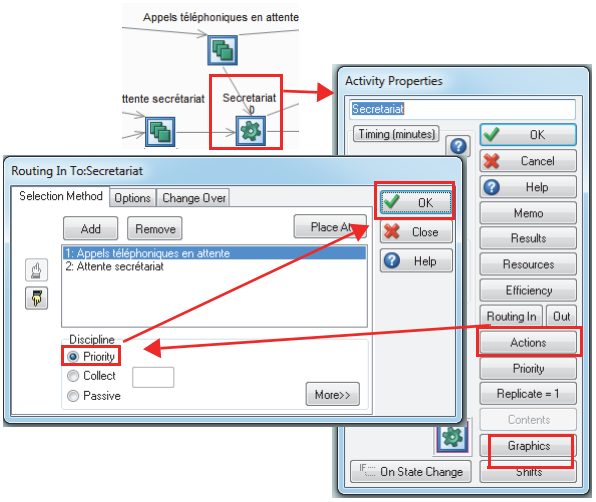


Une fois dans la salle d'attente, le patient est pris en charge par un docteur et retournera ensuite au secrétariat (éventuellement après une prise de radio et une consultation complémentaire). Pour éviter que le patient ne retourne en salle d'attente, il faut modifier la valeur de son *label* «L Sortie Secrétariat» en lui assignant la valeur 2, ce qui doit se faire en passant par le Actions de l'activité «Cabinet Docteur Dupond», puis en appuyant sur le bouton Add a Label to Change pour sélectionner le *label* dans la liste des labels puis en choisissant l'option Set to: en gardant la distribution par défaut Fixed Value: mais en passant à 2 la constante dans la fenêtre de saisie de Fixed Value: 2. Il faut ajouter que le *label* «L Sortie Secrétariat» est affecté à tous les appels téléphoniques entrants, avec la valeur 3, ce qui fait qu'après traitement de l'appel, l'item correspondant ne peut que se rendre au point de sortie «Appels traités».

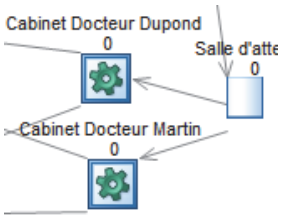


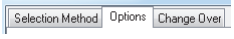
En sortie du cabinet d'un docteur, le patient peut aller au secrétariat ou à la radio. On utilisera, là aussi, l'option Label du *Routing Out* de l'activité en utilisant le *label* «L Possession radio», dont la valeur a été déterminée au point d'entrée. Si cette valeur est 2, il convient, de passer à 1 cette valeur de *label* dans l'activité «Cabinet de radiologie» pour éviter tout bouclage (Actions puis adaptation de la démarche décrite au paragraphe précédent).

1. On expliquera à la page 64, comment éliminer les appels téléphoniques arrivés en file d'attente depuis plus de 2 minutes.

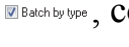


Examinons maintenant les décisions à prendre en entrée des activités, en commençant par le secrétariat. Il semble naturel de mettre en attente un appel lorsque la secrétaire est occupée avec un patient mais une fois ce traitement effectué, l'appel téléphonique en attente est prioritaire. La règle de sélection par défaut *Priority* est à garder, en veillant bien à ce que la première file d'attente de la liste soit celle des appels en attente. Le principe de prélèvement est le suivant: s'il existe un item pouvant être prélevé dans la première file d'attente («Appels téléphoniques en attente»), l'activité le prélève, dans le cas contraire, il passe à la file d'attente suivante («Attente secrétariat»); ce processus se poursuit s'il reste d'autres files d'attente, ce qui n'est pas le cas ici. Ajoutons que si les files d'attentes sont vides lorsque l'activité se libère, la recherche reprend dès qu'un item arrive dans l'une de ces files d'attente. On examinera à la page 72 les autres méthodes de sélection d'un item proposées dans cette fenêtre.



Examinons maintenant la décision de sélection d'un patient en salle d'attente, sachant que le Docteur Dupond et le Docteur Martin vont chercher leurs patients en salle d'attente[1]. Sauf spécification contraire, que nous allons mettre en place, tout docteur venant chercher un patient prend en consultation le premier arrivé dans cette salle d'attente (règle implicite du Premier-Entré - Premier-Sorti; on reviendra sur ce point à la page 64), qu'il ait rendez-vous ou non avec lui, tous les items d'une file d'attente étant considérés comme interchangeables (file d'attente qualifiée d'**homogène**). Il faut donc contraindre l'activité qui se libère («Cabinet docteur Martin» ou «Cabinet docteur Dupond») à ne pouvoir prélever qu'un item pertinent. Aux points d'entrée, les patients ayant rendez-vous avec le docteur Dupond ont tous la valeur 1 pour le *label* «L docteur» et que ceux qui ont rendez-vous avec le docteur Martin ont tous la valeur 2 pour ce label. On va utiliser cette information pour mettre une condition au prélèvement d'un item, ce qui se fait dans la fenêtre de l'onglet 

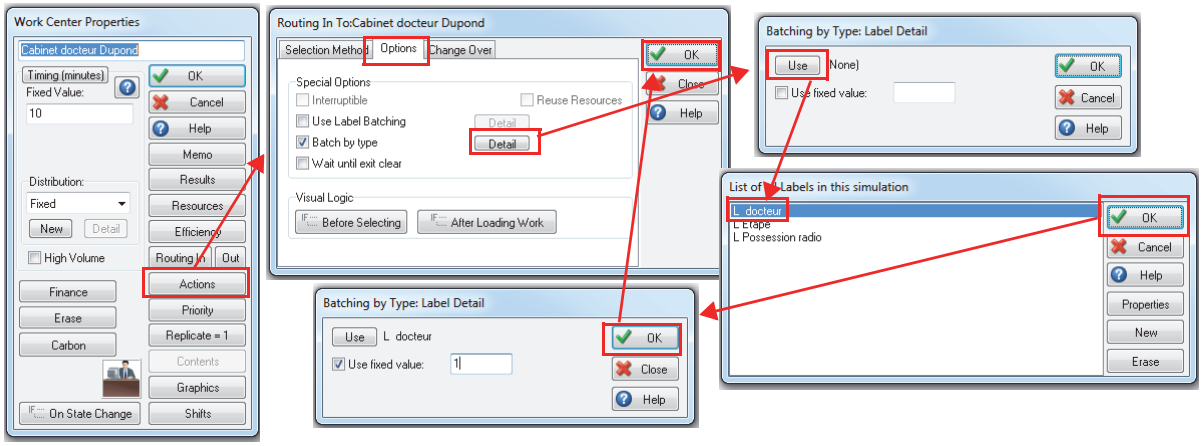
ouverte par le bouton 

, en utilisant l'option 

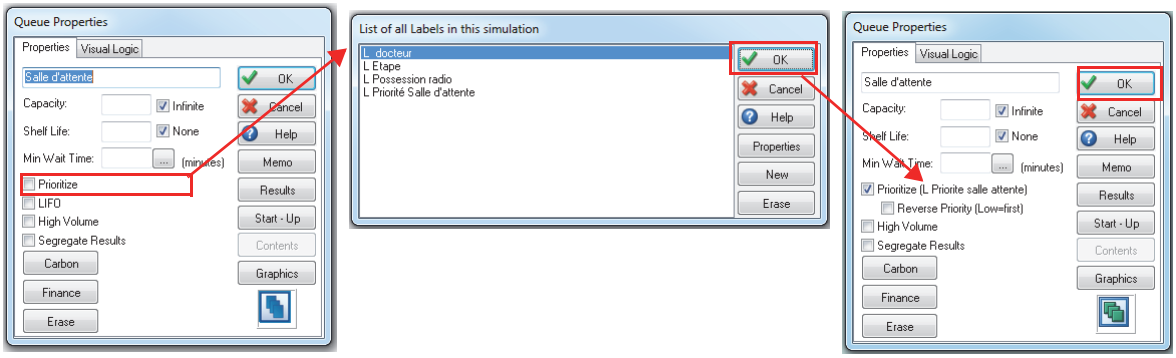
, comme détaillé ci-après.

Un second problème doit être pris en compte dans la modélisation, pour que celle-ci reflète bien la réalité: un patient, qui se fait faire une radio prescrite par un docteur, retourne en salle d'attente après obtention de la radio; il est alors prioritaire par rapport aux autres patients qui viennent d'arriver pour une consultation avec le même praticien. Sans spécification contraire, les items d'une file d'attente

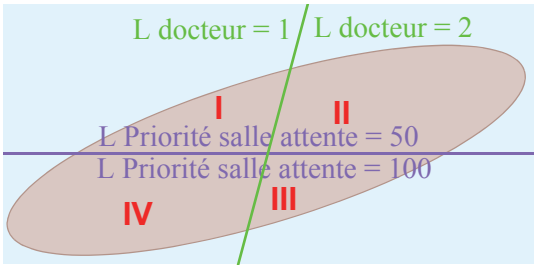
1. Pour utiliser le langage de la modélisation sous Simul8, ces activités (Cabinet Docteur Dupond» et «Cabinet Docteur Martin») cherchent à tirer un item de la file d'attente «Salle d'attente» dès qu'elles se trouvent sans travail.



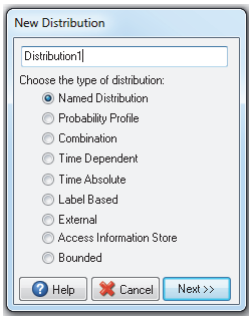
ont tous la même priorité, ce qui implique que la règle Premier Arrivé - Premier Sorti s'applique globalement (ou sur le sous-ensemble des items respectant la condition définie par le ☑ Batch by type). Il convient d'introduire une priorité qui sera définie dans un *label* «L Priorité salle attente» affecté aux items entrant avec une valeur par défaut égale à 50. Cette valeur arbitrairement choisie, correspond à une priorité moyenne, sachant que l'usage est de définir les niveaux de priorité entre 0 et 100. On passe cette valeur de *label* à 100 lorsque l'item est traité par l'activité «Cabinet radiologie» (bouton Actions de l'activité, puis bouton Add a Label to Change pour sélectionner le *label* dans la liste des labels puis option ◉ Set to: en gardant le distribution par défaut Distribution: Fixed mais en passant à 100). L'utilisation de ce *label* dans la file d'attente «Salle d'attente» pour modifier la liste des items en sortie passe par l'option *Prioritize* et s'effectue comme illustré en haut de la page suivante.



Par ces spécifications dans la modélisation on a défini une partition des items présents dans la file d'attente «Salle d'attente» selon deux critères, ce qu'illustre la figure ci-contre. Quand l'activité «Cabinet docteur Martin» se libère (fin d'une consultation) et cherche à prélever un item dans la file d'attente «salle d'attente» (plus concrètement, docteur allant chercher un client dans la salle d'attente), en prenant en compte qu'il ne s'intéresse qu'à ses clients (items avec un *label* «docteur = 2») seule la réunion des ensembles **II** et **III** est concernée; si l'ensemble **III** n'est pas vide, le premier item arrivé dans cet ensemble **III** est prélevé; dans le cas contraire et si l'ensemble **II** n'est pas vide, le premier item arrivé dans l'ensemble **II** est prélevé.



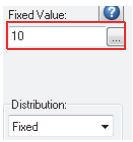
II-2 Définition des lois de durée d'une activité et des lois d'arrivée des items



On a déjà créé des distributions de probabilités de type Named Distribution (voir page 27) et de type Probability Profile (voir page 28). On s'appuiera sur le cas du cabinet médical pour présenter la façon de définir d'autres types de distribution de probabilités qui sont utilisées pour:

- définir l'intervalle de temps séparant deux arrivées successives à un point d'entrée ;
- définir la durée du temps de traitement d'une activité ;
- définir la valeur prise par une variable aléatoire utilisée par un *label* (ce que nous avons illustré avec la distribution «D radio» utilisée pour définir une valeur du *label* «L Possession radio»);
- modifier la valeur – 1 par défaut – du nombre d'items arrivant à un instant donné dans un point d'entrée (bouton Batching de la fenêtre de propriété d'un point d'entrée, voir page 55);
- modifier la valeur – 1 par défaut – du nombre d'items sortant d'une activité (bouton Batching de la fenêtre ouverte par le bouton Out de la fenêtre de propriété de l'activité, voir page 89);
- définir des intervalles de temps entre deux pannes, des durées de réparation, ce qui correspond à des caractéristiques d'une activité voir page 94.

II-2.1 Distribution de type *Fixed*



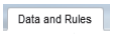
Cette distribution permet d'utiliser une constante pour définir un temps opératoire ou l'intervalle entre deux arrivées dans le système. En se positionnant sur la fenêtre de saisie de la valeur numérique, on fait apparaître un bouton qui permet de choisir l'information pertinente (voir exemple à la page 38).

II-2.2 Distribution de type *Named Distribution*

Le choix d'une distribution de type *Named Distribution* conduit à créer sous un nom unique une distribution de probabilité et les paramètres qui la caractérise. Cette nouvelle distribution est alors ajoutée à la liste des distributions de probabilités disponibles dans la modélisation et peut donc être utilisée à plusieurs endroits. Ceci présente beaucoup d'avantages, notamment en termes de maintenance du modèle, la modification d'une ou plusieurs caractéristiques de cette distribution n'étant à effectuer qu'une seule fois pour être effective partout où cette distribution est employée.

Complétons notre modélisation en définissant tout d'abord les temps opératoires des activités. D'après l'énoncé, le temps de travail d'un médecin dépend du fait que le patient arrive avec une radio ou sans radio. Il convient donc de commencer par créer ces distributions de probabilités. Il existe quatre façons de créer d'une distribution de probabilité.

- La première a déjà été présentée à la page 28. Il suffit de cliquer sur l'icône 

de l'onglet 

pour ouvrir une fenêtre de création d'une nouvelle distribution dans laquelle la distribution Normale $\mathcal{N}(10; 2{,}5)$ est proposée par défaut, ainsi qu'un nom. Ce nom par défaut se termine par un nombre correspond au numéro d'ordre de création de la distribution. Le type de distribution et les paramètres associés peuvent être alors changés.
- La seconde possibilité part de l'utilisation du menu ouvert en cliquant sur le triangle situé en bas à droite de l'icône 

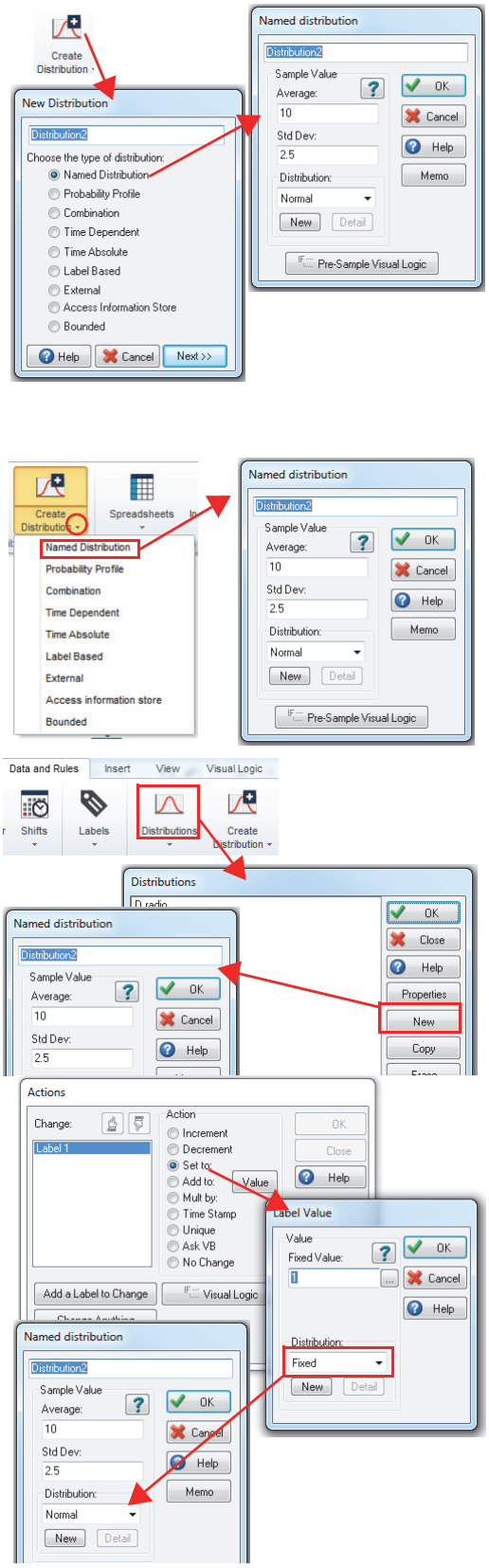
de l'onglet 

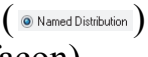
, ce qui ouvre la même fenêtre de spécifications de la nouvelle distribution que précédemment.
- La troisième possibilité part de la fenêtre ouverte en cliquant sur l'icône 

qui ouvre alors une fenêtre listant les distributions existantes. Il suffit alors de cliquer sur le bouton 

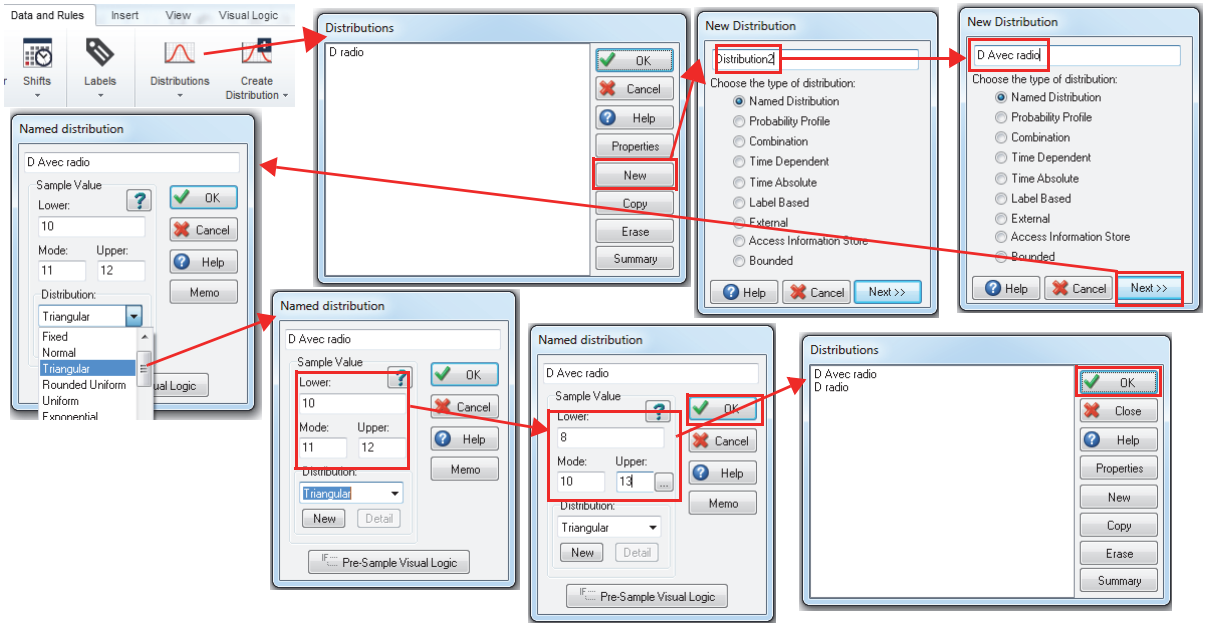
pour créer une nouvelle distribution. Cette possibilité est illustrée ci-contre.
- La dernière possibilité consiste à créer au tout dernier moment la distribution dont on a besoin, c'est-à-dire au moment de la définition de la valeur que doit prendre le label (

) qui propose par défaut la distribution Fixed qui correspond à une constante, distribution qu'il suffit alors de modifier.



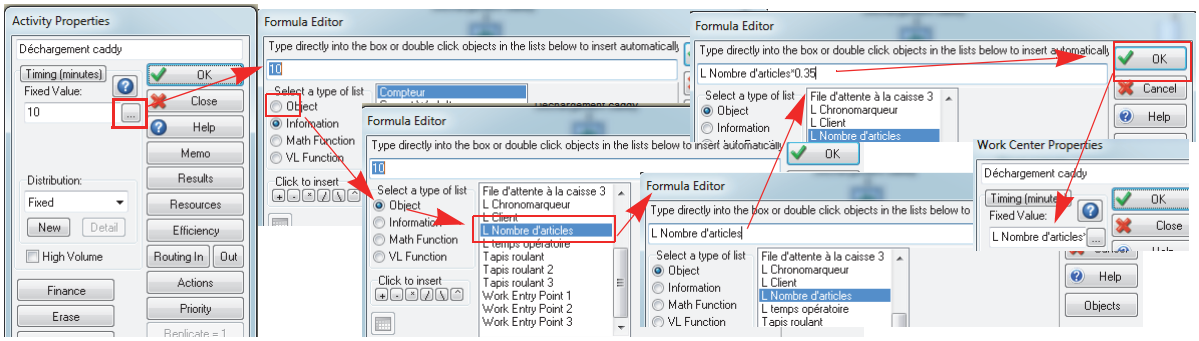
La création de la distribution «D Avec radio» du temps de consultation d'un patient avec radio s'effectue de la façon suivante, qui est la seconde création d'une distribution de type ***Named Distribution*** (

). La création de la distribution «D Sans radio» s'effectue de la même façon).

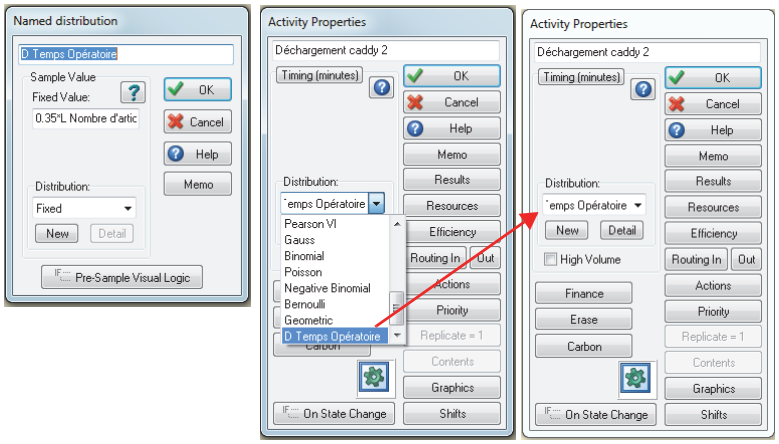


Ajoutons, deux remarques pour terminer. Ces remarques peuvent être sautées en première lecture.

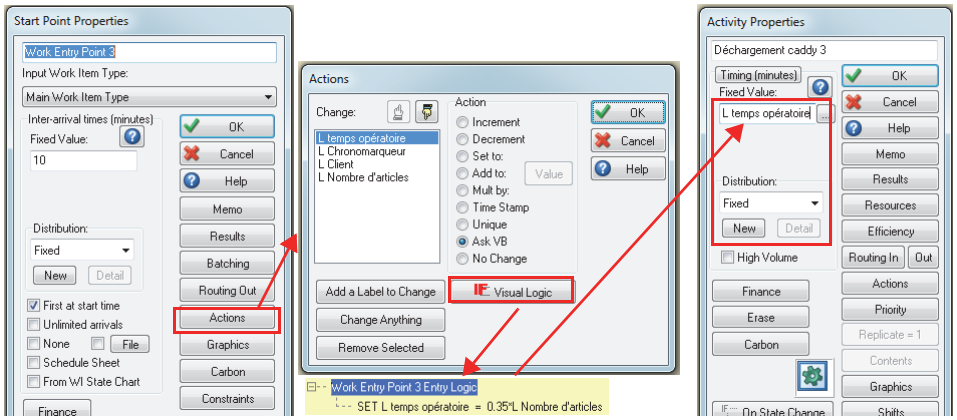
- Un **paramètre d'une distribution** de type *Named Distribution* peut être **calculé** à partir de valeur(s) de *label(s)* possédé(s) par l'item, valeur pouvant être l'occurrence d'une variable aléatoire. Trois moyens d'implémenter ce principe sont disponibles et tous les trois illustrés par un même exemple dans lequel le temps de déchargement d'un caddie à une caisse de supermarché peut être considéré, en première approche, comme proportionnel aux nombres «L Nombre d'articles» d'articles à décharger, occurrence d'une variable aléatoire discrète Uniforme (Parametrage_TO_fonction_label_quanti.S8).
  - On peut tout d'abord définir la valeur du paramètre de la distribution (constante dans notre exemple) par une formule faisant intervenir le *label* directement dans la fenêtre de l'activité.



- On peut également utiliser une distribution de probabilité, préalablement créée, et l'utiliser comme distribution de référence dans la fenêtre de détermination du temps de traitement de l'activité. Cette solution alternative (tout comme la suivante) est préférable, pour des raisons de maintenance du modèle, si cette distribution est utilisée à plusieurs endroits (dans notre exemple, existence de plusieurs caisses dans le supermarché).



- Enfin, on peut remplacer la formule de la première solution par un *label* de l'item, défini au point d'entrée et dont la valeur est le résultat de cette formule de calcul.

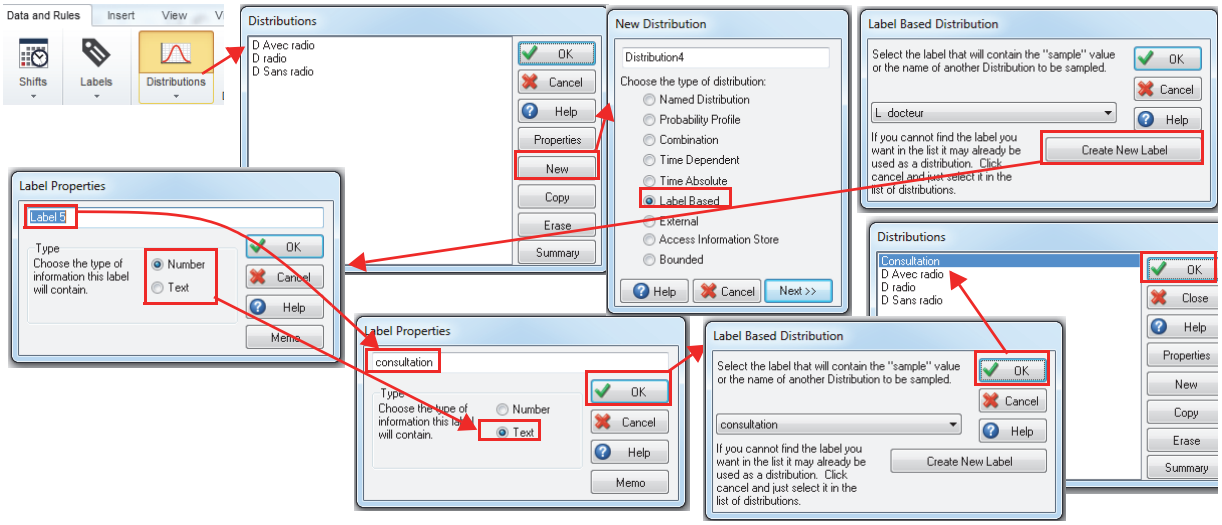


- Les **paramètres d'une distribution** de type *Named Distribution* peuvent être **lus** dans un tableau stocké dans l'***Information Store*** à des adresses calculées à partir des valeurs d'un ou plusieurs *labels*, pour remplacer une distribution de type *Time dependant* (§ II-2.5, page 43) ou de type *Label Based* (voir § II-2.3); on reviendra sur les fondements de la programmation en *Visual Logic* à la page 101 et sur les principes d'utilisation de l'*Information Store* à la page 98.
  - L'exemple Distribution_Time_Dependant_Jour_semaine_IS.S8, sur lequel on reviendra à la page 44, illustre cet usage de données stockées dans l'*Information Store* pour remplacer par une unique distribution de type *Named Distribution*, une distribution de type *Time dependant* mobilisant un ensemble de distributions de type *Named Distribution* correspondant à une même loi statistique.
  - L'exemple DOCTEUR_Information_Store_0.S8 illustre cet usage de données stockées dans l'*Information Store* pour remplacer par une unique distribution de type *Named Distribution*, l'utilisation d'une distribution de type *Label Based* (voir ci-après) mobilisant plusieurs distributions de type *Named Distribution* correspondant à la même loi statistique.

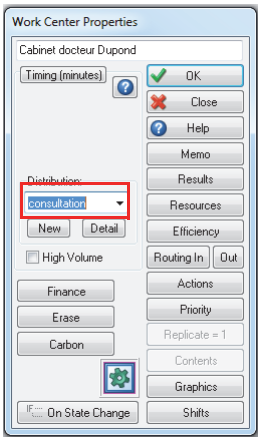
II-2.3 Distribution de type *Label Based*

L'utilisation de ces deux distributions par les activités «Cabinet Dupond» et «Cabinet Martin» va se faire par l'intermédiaire d'une troisième distribution, de type **Label Based**, en choisissant l'option ◉ Label Based. Le nom choisi pour cette distribution est «Consultation». Elle est facilement utilisable par ces deux activités car, dès sa création, cette distribution «Consultation» est ajoutée à la liste des distributions de probabilités accessibles par le menu déroulant des distributions disponibles. Le nom «Consultation» est en fait à la fois le nom d'une distribution et celui d'un *label* que l'on affecte à un item qui «embarque» avec lui l'information nécessaire à l'activité (ce statut hybride expliquant pourquoi on n'a pas retenu de faire précéder «Consultation» de la lettre D ou de la lettre L). Ce *label* est nécessairement de type littéral et non de type numérique parce qu'il doit contenir le nom de la distribution à utiliser.

La création de cette distribution «Consultation» s'effectue comme illustré ci-dessous.



Il faut ensuite changer la distribution du temps de traitement d'un item dans l'activité «Cabinet Docteur Dupond», ce qui se fait dans la fenêtre de propriété de cette activité, en choisissant cette distribution dans le menu déroulant des distributions. (voir ci-contre). L'affectation du *label* «Consultation» aux items entrant s'effectue comme montré à la page suivante. Par défaut, ce *label* prend la valeur «D Avec radio», ce qui n'est pertinent que si le *label* «L Possession radio» vaut 1. Si tel n'est pas le cas, il faut modifier la valeur prise par le *label* «Consultation», ce qui conduit à utiliser les possibilités de programmation offertes par *Visual Logic*.

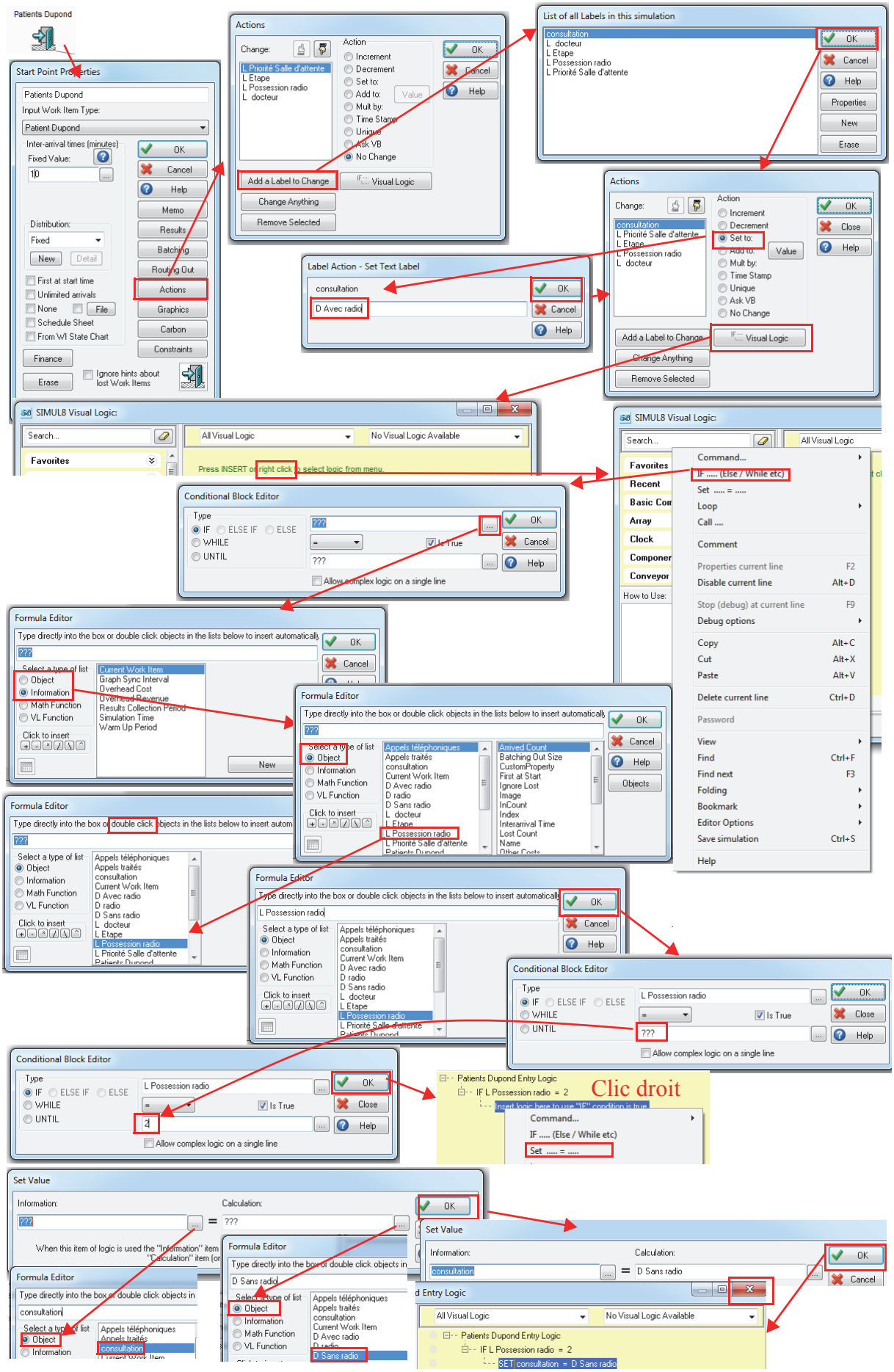


Examinons donc comment changer la distribution associée à «Consultation» lorsque le patient arrive sans radio. D'un point de vue logique, on peut écrire :

Si «L Possession radio» = 1 alors

Assigner au label «Consultation» la valeur «D Sans radio»

L'utilisation de *Visual Logic* est facile car l'utilisateur crée des instructions sans avoir à les écrire. La création d'un programme passe par un menu (voir page suivante) qui


Patients Dupond
Start Point Properties
Actions
List of all Labels in this simulation
Label Action - Set Text Label
SIMUL8 Visual Logic:
Conditional Block Editor
Formula Editor
Clic droit
Set Value
Patients Dupond Entry Logic
IF L Possession radio = 2
SET consultation = D Sans radio

liste les instructions de base utilisables et permet de rédiger en «mode assisté» les instructions à créer. Ici, on a besoin de l'instruction IF ..... (Else / While etc) pour exécuter le test et de l'instruction Set ..... = ..., pour exécuter l'assignation. On reviendra sur les fondements de la programmation en *Visual Logic* au chapitre VII, page 98. L'exemple Distribution_Label_Based.S8 reprend le «sous-problème» que l'on vient de traiter.

Il ne reste plus qu'à utiliser la même démarche pour définir le travail de la secrétaire qui effectue trois types de tâches: accueil d'un patient, réponse téléphonique, formalités de sortie d'un patient, ce qui conduit à créer les distributions de type *Named Distribution*: «D Secrétariat accueil», «D Secrétariat paiement» et «D Secrétariat téléphone». Il faut créer ensuite la distribution de type *Label Based*, «Travail de la secrétaire».

Deux rappels doivent être faits.

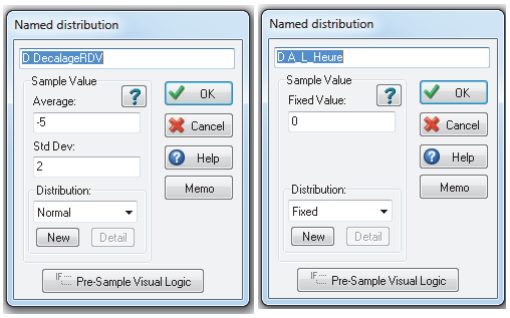
- Tout d'abord, comme on l'a signalé avec les deux premières remarques formulées à la page 37, une distribution de type *Label Based* faisant appel toujours à la même distribution de type *Named Distribution* (triangulaire, par exemple) peut être remplacée par une distribution unique de type *Named Distribution*, dans laquelle les paramètres sont des labels de l'item qui utilisera cette distribution (voir Parametrage_TO_fonction_label_quanti.S8).
- On a également évoqué, à la page 38, la possibilité de remplacer une distribution de type *Label Based* mobilisant plusieurs distributions de type *Named Distribution* de la même loi statistique par une distribution unique de type *Named Distribution*, correspondant à cette loi, les paramètres de cette loi étant lus dans un tableau de l'*Information Store* à des adresses calculées à partir des valeurs d'un ou plusieurs labels de l'item qui utilise cette distribution (voir exemple DOCTEUR_Information_Store_0.S8)

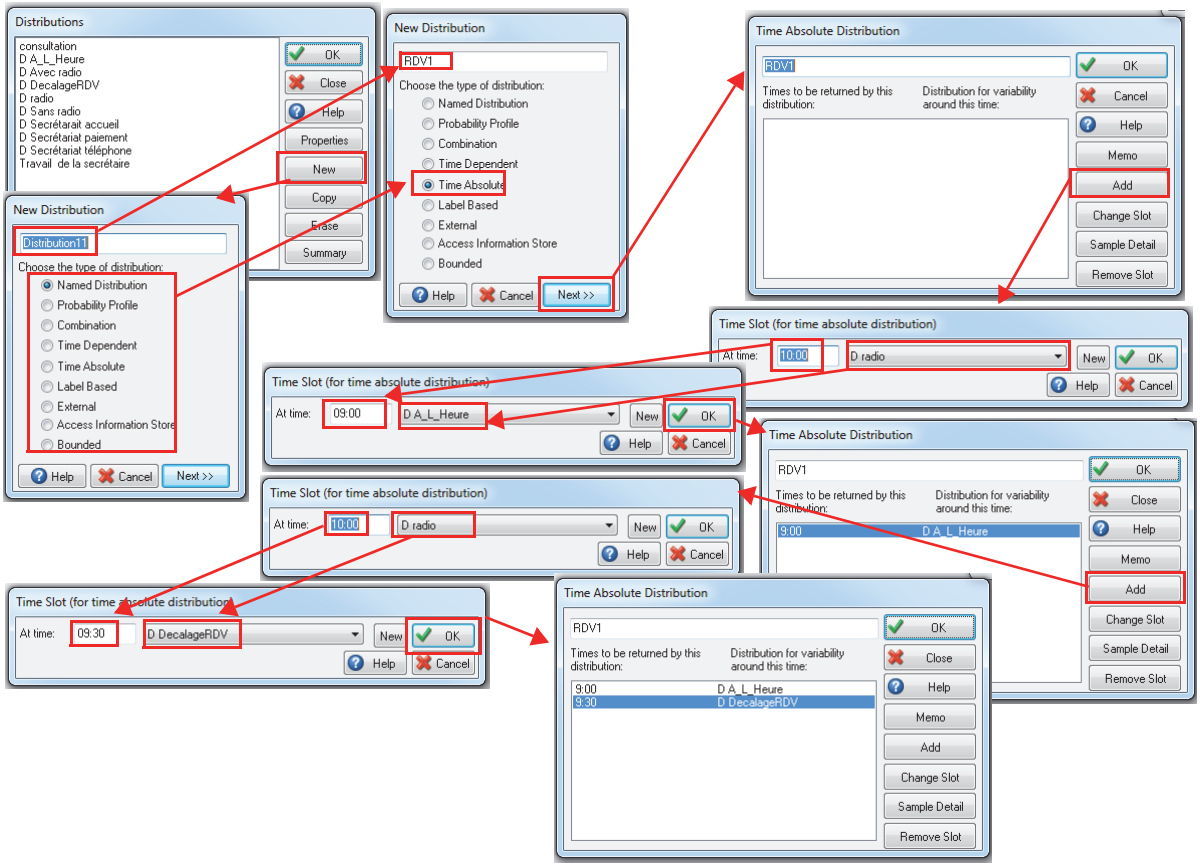
II-2.4 Distribution de type *Time Absolute*

Il faut maintenant définir les lois des arrivées à un point d'entrée. Ces lois ont en commun qu'elles caractérisent la distribution de probabilités de l'intervalle de temps séparant deux arrivées successives d'items au point d'entrée considéré. Deux cas de figure doivent être distingués.

Le premier correspond à des arrivées aléatoires indépendantes, aucune relation n'étant décelable entre deux arrivées successives. Très souvent la loi de cet intervalle entre deux arrivées successives est une loi exponentielle, ce qui correspond à un processus de Poisson (le produit des paramètres de ces deux lois étant égal à 1). Dans de nombreux processus de production de services (péages d'autoroute, centres d'appels, services d'urgence…), le paramètre de la loi n'est pas stable au cours de la journée mais suit une évolution telle que l'on peut, par un découpage temporel approprié, considérer que le paramètre est stable sur chacune des plages de temps retenues, la valeur de ce paramètre variant d'une plage à l'autre. Ce type de loi d'arrivée est facilement modélisable sous Simul8 en faisant appel à une distribution de type ***Time dependant***, à condition d'avoir préalablement défini les lois d'arrivées de chacune des plages de temps considérées.

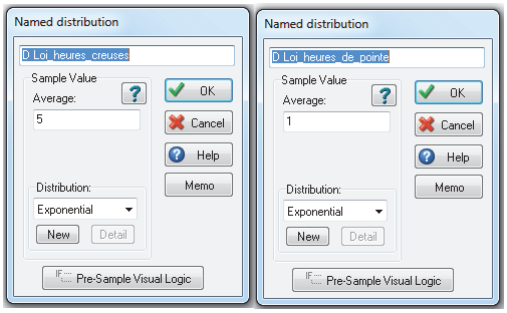
Le second cas est illustré par l'exemple du cabinet médical où les arrivées se font sur rendez-vous. Bien évidemment, les patients ne se présent jamais à l'heure exacte du rendez-vous; ils sont en moyenne en avance, ce qui n'empêche pas certains d'arriver en retard. Dans l'énoncé, on suppose que le décalage par rapport à l'heure du rendez-vous suit la loi $\mathcal{N}(-5,3)$, sauf le premier rendez-vous que l'on considère ici comme arrivant à l'heure[1]. On commence donc par créer deux distributions correspondant à ces écarts (voir page précédente), avant de remplir le carnet de rendez-vous à l'aide de la distribution ***Time Absolute*** (Time Absolute), comme suit (on s'est arrêté ici au 2e rendez-vous).





1. Le système productif est censé ouvrir à 9 heures; toute arrivée d'items s'effectuant en dehors de la plage d'ouverture du système productif est «perdue», puisque l'item entrant ne dispose pas de «lieu» où se rendre. On peut, dans une modélisation un peu plus complexe, traiter ce cas de figure sans difficulté mais l'intérêt en est limité pour cet exemple introductif.

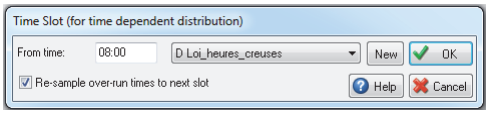
II-2.5 Distribution de type *Time Dependant*



Dans l'exemple du cabinet médical, ce type de distribution n'est pas utilisé. Nous allons cependant l'introduire maintenant car vous en aurez besoin pour les premiers cas à traiter. Le contexte d'utilisation de ce type de distribution a été décrit à la page 41. Le plus simple pour présenter ce type de distribution est de s'appuyer sur l'exemple suivant. Supposons que les arrivées au GAB (Guichet Automatique Bancaire) du hall d'entrée de l'université d'Alphaville connaît une fréquentation en heures creuses, c'est-à-dire entre 8 heures et 11h30 et entre 14 heures et 22 heures, caractérisée par une distribution exponentielle de paramètre 4 minutes (arrivées Poisson $\mathcal{P}(0{,}25)$ par minute, ce qui correspond à un intervalle moyen de 4 minutes entre deux arrivées). En heures de pointe, entre 11h30 et 14 heures, la loi des arrivées est une loi exponentielle de paramètre 1. Dans la réalité, la modulation des arrivées est plus importante et varie d'un jour à l'autre mais, pour illustrer la démarche, disposer de trois plages est suffisant. Une fois créé les distributions de type *Named Distribution* à utiliser, on passe à la création de la distribution de type *Time Dépendent* comme illustré ci-dessous (voir exemple Distribution_-Time_Dependant.S8.)

Plusieurs compléments doivent être apportés pour utiliser correctement les distributions de type *Time dependant*.

- Rappelons que, dans une simulation à événements discrets, toute arrivée à un point d'entrée (par exemple à 11h28) déclenche immédiatement le calcul de la date de l'arrivée suivante. Si la date de cette prochaine arrivée reste dans la même plage de temps, pas de problème. Si elle se situe dans la plage suivante (par exemple 11h31, en supposant que la plage courante commence à 8 heures et s'achève à 11h30), on peut soit décider de conserver cet événement, soit de le remplacer par une arrivée calculée comme la somme de la date de début de la plage suivante (11h30) et d'un intervalle entre deux arrivées successives générées sur la base de la distribution de probabilités associée à cette nouvelle plage.

  Cette seconde solution est retenue par Simul8 si l'option Re-sample over-run times to next slot est cochée dans la plage de temps précédente (ici celle allant de 8 heures à 11h30). Cette remarque a trois conséquences.
  - Si les caractéristiques des arrivées ne diffèrent pas trop d'une plage A à la suivante B, cocher cette option pour la plage A ne présente pas beaucoup d'intérêt.
  - Si la plage considérée est la dernière plage et que la prochaine arrivée calculée intervient après la fermeture du système productif (en supposant que le système ne fonctionne pas 24 heures sur 24), cette arrivée est supprimée et la suivante est calculée à partir du début de l'ouverture du système le lendemain et d'une occurrence de la loi des arrivées de la première plage.

- Si aucune précaution n'est prise, des items peuvent rester dans le système productif à la fermeture, ce qui n'est pas envisageable dans les systèmes de production de services à la personne. Dans notre exemple, on ne peut imaginer que l'université ferme avec des étudiants faisant encore la queue devant le GAB. Pour éviter l'arrivée d'items à partir d'une certaine heure (par exemple 21h45), il suffit de créer une nouvelle plage de temps commençant à partir de cette heure et d'associer à cette dernière plage de temps une distribution *Fixed* de paramètre supérieur à l'amplitude de cette dernière période (par exemple 20 minutes, si l'université ferme à 22 heures), sans oublier de cocher l'option

de la période précédente. L'amplitude de cette dernière plage doit être suffisante pour que le système productif puisse traiter tous les items présents au tout début de cette dernière plage. L'exemple Distribution_Time_Dependant_0.S8 illustre l'utilisation de ce principe.

Plusieurs remarques complémentaires sur l'usage de distributions de type *Time dependant* doivent être faites.

- Un certain nombre de systèmes productifs de production de services fonctionnent 24 heures par jour avec des lois d'arrivée dont les paramètres varient au cours de la journée. Il convient dans ce cas, pour éviter d'avoir une simulation erronée, d'avoir la première période commençant à 0 heure, ce qu'illustre l'exemple Distribution_Time_Dependant_24heures.S8.
- Lorsque les arrivées dans le système sont de type *Time dependant*, l'analyse des indicateurs obtenus au point de sortie n'a pas grande signification et n'est pas pertinente pour la recherche d'amélioration de la performance du système. Il est conseillé d'utiliser une segmentation des résultats sur la base du numéro de plage d'arrivée. Voir les deux solutions de l'exemple détaillé Distribution_Time_Dependant_partition.S8 et les remarques de la page 103.
- Dans de nombreux systèmes de production de services, les caractéristiques des lois d'arrivée ne varient pas seulement au cours de la journée, elles varient souvent en fonction du jour de la semaine et de la plage horaire; l'exemple Distribution_Time_Dependant_Jour_semaine.S8 présente une solution utilisable dans des cas simples; une solution plus performante basée sur l'exploitation d'un tableau de paramètres stockés dans l'*Information Store* est proposée avec l'exemple Distribution_Time_Dependant_Jour_semaine_IS.S8, on peut noter qu'alors que cette paramétrisation conduit à utiliser une distribution de type *Named Distribution* dont le paramètre varie au cours du temps.

II-2.6 Autres distributions disponibles

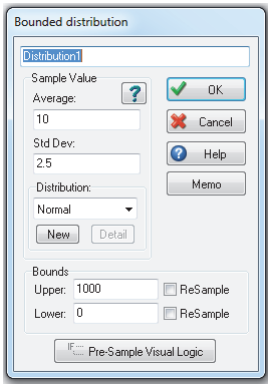
L'option ***External*** permet de lire sur un fichier externe (fichier Excel, par exemple) des nombres correspondant à des intervalles de temps entre deux arrivées successives ou à des temps opératoires. Cette option permet de travailler avec des jeux de données réels.

L'option ***Combination*** permet de créer des réalisations d'une somme de variables aléatoires définies par des distributions de probabilités préalablement créées (exemple, un temps de traitement égal à la somme de deux temps d'opéra-

tion successivement réalisés dans cette activité, ces temps étant tous deux aléatoires et s'enchaînant sans interruption).

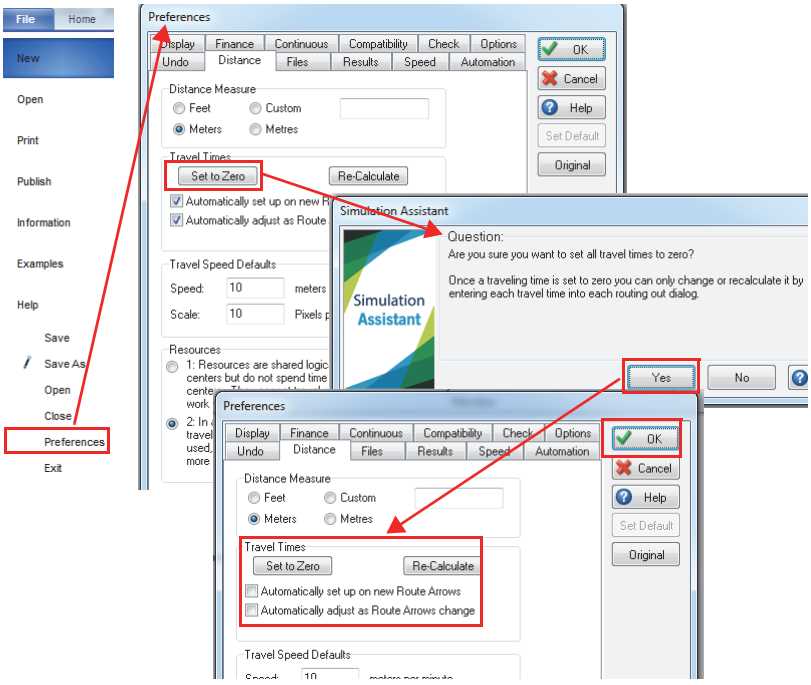
L'option ***Acces Information Store*** revient à lire une durée certaine dans un tableau d'information Store. Cette option est assez peu intéressante : les deux derniers exemples du paragraphe précédent montrant des utilisations plus intéressantes d'une lecture de données dans l'*Information Store*.

L'option ***Bounded distribution*** est équivalente à l'option *Named distribution*, avec une adjonction de bornes supérieures et inférieures à ne pas dépasser dans la génération d'une occurrence de la variable aléatoire par la méthode de Monte Carlo. Si l'option ReSample de la borne supérieure et que la valeur générée excède cette borne, une nouvelle valeur est générée ; si cette option n'est pas cochée, c'est la borne supérieure qui est retenue. La transposition de ce qui vient d'être dit à la borne inférieure est immédiate.

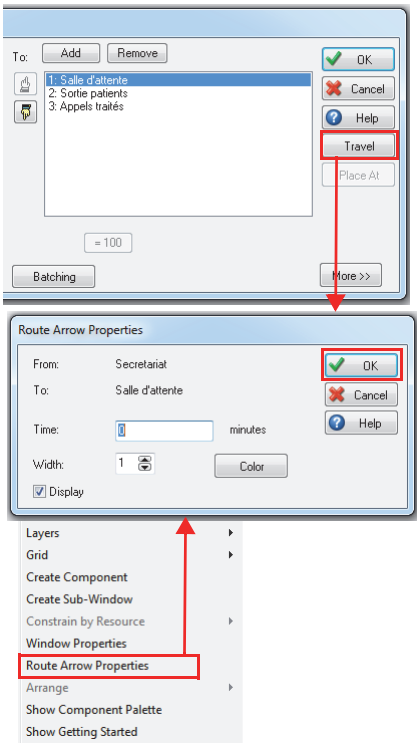


II-2.7 Temps de déplacements entre files d'attente et activités

Le temps de **transport** d'un item entre points d'entrée, de sortie, files d'attente et activités (ou d'une ressource vers une activité) est, par défaut, proportionnel à la distance entre les icônes les représentant sur la cartographie. La vitesse de déplacement par défaut est 10 m/min, vitesse modifiable dans la fenêtre de saisie Speed: 10 de l'onglet Distance du menu Preferences Préférence de l'onglet File de la barre supérieur du menu général. On peut également forcer à 0 les temps de déplacement des items avec le bouton Set to Zero. Cette neutralisation des temps de transport est nécessaire dans de très nombreuses modélisations/simulations de processus de services.



Dans certains processus logistiques, le temps de déplacement peut être important par rapport au temps de traitement ; le temps de transport se définit dans la fenêtre *Routing Out* en sélectionnant la file d'attente de destination, ce qui active le bouton Travel et permet la saisie du temps de transport (ce qu'illustre la fenêtre supérieure ci-contre, en supposant que veuille tenir compte d'un temps de transport du secrétariat à la salle d'attente, ce qui n'est évidemment pas pertinent).



Cette saisie du temps de transport peut aussi s'effectuer en sélectionnant une flèche par un click-gauche puis en ouvrant la fenêtre ci-contre par un click-droit et en choisissant l'option Route Arrow Properties et saisir la durée.

La distance entre une file d'attente et l'activité qu'elle approvisionne est toujours réputée nulle (voir Temps_de_transport.S8). Le cas du temps de transport des convoyeurs est traité de manière spécifique (voir page 95).

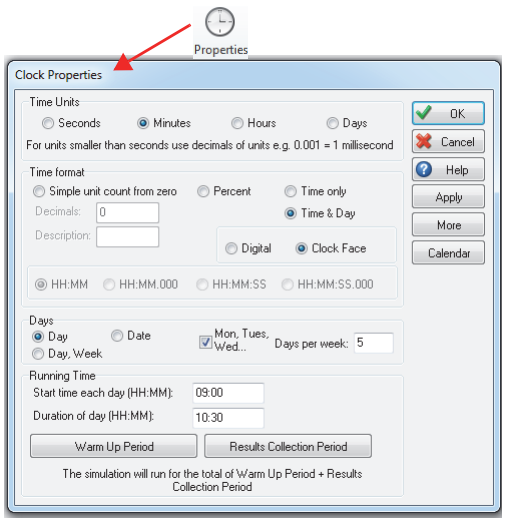
III Exécution de la simulation et analyse des résultats

III-1 Exécution de la simulation

Le modèle de simulation qui vient d'être créé (DOCTEUR.S8) est utilisable tel quel. La simulation est lancée/arrêtée par le bouton ▶ de l'onglet Home et la vitesse de simulation est contrôlée par le curseur Speed: . L'exécution en «pas à pas» (arrêt commandé par le prochain événement survenant dans le système productif) est commandée par le bouton Step. La réinitialisation de la simulation est commandée par le bouton Reset. Pour que les résultats soient exploitables, il convient d'abord de modifier certains paramètres définis par défaut, pour les adapter au problème posé.

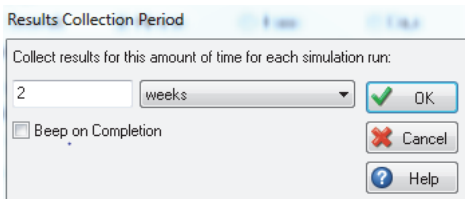
III-1.1 Temps d'ouverture du système productif

Le temps d'ouverture du système productif est défini dans la fenêtre ouverte par l'option Properties du menu de l'onglet Data Rules de la barre de menu; ici on considérera que le cabinet fonctionne 5 jours par semaine de 9 heures (Start time each day (HH:MM): 09:00) à 19 heures 30 sans interruption (soit 10h30 par jour Duration of day (HH:MM): 10:30) même si, dans la réalité, il y a une pause déjeuner d'environ 1 heure 30, dont l'absence de prise en compte ici n'est pas très gênante; on verra à la page 53 comment introduire explicitement ces pauses.

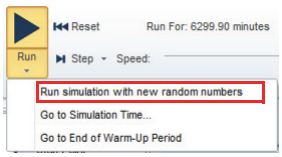


III-1.2 Durée de la simulation

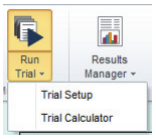
La **durée de la simulation** est définie par le bouton Results Collection Period de la fenêtre précédente ou directement par l'option Run For: de l'onglet Home de la barre de menu. On a intérêt à définir cette durée de simulation (exprimée en minutes) sur une période correspondant à un multiple du nombre de jours ouvrables. Dans notre exemple, on retiendra, dans un premier temps, deux semaines[1]. Implicitement, cette durée de simulation définit un échantillon sur lequel des indicateurs de performance seront mesurés (voir ci-après). On retrouve donc les problèmes classiques de distribution d'échantillonnage, ce qui fait que les valeurs d'indicateurs calculées sur une simulation seront des réalisations de variables aléatoires (dont la variance est d'autant plus faible que la simulation est longue).



Après simulation, on peut régénérer une nouvelle simulation utilisant d'autres nombres aléatoires que la simulation précédente, en procédant comme indiqué ci-contre; ce processus peut être répété autant de fois que souhaité.



On peut aussi vouloir garder sous contrôle l'ensemble des nombres aléatoires utilisés: il est modifiable en cliquant sur l'option de l'onglet Home. L'obtention d'autant de valeurs différentes d'indicateurs que de simulations différentes réalisées pose le problème d'une synthèse de ces informations laquelle passe par la détermination d'une estimation ponctuelle et d'un intervalle de confiance pour chacun de ces indicateurs, ce qui s'obtient facilement comme nous allons le voir à la page 48.

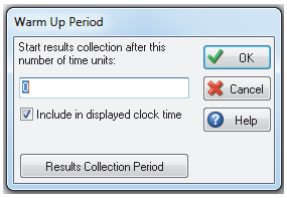


1. Deux semaines correspondent à 2 x 5 x 10,5 x 60 = 6 300 minutes (et 480 patients). Dans les simulateurs à événements discrets, une date de simulation (*Simulation time*) repère à la fois un instant ponctuel mais aussi le début de la période suivante ; en conséquence t = 6300' correspond au tout début du lundi d'une troisième semaine de simulation, au moment même où deux nouveaux patients arrivent au cabinet médical. En conséquence, si l'on désire exprimer cette durée en minutes, il convient de saisir une valeur légèrement inférieure (6299,99 par exemple)

III-1.3 Neutralisation du début de la simulation (*warm-up*)

Peu de processus se déroulent de manière ininterrompue, 24 heures sur 24, comme c'est le cas dans les services d'urgence ou aux péages d'une autoroute. Généralement, les systèmes productifs sont ouverts sur des plages de temps disjointes (par exemple de 8 heures à 20 heures). Il convient alors de distinguer deux cas de figure.

- On peut recommencer une période d'activité en retrouvant le système dans l'état exact où on l'avait laissé à la fin de la période précédente (*back-office* bancaire, par exemple), auquel cas, on peut modifier le repérage temporel pour ne garder que des heures ouvrées. Cette catégorie de système productif peut être qualifiée de **système permanent** [1]. D'un point de vue technique, la simulation de tels processus implique d'attendre un certain temps (régime transitoire) avant que le comportement du système soit indépendant des conditions initiales, c'est-à-dire que l'on soit en **régime de croisière**. La durée mise pour atteindre le régime de croisière est qualifiée de **préchauffage**[2] (***warm up***). Cette durée est généralement définie empiriquement: en régime de croisière, les niveaux de files d'attente varient autour de valeurs centrales, sans évolutions tendancielles; par ailleurs, le surdimensionnement de ce temps de préchauffage est sans problème. La définition de la durée de cette période, durant laquelle aucun recueil d'information statistique n'est effectué pour des raisons de pertinence, se fait:
  - soit par le bouton Warm Up Period de la fenêtre ouverte par l'option Properties du menu de l'onglet Data_Rules de la barre de menu général;
  - soit de la fenêtre ouverte par l'option Warm-Up period: 0.00 minutes du menu de l'option Advanced de la barre de menu général.



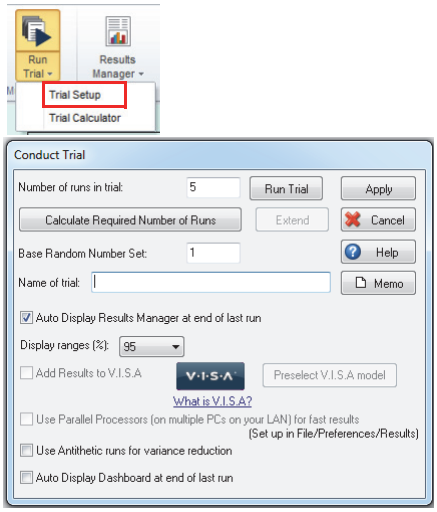
- D'autres processus sont tels qu'il y a systématiquement réinitialisation de l'état du système au début de chaque de période. Par exemple, l'état d'un centre d'appels à l'ouverture est le même d'un jour sur l'autre et est indépendant de son état en fin de journée la veille. Cette catégorie de système productif peut être qualifiée de **système à réinitialisation périodique**. Les simulations faites sur chacune des périodes sont indépendantes, contrairement au cas précédent.

Dans l'exemple du Cabinet médical, on est clairement en présence d'un système à réinitialisation périodique.

1. La littérature anglo-saxonne utilise le plus souvent l'expression *steady state system* pour désigner ce cas de figure et celui de *terminating system* pour désigner le second type.

2. Il s'agit d'une analogie mécanique, le préchauffage étant la période de lancement d'un moteur diesel avant que celui-ci ne soit opérationnel.

III-1.4 Jeu de simulation (*Trials*)

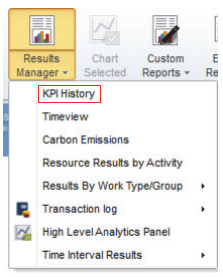


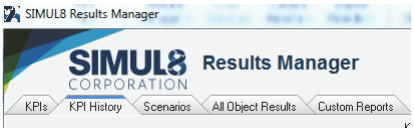
On peut automatiser ce processus de simulations multiples, pour obtenir un **jeu de simulations** (*trials*), en utilisant l'option de l'onglet Home qui recommence la simulation un nombre de fois défini par défaut à 5. Ce nombre est modifiable (voir ci-contre) dans la fenêtre ouverte par l'option Trial Setup de l'onglet Home de la barre de menu général (exemple: Number of runs in trial: 100), ce qui permet d'obtenir des intervalles de confiance de certains indicateurs synthétiques. Prenons comme exemple d'indicateur le temps moyen passé par les patients dans le cabinet médical sur deux semaines de fonctionnement du cabinet médical. Cet indicateur, calculé en fin d'une simulation correspond à la réalisation d'une variable aléatoire qui est une moyenne des temps passés par chaque patient traité au cours de la simulation. Le temps passé par un patient dans le système productif est l'occurrence d'une variable aléatoire dont les caractéristiques dépendent de l'état du système à l'entrée du patient. L'indicateur «temps moyen passé par les patients dans le cabinet médical sur deux semaines de fonctionnement du cabinet médical», calculé sur une simulation, est donc une moyenne de variables aléatoires **auto-corrélées** (voir page 16). Dès lors, il n'est pas possible d'utiliser la théorie classique de l'estimation pour établir l'intervalle de confiance du temps de séjour ***d'un patient quelconque***. Par contre, si au moins une trentaine de simulations sont réalisées dans les mêmes conditions, on peut invoquer le **théorème de la limite centrale** pour déterminer un intervalle de confiance de l'indicateur ***temps moyen passé par les patients dans le cabinet médical sur deux semaines de fonctionnement du cabinet médical***. Il est donc **impératif** de changer la valeur par défaut (5) du nombre de simulations et de la passer au moins à 30.

Lorsque l'on est en présence d'un système à réinitialisation périodique, *il est préférable de définir les indicateurs présentés dans le paragraphe suivant sur l'intervalle de temps séparant deux réinitialisations périodiques* (1 jour ouvrable dans notre exemple); la définition d'une période plus longue est arbitraire et conduit à une diminution de la dispersion des valeurs des indicateurs. Cela étant, on conservera ici une simulation sur deux semaines, qui, dans un premier temps est «plus parlante».

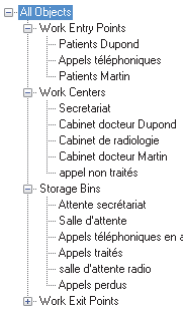
III-2 Analyse des résultats

Plusieurs types d'informations sont disponibles dans Simul8. Le bouton Results de l'onglet Home de la barre de menu général permet d'obtenir des tableaux synthétiques des indicateurs de performance retenus (KPIs pour *Key Performance Indicators*).





- L'onglet KPIs affiche les indicateurs explicitement retenus dans les fenêtres de résultats des activités, files d'attente, points d'entrée et points de sortie.
- L'onglet All Object Results affiche tous les points de passage d'un item (voir ci-contre); la sélection de l'un de ces points de passage déclenche l'affichage des informations que l'on obtient avec le bouton Results après double-clic sur le point considéré.



Start Points

	Number Entered	Number Lost	Net Number Entered
Patients Dupond	240	0	240
Appels téléphoniques	618	0	618
Patients Martin	240	0	240

Activities

	Waiting %	Working %	Blocked %	Stopped %	Number Completed Jobs	Minimum Use	Average Use	Maximum Use	Current Contents	Change Over %	Off Shift %	Resource Starved %
Secretariat	57,781	42,219	0	0	1412	0	0,46	1	0	0	0	0
Cabinet docteur Dupond	33,698	66,302	0	0	425	0	0,654	1	0	0	0	0
Cabinet de radiologie	53,52	46,48	0	0	379	0	0,465	1	0	0	0	0
Cabinet docteur Martin	33,918	66,082	0	0	434	0	0,652	1	0	0	0	0
appel non traités	100	0	0	0	166	0	0	1	0	0	0	0

Queues

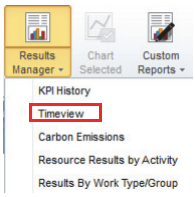
	Minimum Queue Size	Average Queue Size	Maximum Queue Size	Minimum Queuing Time	Minimum (Non-zero) Queuing Time	Average Queuing Time	Average (Non-zero) Queuing Time	Maximum Queuing Time	Number of Non-zero Queuing Times	% Queued Less Than Time Limit	"Queued Less Than" Time	St Dev of Queuing Time	Current Contents	Items Entered
Attente secrétariat	0	0,152	3	0	0,005	0,916	2,141	9,076	411	100	10	1,585	0	960
Salle d'attente	0	0,442	3	0	0,031	3,277	5,413	19,114	520	78,231	6	4,204	0	859
Appels téléphoniques en attente	0	0,028	3	0	0,015	0,344	0,787	1	270	100	10	0,446	0	618
Appels traités	0	236,059	452	0	0	0	0	0	0	0	10	0	452	452
salle d'attente radio	0	0,102	2	0	0,028	1,685	4,314	14,457	148	98,681	10	2,902	0	379
Appels perdus	0	86,927	166	0	0	0	0	0	0	0	10	0	166	166

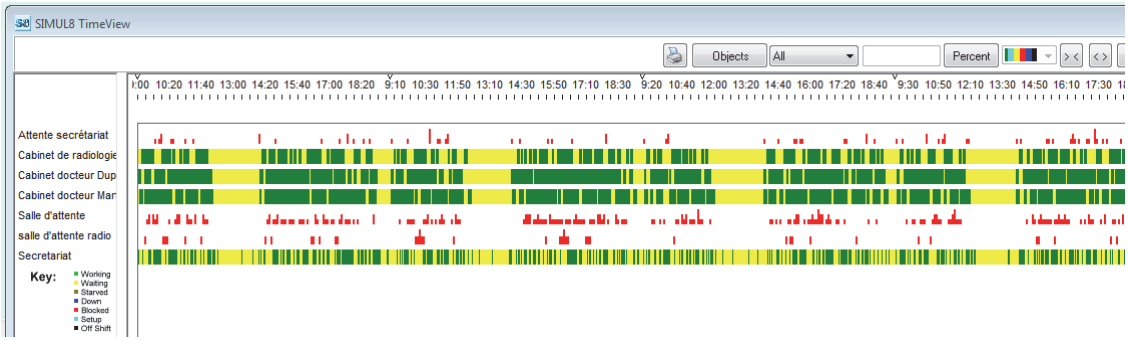
Ends

	Average Time in System	Number Completed	"In System Less Than" time	% In System Less Than Time Limit	St Dev of	Maximum Time in System	Minimum Time in System
Sortie patients	37,19	480	45	73,125	10,939	59,621	11,619

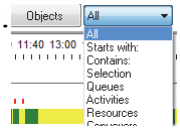
Le choix des «éléments» à étudier dans l'analyse d'un jeu de simulation est fait en ouvrant la fenêtre de la classe d'éléments (activité, file d'attente, point d'entrée, point de sortie, ressources) listée en bas de la liste d'options offerte du menu Results. Une fois la fenêtre ouverte, il ne reste plus qu'à inclure les indicateurs de l'objet sélectionné dans la liste (Add all results (selected above) to KPI's) ou, au contraire, de les exclure (Remove all results (selected above) from KPI's). En cliquant sur l'objet, on peut aussi faire apparaître directement sa fenêtre de résultats.

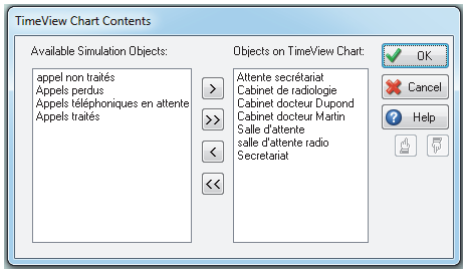
Une **synthèse graphique** de l'évolution de l'utilisation des activités, des files d'attente (et des ressources) est disponible dans la fenêtre «***Timeview***» (voir ci-dessous) ouverte par l'option TimeView du menu affiché en sélectionnant Results Manager dans l'onglet Home de la barre de menu général (voir ci-contre). On obtient alors le graphe de la page suivante.





La sélection des informations souhaitées dans l'affichage de TimeView s'effectue dans une fenêtre ouverte par le bouton Objects de la fenêtre Timeview ou en sélectionnant une catégorie de points de passage.

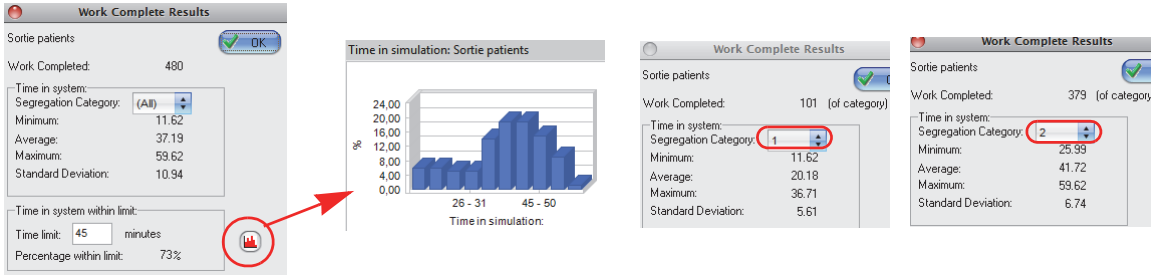




Examinons plus en détail les informations que l'on peut obtenir à l'issue d'une simulation.

III-2.1 Informations collectées au point de sortie

On remarque que 480 patients ont été reçus, ce qui est cohérent, puisque chaque docteur prend 120 rendez-vous par semaine. C'est à cet endroit que s'effectue l'analyse du temps passé X par un patient dans le cabinet, avec une simulation de deux semaines conduit à $\bar{x} = 37{,}19$ $\sigma_x = 10{,}94$, Min = 12,09, Max = 56,5 et P ($X < 45$) = 75%. Rappelons que l'on a ici neutralisé les temps de déplacement des patients à l'intérieur du «système productif» étudié parce qu'insignifiants (voir page 46). L'histogramme de la distribution de probabilités de ces temps de séjours est fourni.



On peut obtenir ces informations pour chacun des sous-ensembles définis par la valeur d'un label, pour tenir compte de l'hétérogénéité de l'ensemble étudié. Par exemple, on peut utiliser une **segmentation** basée sur la possession préalable d'une radio; comme la valeur du *label* «L Possession radio» peut être modifié dans la simulation, il faut en faire une copie à l'entrée (recopie, sous *Visual Logic*, de la valeur initiale dans le *label* «L Possession initiale Radio»; ensuite, avant la simulation, on sélectionne l'option Segregate Results dans la fenêtre du point de sortie et l'on choisit le *label* qui assure la partition de l'ensemble étudié (Segregate Results (L Possession initiale Radi); dans notre exemple, la fenêtre de résultats comporte un pop-up permettant de sélectionner l'ensemble désiré (par défaut, l'ensemble complet Segregation Category: (All)); l'analyse de

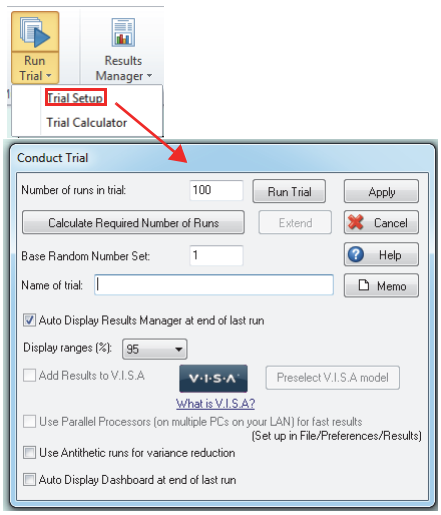
l'ensemble des patients ne possédant pas initialement de radio (Segregation Category: 2) conduit à des valeurs sensiblement différentes des valeurs de l'ensemble ($\overline{x}_2 = 40{,}94$, $\sigma_{x_2} = 6{,}55$, $Min_2 = 27{,}49$, $Max_2 = 56{,}5$; l'information $P(X_2 < 45)$ et l'histogramme ne sont pas disponibles[1]). Cette segmentation des résultats est utile pour analyser les résultats de la simulation de processus dans lesquels les arrivées sont aléatoires, avec des caractéristiques variant au cours de la journée (arrivées de type *Time dependant* ; voir page 43 et l'exemple Distribution_Time_Dependant_partition.S8.

Insistons une dernière fois sur le fait que les temps de séjour des patients étant **auto-corrélés**, cette dépendance des observations implique ***que l'on ne peut pas déterminer un intervalle de confiance du temps de séjour d'un patient*** car il dépend de l'état du système lorsqu'il rentre dans le système. La fréquence associée à un intervalle de temps de séjour calculée dans une simulation varie d'une simulation à l'autre et doit être interprétée avec prudence.

L'analyse d'un jeu de simulation obtenu par le bouton de l'onglet Home de la barre de menu général est offerte dans la fenêtre qui s'ouvre à la fin de ces simulations multiples. Les informations disponibles sont les mêmes que celles proposées à l'issue d'une simulation avec, ce qui fait la différence, un calcul d'intervalle de confiance de ces KPIs. Dans notre exemple, avec 100 simulations sur 1 jour ouvrable (629,9 minutes), l'application de la théorie de l'estimation permet de dire que la durée moyenne quotidienne de présence des patients au cabinet médical a 95% de chance d'être compris entre 36,57 et 37,33 minutes[2].

		Low 95% Range	Average Result	High 95% Range
Sortie patients	Average Time in System	36.57	36.95	37.33
	Number Completed	48.00	48.00	48.00
	"In System less than" time		*45.00*	
	% In System less than time limit	74.49	75.92	77.35
	St Dev of	9.93	10.12	10.31
	Maximum Time in System	53.15	53.77	54.40
	Minimum Time in System	13.97	14.28	14.59

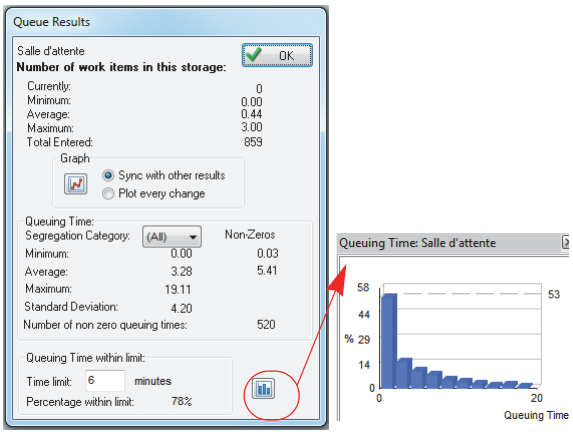
Le coefficient de confiance retenu est modifiable en cliquant sur les valeurs par défaut proposées. La fenêtre ouverte en cliquant sur l'option Run Trial offerte en utilisant le bouton de l'onglet Home de la barre de menu général (voir ci-contre) permet de modifier le nombre de simulations associé à un jeu de simulation Number of runs in trial: 100, de définir le générateur de nombre aléatoire à utiliser Base Random Number Set: 1, le coefficient de confiance associé à l'intervalle de confiance désiré pour les indicateurs Display ranges (%): 95. On ne s'intéressera pas ici aux autres options disponibles qui sont très techniques.



1. sauf à utiliser deux points de sortie différents.
2. L'analyse des résultats détaillés montre que l'écart-type estimé de la durée moyenne de séjour pendant une journée, sur 100 simulations est 0,663 ; l'application de la théorie classique de l'estimation permet de calculer l'intervalle de confiance trouvé. Il convient de souligner que si les durées de séjour dans une simulation sont corrélées, les durées moyennes de différentes simulations d'un même processus sont indépendantes à partir du moment où elles utilisent des générateurs de nombres aléatoires différents.

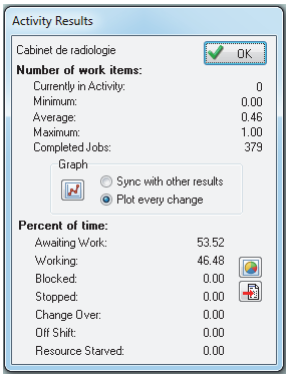
III-2.2 Informations collectées dans une file d'attente

L'analyse du temps Z passé par les patients dans la salle d'attente correspond aux informations fournies ci-contre. On pourrait aussi utiliser le principe de segmentation dans l'analyse des résultats mais, ici, l'usage de la segmentation des patients selon la possession initiale d'une radio n'est pas pertinent car, dans l'analyse statistique, les temps d'attente d'un patient passant deux fois en salle d'attente ne sont pas cumulés.



III-2.3 Informations collectées dans une activité

Analysons maintenant l'utilisation des activités. Commençons par l'exemple du cabinet de radiographie, on constate une certaine sous-utilisation de cette activité qui ne travaille qu'à 46,48% de son temps (53,52% d'attente). Cette information doit cependant être corrigée car elle est calculée comme le quotient du temps travaillé par le temps ouvré (10h30); si l'on tient compte d'une pause de 1 heure 30, il faut multiplier cette information par 10,50/(10,5-1,5)=1,167, ce qui donne un taux d'occupation corrigé de 54,24%. Cette sous-utilisation (que l'on retrouve pour la secrétaire et l'opérateur radio) met en évidence qu'il est sans doute possible d'adjoindre un troisième médecin à ce cabinet (si un réaménagement est possible), sans dégradation notable du niveau de service (point à vérifier par la simulation).

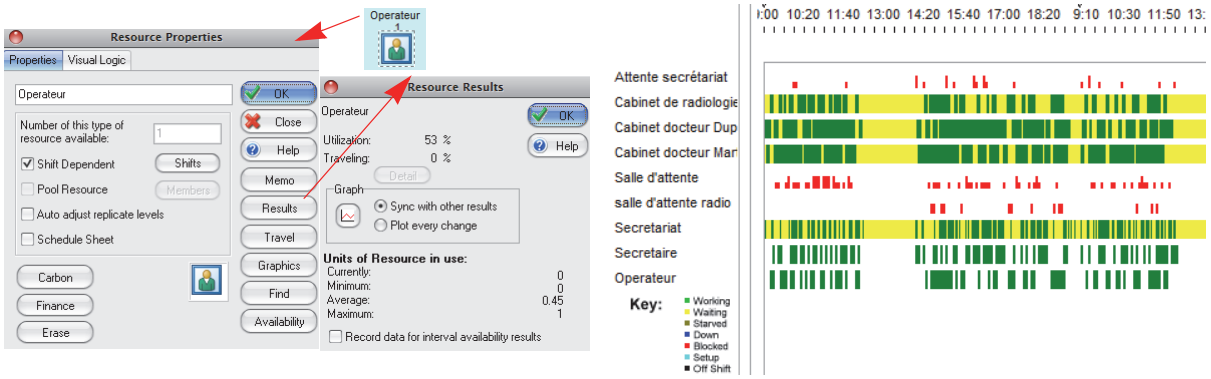


La non-prise en compte de la pause du midi fausse l'appréciation visuelle de l'utilisation du système productif. Deux transformations de la modélisation sont possibles pour traiter correctement ce problème. Elles passent par la notion d'équipe (*shift*) à laquelle on associe une ou plusieurs plages horaires de présence. Ici, on considérera deux plages de présence[1]: 9 heures- 12h30 et 14 heures-19h30.

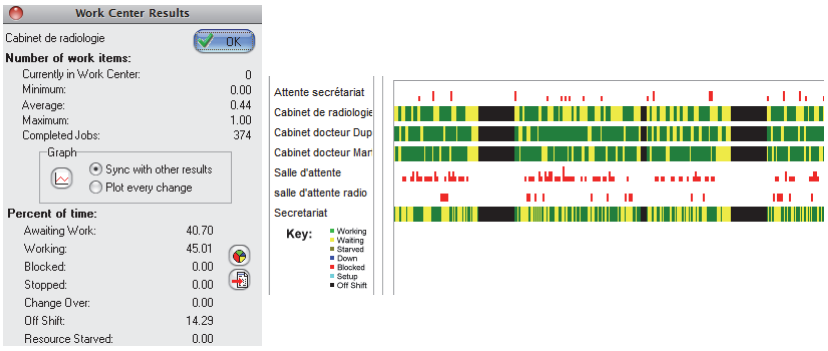
- La ***première solution*** (voir exemple (DOCTEUR_ressource.S8)) consiste à créer des types de ressource, ici la ressource secrétaire, la ressource opérateur et les ressources Martin et Dupond, puisque ces docteurs ne sont pas interchangeables. Cette création passe par l'usage du bouton Resource de la barre de menu; après avoir positionné la ressource sur représentation du système production, il faut faire un double-clic sur l'icône qui ouvre (voir ci-dessous) une fenêtre permettant de spécifier les plages de disponibilité de chaque type de ressource, ainsi que leur nombre (un type de ressource peut concerner un ensemble de personnes). Les ressources sont représentées par

1. Dans la réalité, la pause débute quand tous les patients ont été traités et non à heure fixe, ce qui peut conduire à une fin de consultation juste après le début de la pause (par défaut, tout travail commencé est terminé par l'activité) et à un paiement auprès de la secrétaire, à sa reprise de travail en début d'après-midi.

des icônes dans le modèle. On obtient directement un taux d'utilisation effectif de 53% pour l'opérateur radio (valeur légèrement différente de celle trouvée précédemment en raison de la différence de modélisation qui conduit à utiliser d'autres réalisations des variables aléatoires). On reviendra plus en détail sur la modélisation des ressources (pages 68 et suivantes).



- Une ***seconde solution*** (voir exemple (DOCTEUR_Shift.S8)) consiste à appliquer le concept de plages de disponibilité aux activités. En cliquant sur le bouton Shifts d'une activité, on fait apparaître une fenêtre dans laquelle on coche l'option de l'activation de l'activité seulement sur des plages de temps Use shift patterns directly que l'on spécifie avec le bouton Allocation qui ouvre une fenêtre listant les plages de temps affectées à l'activité (au départ aucune), le bouton Shift Work Patterns permet de lister les plages déjà créées, d'en créer de nouvelles et d'affecter celles choisies à l'activité considérée. Cette solution ne modifie pas le fait que les indicateurs soient calculés sur la base du temps d'ouverture du système productif (10h30) et non le temps d'ouverture de l'activité (9 heures); elle fait simplement apparaître, dans la décomposition du temps d'utilisation de l'activité, une rubrique *Off shift* qui correspond à la fermeture de l'activité pendant le temps d'ouverture du système productif.

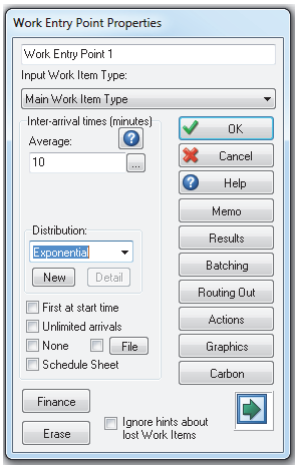


Chapitre III

CARACTÉRISATION DES *ITEMS*, DES POINTS D'ENTRÉE ET DES POINTS DE SORTIE

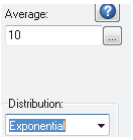
La modélisation d'un processus de production de biens ou de services vise à décrire la circulation, dans un système productif, d'objets au sens large (personnes, dossiers, produits, informations…) qui arrivent dans ce système pour y subir un ensemble de traitements, après quoi ils le quittent. On retient ici le terme générique item pour désigner ces objets, en s'alignant sur la terminologie retenue par Simul8. On parle de **cycle de vie** d'un item pour décrire la succession de points de passage d'un item dans un système productif, entre son arrivée et son départ du système.

I ARRIVÉE DES ITEMS DANS LE SYSTÈME PRODUCTIF



Les items entrent dans le système productif à un **point d'entrée**, créé par l'outil Start Point (voir exemple page 25), puis transitent par une succession de files d'attente et d'activités avant de sortir par un point de sortie créé par l'outil End. Plusieurs points d'entrée peuvent coexister. Pour faciliter la modélisation, on peut utiliser fictivement plusieurs points d'entrée, comme cela a été fait dans l'exemple du cabinet médical. Le cheminement d'un item dans le système productif obéit à des règles sur lesquelles on reviendra à la page 113. Les items quittent un point d'entrée pour se diriger vers une (ou plusieurs) file(s) d'attente en utilisant l'une des règles décrites à la page 59.

I-1 Lois d'arrivées des items



Les **arrivées** des items à un **point d'entrée** (***Entry Point***) sont définies par des distributions de probabilités (appelées encore lois d'arrivées) qui sont de deux types.

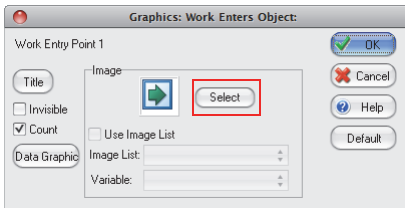
- On peut tout d'abord définir ces arrivées par une distribution de probabilités de l'**intervalle de temps séparant deux arrivées successives** d'items au point d'entrée considéré. Deux cas de figure doivent être distingués. Les caractéristiques de ces lois peuvent être stables, auquel cas la loi peut directement être définie au point d'entrée par l'une des distributions statistiques disponibles du menu déroulant (

). Ces caractéristiques peuvent varier dans le temps, ce qui conduit à une loi par plage de temps (distribution *Time dependant*), ce qui a été présenté en détail aux pages 43 et suivantes.
- On peut également définir une programmation des arrivées au point d'entrée, de type carnet de rendez-vous, comme on l'a utilisé dans l'exemple du cabinet médical (*Time absolute*). Ces arrivées ne se font pas au moment exact

prévu mais avec aléa se traduisant par une avance ou retard en fonction d'une distribution de probabilités prédéterminée, ce qui a été présenté en détail aux pages 41 et suivantes.

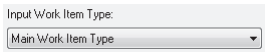
On peut demander qu'une arrivée se fasse au démarrage de la simulation en cochant First at start time.

Un point d'entrée peut être remplacé par une **file d'attente** comportant au démarrage à un certain nombre d'items (via le bouton Start - Up de la file d'attente) dont on détermine les caractéristiques comme on le fait à un point d'entrée (via le bouton Label Actions). Cette initialisation peut être déterministe (valeurs de labels prédéterminées) ou stochastique (valeurs de labels générées aléatoirement). On étudiera cette possibilité à la page 65.



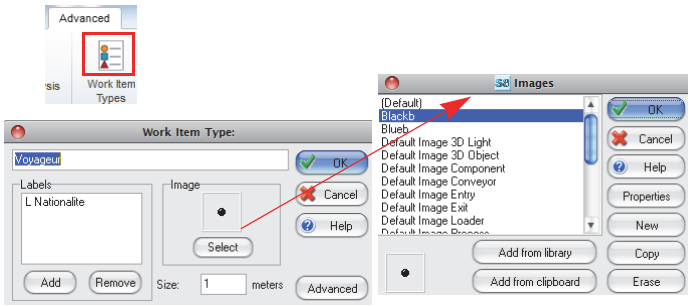
L'**icône** par défaut d'un point d'entrée peut être remplacé par une autre image de la bibliothèque des images dans la fenêtre ci-contre, ouverte à partir du bouton Graphics de la fenêtre des propriétés du point d'entrée. C'est également dans cette fenêtre que l'on peut demander l'affichage du nom du point d'entrée Title et la neutralisation de l'affichage du décompte du nombre d'items entrés (option par défaut) en décochant Count.

I-2 Type d'un item

L'item entrant appartient obligatoirement 

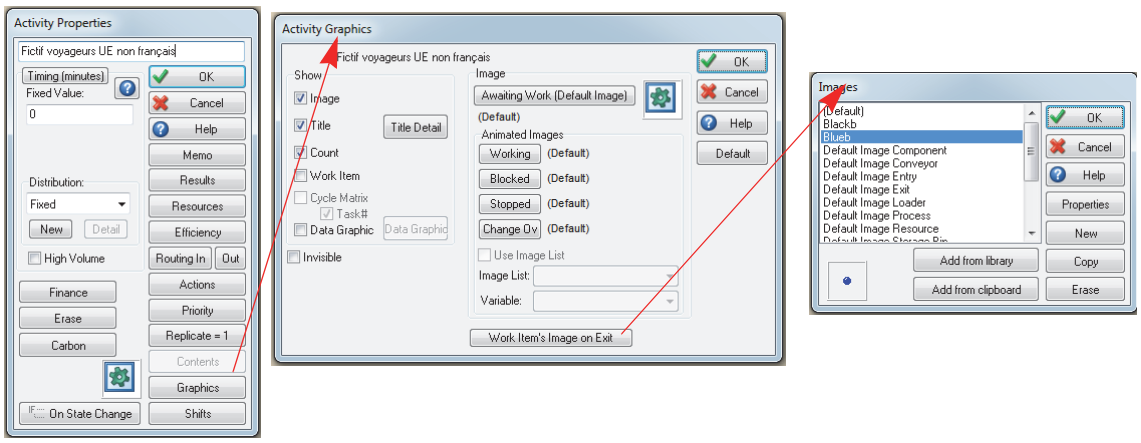
à un **Type d'items** (*Main Work Item Type*), qui peut être unique et être celui défini par défaut (Main Work Item Type). La création des types d'item a été illustrée à la page 26. Un seul type d'items n'est autorisé à un point d'entrée.

La modification de l'**icône** et du **nom** attribués par défaut est effectuée dans la fenêtre de propriétés d'une classe d'items, ouverte par l'option de l'onglet Advanced de la barre de menu principal; on modifie l'icône en choisissant l'un des fichiers de la bibliothèque d'images de Simul8. Cette icône par défaut d'un type d'items reste normalement la même au cours de la simulation pour tous les items appartenant à un même type. L'icône d'un item (mais pas celle d'un type d'item) peut cependant être changée à la sortie d'une activité (pour visualiser un «changement d'état», par exemple), sans que cet item change pour autant de type (voir paragraphe suivant). La taille de l'item dans la simulation (Size: 0.1 meters) est modifiable. Cette information de taille est utile lors de l'utilisation des convoyeurs (page 95).



I-3 Modification possible de l'icône d'un item en sortie d'une activité

En sortie d'une activité, l'icône de l'item entrant peut être modifiée, ce qui permet d'attirer l'attention sur des transformations de l'état d'un item. Cette modification s'effectue à partir de la fenêtre de propriétés de l'activité, comme suit, la fenêtre de sélection de l'image de l'icône étant la même que ci-dessus.



Dans l'exemple PoliceAeroport.S8, les arrivées de passagers à la Police des frontières d'un aéroport, l'ensemble de guichets de destination d'un passager varie avec sa nationalité (ainsi que l'icône utilisée).

Ajoutons que l'icône d'un item peut être conditionnée par la valeur prise par un label. Il suffit alors de modifier l'icône de l'item via Visual Logic après avoir sélectionné Label Actions. Ce changement conditionnel peut se faire dans une activité ou à un point d'entrée, ce qu'illustre l'exemple Icone_Item_en_sortie.S8 d'arrivées à un point d'entrée, l'icône des items varie selon que l'on a affaire à un homme ou une femme (usage de *Visual Logic* nécessaire, contrairement à l'exemple précédent).

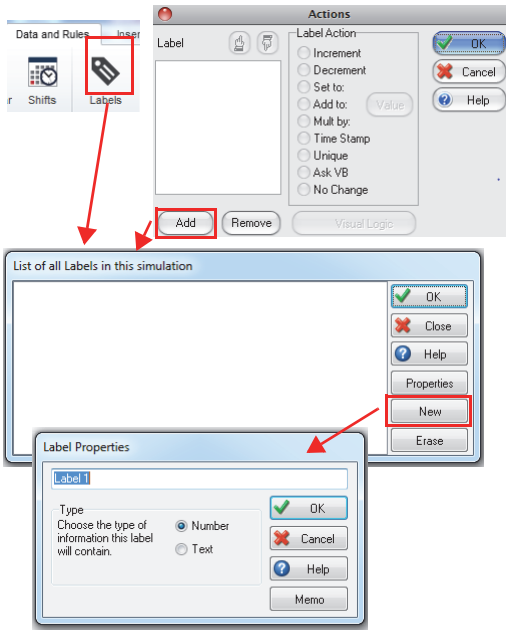
I-4 Arrivée par lot d'items

Par défaut, chaque entrée porte sur un seul item. On peut vouloir modéliser le fait que chaque arrivée porte sur un groupe d'items (par exemple, le client donneur d'ordre d'un fastfood est une personne seule ou le représentant d'un groupe de clients (famille, etc.)). L'utilisation du bouton Batching du point d'entrée permet de définir la loi de la taille de groupe (variable certaine ou réalisation d'une variable aléatoire), sachant qu'en suite chaque item du groupe est traité individuellement (au niveau des actions sur les labels et en sortie); avec cette solution, on perd l'information sur le groupe auquel il appartient. S'il est utile de la conserver, il vaut mieux alors travailler en 2 temps avec l'arrivée des groupes puis utilisation d'une activité fictive qui éclate en sortie (Out, puis Batching) le groupe en un nombre variable ou fixe d'items ayant leurs caractéristiques propres; ces deux possibilités sont illustrées par l'exemple Arrivee_items_par_groupe.S8.

I-5 Labels d'un item

Une première caractérisation des items par des valeurs spécifiques de *labels* est réalisée aux points d'entrée par l'intermédiaire du bouton Actions. Ce point a été largement illustré dans l'exemple introductif aux pages 25 et suivantes. La liste des actions possibles est la même que celles disponibles pour une activité (voir page 26).

Les **attributs** (***labels***) caractérisent l'item entrant; ils correspondent aux variables qualitatives ou quantitatives que l'on associe, en statistique descriptive à chaque élément de l'ensemble étudié. Ils sont créés dans la fenêtre ouverte par l'option de l'onglet Data Rules ou par le bouton Add a Label to Change de la fenêtre ouverte par le bouton Actions de la fenêtre de propriétés d'une activité ou d'un point d'entrée. Tous les items d'une classe d'items se caractérisent par la même liste de *labels*, même si certains ne sont pas toujours utilisés. Les *labels* permettent de fournir des caractéristiques intrinsèques de l'item qui permettront, principalement, de définir les gammes opératoires qui lui seront appliquées.

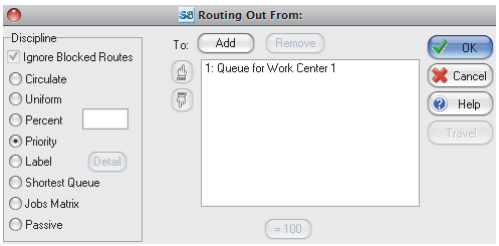


D'une manière générale, la valeur d'un *label* d'un item définie par une activité (Actions) peut être ensuite modifiée par un programme en *Visual Logic* de cette même activité (dans la fenêtre ouverte par le bouton Visual Logic). Elle peut l'être également dans une activité dans lequel l'item passe ultérieurement. Enfin, dans quelques cas peu fréquents, elle peut l'être à l'entrée d'une file d'attente ou à sa sortie (voir page 67). Les possibilités de **définition de la valeur d'un *label*** sont les suivantes, en commençant par celle qui est la plus fréquemment utilisée et abondamment illustré dans l'exemple introductif du cabinet médical:

- Set to: Assignation au *label* de la **réalisation d'une variable** certaine (voir exemple page 27) ou aléatoire (voir exemple page 28).
- Unique Assignation d'un **numéro d'ordre unique** à un item entrant, quel que soit le point d'entrée et sans doublon possible; cette valeur est assignée chronologiquement à un *label* sans que ce *label* ne soit nécessairement le même (autrement dit, plusieurs labels peuvent partager une même liste de rang d'arrivée dans le système productif). Il n'est pas possible de définir avec cette option plusieurs numéros d'ordre unique associés à des *labels* différents; pour obtenir, des incrémentations locales différentes complémentaires, il faut mobiliser un programme en *Visual Logic*, ce qu'illustre l'exemple Unique.S8.
- Increment **Incrémentation de la valeur d'un *label* numérique**; ceci a un sens postérieurement au point d'entrée (au début, il faut utiliser l'option Set to: puis saisir la valeur initiale du *label* considéré, normalement 1). L'une des utilisations intéressante de cette possibilité est de pouvoir déterminer le «nombre d'étapes» franchies dans un processus dans lequel un item peut être traité plusieurs fois par un même centre de production.
- Decrement **Décrémentation de la valeur d'un *label* numérique**; mêmes remarques que précédemment.
- Add to: **Ajouter** la valeur courante d'un *label* numérique à un autre *label* numérique (utile dans certains cas).

- Mult by: **Multiplier** la valeur courante d'un attribut numérique par une constante (utile dans certains cas).
- Time Stamp Le **chronomarquage d'un item**, généralement à un point d'entrée, est souvent utilisé pour suivre les items dans le système productif, dans la mise au point du modèle.
- Ask VB **Asq VB** conduit à définir par un programme en *Visual Logic* la valeur prise par l'attribut. On peut aussi donner une valeur par défaut et la modifier par un programme en *Visual Logic*. L'usage de *Visual Logic* s'impose quand la valeur que peut prendre un *label* est conditionnée par celle prise par un autre *label* (dans l'exemple du cabinet médical, la valeur du *label* «consultation» dépendait de celle du *label* «L Possession radio»). On reviendra sur ce point au § I-7.

I-6 Les sorties du point d'entrée



Normalement, un item entrant dans le système productif quitte son point d'entrée pour se rendre directement dans une file d'attente. On peut l'adresser à une activité, à condition qu'elle soit fictive (temps opératoire nul). Lorsque plusieurs directions sont possibles en sortie, se pose un problème de choix de la direction à prendre qui est exactement le même que celui qui se pose à la sortie d'une activité. Cette problématique a été traitée en détail pour l'option Label à la page 31. Pour les autres options, reportez-vous à la description qui en est faite à la page 87 qui traite en détail ce problème de la détermination de la direction en sortie d'une activité.

Ajoutons, deux remarques pour terminer un item entrant est «perdu» s'il ne trouve de place où aller (par exemple, la file d'attente vers lequel il est dirigé est saturé (capacité d'accueil limitée, voir page 62). Pour éviter de «perdre» cet item, il est possible si l'option Priority est retenue, d'ajouter en fin de liste des directions en sortie, un point de sortie (ou une file d'attente) qui recueillera l'item n'ayant pas trouvé de place dans le ou les files d'attente placées plus haut dans la liste.

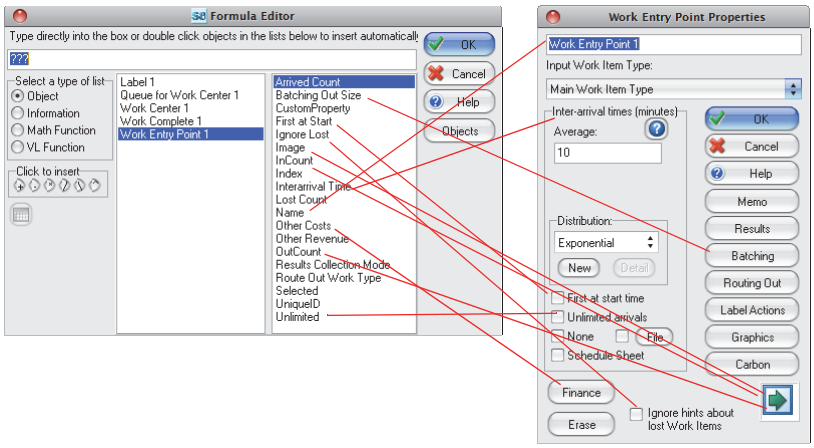
I-7 Point d'entrée et *Visual Logic*

Il est possible d'exécuter un programme en *Visual Logic* au point d'entrée. Il est également possible d'exploiter les propriétés d'un point d'entrée dans une instruction en *Visual Logic*, en lecture et, pour certaines propriétés, en écriture.

I-7.1 Possibilité d'exécution de programmes en *Visual Logic* au point d'entrée

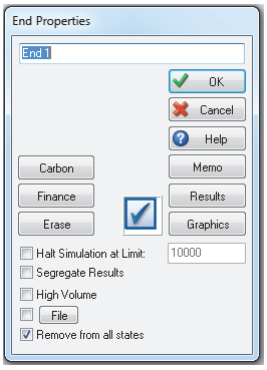
Un programme en *Visual Logic* peut être écrit dans la fenêtre ouverte par le bouton Visual Logic de la fenêtre ouverte par le bouton Actions de la fenêtre de propriétés du point d'entrée. Cette possibilité a été illustrée à la page 39.

I-7.2 Propriétés d'un point d'entrée mobilisables par *Visual Logic*



Les instructions de *Visual Logic* peuvent utiliser des propriétés de la classe «Point d'Entrée». Il s'agit d'un ensemble de caractéristiques prédéterminées, associées à n'importe quel Point d'Entrée et jouant le même rôle que les *labels*. Ces caractéristiques ont des valeurs par défaut utilisées lors de la création d'un Point d'Entrée pendant la modélisation d'un processus. Sont accessibles sous *Visual Logic* les différentes propriétés d'un point d'entrée que vous pouvez définir dans sa fenêtre de propriétés ou dans l'une des fenêtres que vous pouvez ouvrir à partir d'un bouton.

II SORTIE DES ITEMS DU SYSTÈME PRODUCTIFS



L'item sort obligatoirement, après traitement, par un **point de sortie** ☑ (*end*) au niveau duquel des résultats synthétiques sont disponibles (voir page 51). Toutefois, certains items peuvent rester en permanence dans le système productif. Il s'agit, par exemple, d'équipements associés à la logistique interne, comme des conteneurs ou des kanbans; il peut s'agir aussi de biens loués qui, après utilisation, reviennent pour être offerts à nouveau à la location.

À des fins de contrôle lors de la mise au point d'un modèle de simulation, un point de sortie peut être remplacé par une «**file d'attente fictive**» dont le contenu, en fin de simulation, peut être analysé. C'est la raison pour laquelle cette dernière solution est souvent utilisée dans les simulations créées pour illustrer ce cours. L'option ☑ File permet d'enregistrer dans un fichier toutes les informations caractérisant les items sortants, avec leur date de sortie et les dernières valeurs prises par leurs *labels*, ce qui est un peu plus compliqué mais peut permettre des exploitations plus poussées, pour peu que certains événements complémentaires (date d'arrivée de l'item dans le système productif, par exemple) aient été enregistrés dans des *labels*. Il est à noter que l'option ☑ Segregate Results est sans effet si cette file d'attente n'est pas connectée, en aval, à une activité (ce qui fait l'on n'est plus en présence d'une file d'attente fictive.

L'intérêt d'une **segmentation** de l'ensemble des items traités dans une simulation a été présenté à la page 51 (option ☑ Segregate Results . aux points de sortie du système productif). Le **critère de la période d'arrivée** des items, correspondant au découpage de la loi des arrivées suivant une loi *Time dependant* est souvent judicieux car l'adaptation des ressources à cette variation des caractéristiques de la

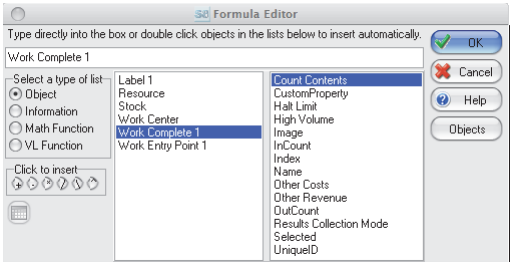
demande n'est pas toujours possible, ce qui conduit à des variations explicables de certains indicateurs de niveau de service. Il convient alors d'attribuer aux items entrants un *label* dont la valeur correspond à la plage horaire de leurs arrivées. Cette détermination est relativement aisée et est expliquée et illustrée par l'exemple Distribution_Time_Dependant_partition.S8.

L'option High Volume ne peut être utilisée que si elle a déjà été cochée pour une activité. Cette option est utile dans le traitement de lots de grande taille, susceptibles d'être fractionnés.

Évoquons pour terminer un problème que l'on rencontre dans certains services. Habituellement, un item est présent dans le système productif tant qu'il n'a pas subi tous les traitements qu'il a à subir. La production d'un service suit généralement cette logique sauf quand cette prestation consiste en une mise à disposition d'un ensemble de services utilisables sans restriction pendant une période donnée, ces prestations étant offertes simultanément à un ensemble de clients (parc d'attractions, domaine skiable…). Les arrivées des items (clients) dans le système productif ne se font pas simultanément en début de période et les clients restent le temps qu'ils veulent à condition de quitter le système productif avant sa fermeture. L'exemple Sortie_du_systeme_fonction_du_temps.S8 montre comment faire pour que les clients quittent le système avant l'heure de fermeture, le moment du départ étant défini à la minute près et le temps de séjour désiré pour un item pouvant varier (sans violation possible de la contrainte de fermeture). On qualifiera de type de sortie, de **sortie aléatoire**.

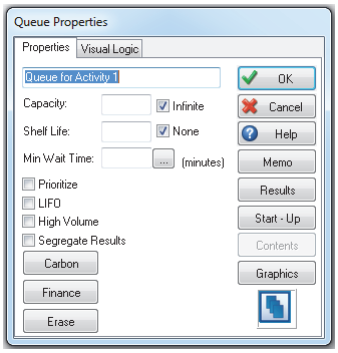
Pour terminer, rappelons que l'analyse des résultats en sortie a déjà été commentée en détail à la page 51.

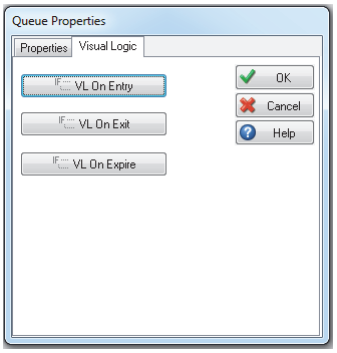
Ajoutons enfin qu'aucun programme *Visual Logic* ne peut être exécuté à ce niveau. Si cela est nécessaire, il suffit de faire précéder le point de sortie d'une activité fictive. Par contre, les propriétés d'un point de sortie sont accessibles par des instructions en *Visual Logic*.



Chapitre IV

CARACTÉRISATION DES FILES D'ATTENTE

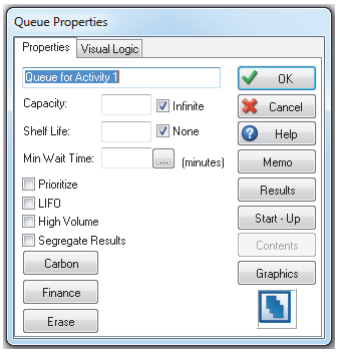




Une file d'attente peut **recevoir** des **items** provenant de différents points d'entrée, ce qui ne se justifie que si ces points d'entrée permettent de caractériser de manière différente les items entrants. De la même façon, une file d'attente peut recevoir des items provenant d'un ou plusieurs centres de production. Une file d'attente est toujours passive, dans le sens qu'elle n'envoie pas d'items vers les activités sans que celles-ci n'en fassent explicitement la demande. Il est possible de déclencher un programme en *Visual Logic* à l'entrée ou à la sortie d'un item, d'une file d'attente (onglet Visual Logic). Par ailleurs, l'un des programmes en *Visual Logic* lancé par un processeur quelconque peut à tout moment récupérer les informations caractérisant la file d'attente (par exemple, nombre d'items dans la file d'attente au moment de l'exécution de l'instruction en *Visual Logic*) et intervenir sur certaines de ses caractéristiques, comme sa capacité (voir exemple, page 93) ou celles (*labels*) des items qu'il contient. La fenêtre de définition des propriétés d'une file d'attente permet d'associer à la file d'attente plusieurs propriétés.

I PROPRIÉTÉS DE BASE D'UNE FILE D'ATTENTE

I-1 Capacité maximale

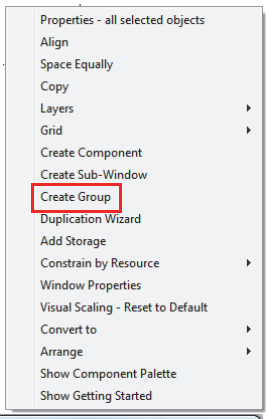


On étendra la notion de capacité maximale d'une file d'attente à celle d'un groupe de files d'attente et d'activités (fictives ou non) qui permet d'avoir sous contrôle le nombre maximal d'items pouvant se trouver dans un sous-ensemble du système productif.

I-1.1 Capacité maximale d'une file d'attente

Par défaut, la **capacité** (Capacity), c'est-à-dire le nombre maximal d'items que la file d'attente peut recevoir à un instant donné, est infinie (Infinite). Cette valeur peut être modifiée lors de la conception du modèle en saisissant une valeur dans la fenêtre ci-contre ou, à tout moment au cours de la simulation, par un programme en *Visual Logic* (voir exemple de la page 93). Dans cette perspective, il faut noter que conventionnellement, la valeur -1 est associée à une capacité infinie.

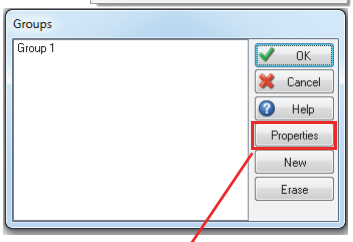
I-1.2 Capacité maximale d'un Groupe

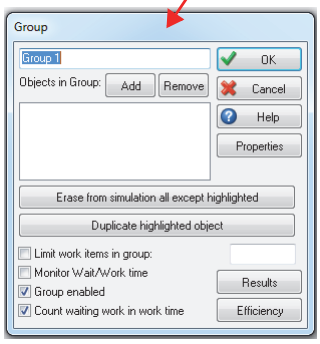


Le **groupe** est un ensemble de files d'attente et/ou d'activités. Cet ensemble se définit

- soit par une sélection directe de ces objets sur la représentation du processus suivi d'un clic-droit, ce qui provoque l'affichage de la fenêtre ci-contre; la sélection de l'option Create Group provoque alors un affichage fugitif d'une fenêtre indiquant que le groupe est créé et en lui donnant un nom par défaut;

- soit en utilisant l'option Groups de l'onglet Data and Rules, ce qui provoque l'affichage de la fenêtre ci-contre qui liste les groupes déjà créés et permet d'en créer un nouveau avec le bouton New, mais ce dernier ne comprend aucune file d'attente ni activité.

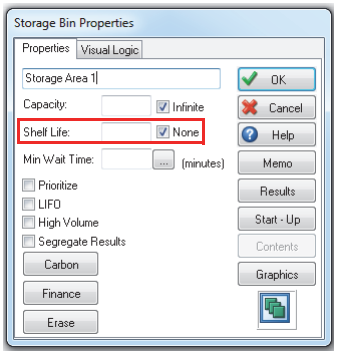




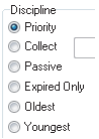
Il faut sélectionner le groupe existant ou venant d'être créé pour en modifier les caractéristiques, à l'aide du bouton Properties qui permet de modifier la liste des membres (files d'attente et activités) du groupe (Objects in Group: Add Remove). La propriété d'un groupe justifiant sa création pour certaines simulations est le nombre maximal d'items pouvant se trouver simultanément dans le groupe (Limit work items in group:). L'exemple Synchronisation_stock_produits_Multiples.S8 (voir page 116) illustre l'utilisation de ce concept de groupe réunissant une activité fictive chargée de prélever un item ayant une valeur particulière de label, dans un ensemble de files d'attente, et la file d'attente dans lequel l'item prélevé est rangé dans une file d'attente avant d'être appelé par une autre activité. L'exemple Box.S8 traite le cas de plusieurs box d'un service d'urgence; le patient est installé dans le box par une infirmière; il attend ensuite qu'un docteur fasse un diagnostic, en présence de l'infirmière; puis il attend qu'une infirmière fasse un prélèvement sanguin avant d'être renvoyé en salle d'attente; ce qui se passe dans ce box est modélisé par 3 activités et 2 files d'attente qui constituent un groupe ne pouvant accueillir qu'un seul item. Cet exemple, montre également deux autres possibilités pour bloquer l'entrée dans un box occupé, utilisant quelques instructions en *Visual Logic*; la première bloque l'accès au box si le cumul des items présents dans les 3 *activités* et les 2 files d'attente est égal à 1; la seconde utilise un indicateur d'occupation (paramètre mis à jour dans l'*Information Store*) et est la seule façon de traiter astucieusement certains problèmes complexes. Une variante de l'exemple précédent dans lequel se pose un problème d'***open shop*** (plusieurs opérations à effectuer dans un ordre quelconque), est fournie par la simulation Open_shop.S8. On présentera des alternatives à l'usage du groupe pour limiter le nombre d'items présents dans un ensemble de files d'attente et d'activités, à la page 119.

I-2 Durée maximale de séjour d'un item dans une file d'attente

La **durée maximale de séjour** d'un item dans le stock (Shelf Life:) est utilisable pour tenir compte de dates de péremption de produits périssables ou pour simuler le comportement d'un client qui, après une attente jugée trop longue quitte le système productif. Simul8 propose une durée maximale identique pour tous les items.

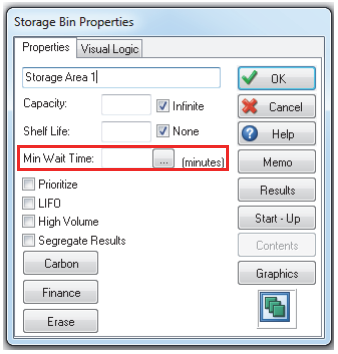


Le retrait de la file d'attente des items ayant atteint cette durée maximale ne peut être déclenché par la file d'attente, celle-ci ayant un comportement passif. Il faut faire appel à une activité fictive (et donc avec un temps opératoire nul): l'usage du bouton Routing In de la fenêtre des propriétés de cette *activité* fictive ouvre une fenêtre permettant de définir la règle de sélection (Discipline) parmi une liste de règles qu'il faut augmenter (usage du bouton More>>) pour pouvoir choisir l'option Expired Only (voir exemple Peremption_fixe.S8). Cette procédure est utilisée dans l'exemple du cabinet médical dans lequel on limite à 2 minutes l'attente des appels téléphoniques ne pouvant être pris immédiatement (voir l'activité «Appels perdus»). Pour chaque item entrant dans la file d'attente, il est possible de définir (via *Visual Logic*) une durée différente de séjour maximal dans cette file, ce qui est utile, par exemple, pour simuler de manière réaliste le fonctionnement de centres d'appel (voir exemple Peremption_variable.S8). On reviendra sur ce point à la page 113.



I-3 Temporisation minimale en file d'attente

Une temporisation minimale peut être requise dans une file d'attente c'est-à-dire le temps de séjour minimal d'un item dans cette file. Ce temps de séjour minimal peut être fixe ou prendre la valeur d'un *label* de l'item définie en appuyant sur le bouton ... (le *label* «L Attente» dans l'exemple Attente_mini_dans_stock.S8) ou une occurrence d'une variable aléatoire.



I-4 Classement des items en sortie

L'item prélevé par une activité est le premier d'une liste définie à partir de l'une des 3 règles décrites ci-dessous. Si le prélèvement de l'item est soumis à une condition (Batch by type , voir pages 33 et 75), c'est le premier de la liste respectant cette condition qui est choisie. Si une activité a besoin d'un ou plusieurs items pouvant se trouver dans plusieurs files d'attente auxquels il est relié, les règles décrites ci-dessous se combinent avec celles décrites à la page 72.

- La règle par défaut est celle du **Premier Entré - Premier Sorti**, utilisée dès que les options Prioritize et LIFO ne sont pas cochées.
- La règle du **Dernier Entré - Premier Sorti** (LIFO) est illustrée par l'exemple LIFO.S8.

- **Classement selon** la valeur décroissante prise par un **label** (Prioritize) choisi dans une liste qui s'affiche après avoir coché la boîte à cocher ; dans ce cas, un item rentrant dans la file d'attente, au lieu de prendre la dernière place, prend la dernière place du groupe ayant sa priorité (FIFO dans le groupe), ces groupes étant classés par priorités décroissantes. Le premier modèle de l'exemple Priority_Items.S8 illustre cette possibilité qui a été utilisée pour favoriser dans la salle d'attente, le client venant de se faire faire une radio (voir détail à la page 33). L'exemple Routing_in_oldest.S8 illustre l'utilisation de cette règle de définition de l'ordre de la liste de sortie quand elle est utilisée avec le même *label* dans plusieurs files d'attente connectées à l'entrée d'une même activité. Si l'on veut utiliser simultanément deux critères, il convient d'utiliser un troisième critère dont la valeur est calculée à partir des valeurs prises par les deux premiers critères selon une formule traduisant les règles de pondération accordées à ces deux critères.

I-5 Analyse du comportement d'une file d'attente pendant la simulation

L'analyse du comportement d'une file d'attente pendant la simulation s'effectue en ouvrant (Results) la fenêtre de résultats à partir de la fenêtre du stock.

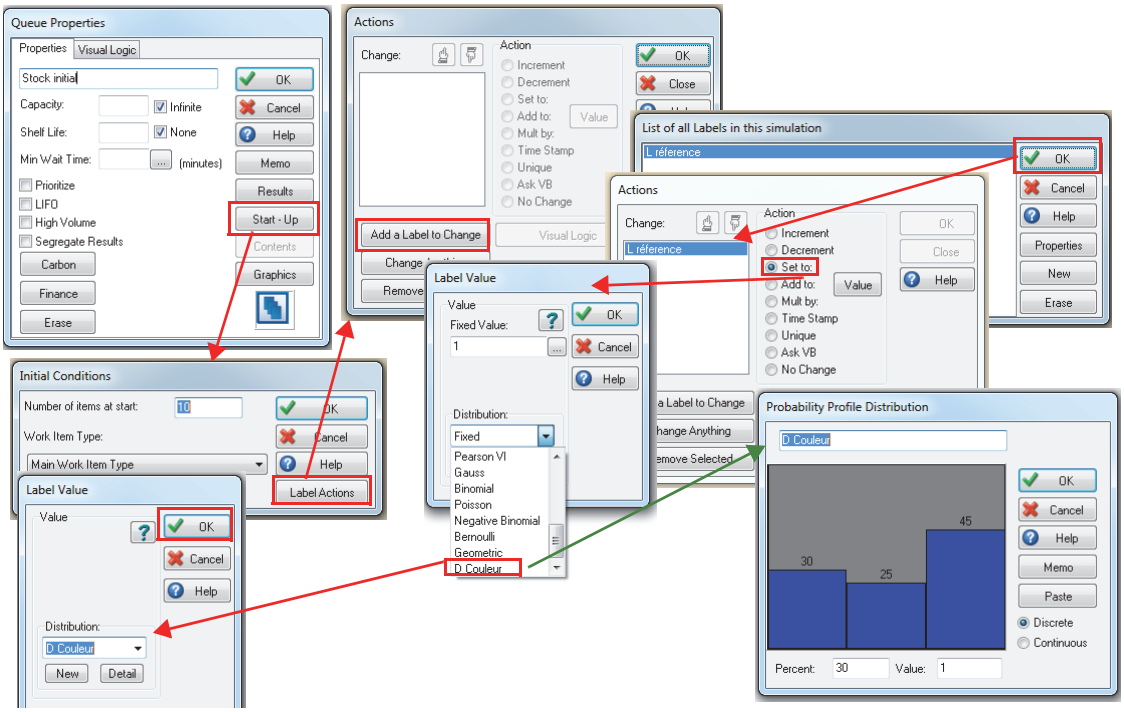
On obtient tout d'abord un ensemble d'**indicateurs** (KPIs) relatifs à l'**occupation instantanée de la file d'attente** avec une analyse du nombre courant d'items en stock durant la simulation, et la possibilité d'ouvrir un graphique permettant, pendant la simulation, de suivre l'évolution de la file d'attente. On a la possibilité de retrouver ce suivi avec les graphiques ouverts sous Results Manager, avec l'avantage d'une analyse du comportement simultané des files d'attente, activités et ressources auxquels on s'intéresse.

On obtient ensuite un ensemble d'**indicateurs** relatifs à la **durée de présence des items** dans la file d'attente et un histogramme de la distribution observée durant la simulation. On peut pousser l'analyse en s'intéressant à des sous-ensembles des items. On définit alors une partition des items entrants, basée sur les modalités prises par un attribut qualitatif, on utilise Segregate Results qui déclenche l'ouverture d'une fenêtre de sélection du *label* définissant la partition. Cette possibilité est identique à celle offerte aux points de sortie (voir la simulation du cabinet médical, page 51).

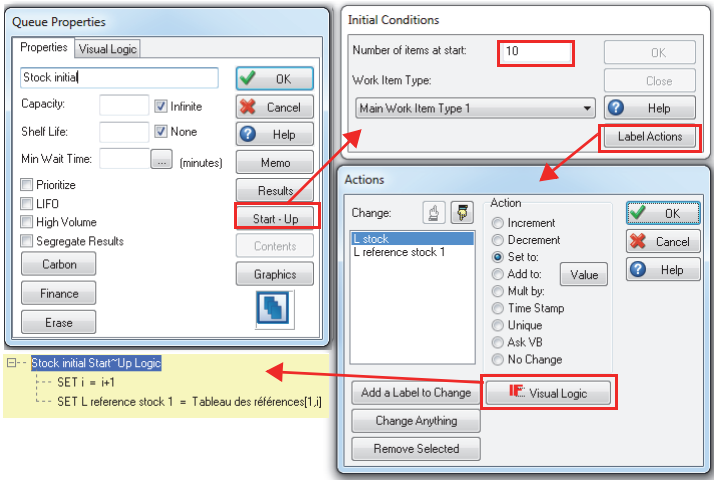
II INITIALISATION POSSIBLE D'UNE FILE D'ATTENTE

On peut commencer une simulation avec des files d'attente contenant des items de caractéristiques connues (ce qui n'a de sens que si la file d'attente correspond à un stock). Cette initialisation permet de remplir la file d'attente d'un certain nombre d'items (via le bouton Start - Up de la fenêtre de définition d'une file d'attente) dont on détermine les caractéristiques comme on le fait à un point d'entrée (via le bouton Label Actions). Deux définitions d'un *label* sont possibles.

- Cette définition peut être stochastique, les valeurs de *labels* d'un item étant générées aléatoirement. L'exemple Initialisation_Stock_stochastique.S8, illustré ci-dessous, traite d'une initialisation des items mis dans une file d'attente, possédant des caractéristiques générées aléatoirement à partir de distributions de probabilités connues.



- Cette définition peut être déterministe, les valeurs de *labels* d'un item étant prédéterminées. L'exemple Initialisation_Stock_deterministe.S8 illustre le cas d'une initialisation d'une file d'attente de caractéristiques lues dans un fichier de données (dans *Information Store* ou un fichier Excel ; exemple qu'il est préférable d'étudier qu'après avoir vu comment exploiter ce type d'information, page 101).



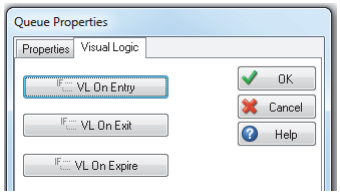
Une file d'attente ainsi initialisée peut remplacer un point d'entrée. Cette initialisation des files d'attente est incontournable si ces items restent présents tout au long de la simulation (comme c'est le cas des kanbans dans une simulation de Juste-A-Temps, par exemple). Elle peut aussi être utilisée, dans certains cas, pour éviter de paramétrer un préchauffage dans la simulation (*warm up,* voir page 48). Bien évidemment, l'initialisation d'une file d'attente n'exclut pas que celle-ci puisse être alimentée au cours de la simulation.

L'exemple Arrivees_stochastiques_clients_connus.S8 est une variante du précédent, dans lequel l'événement déclenchant le prélèvement d'un item dans une file d'attente est conditionné par l'analyse d'un item arrivant dans un autre stock. Ce cas de figure se produit, par exemple, pour une file d'attente correspondant à un ensemble de clients parfaitement identifiés (fichier d'abonnés d'une médiathèque, par exemple) et que l'on cherche à simuler des arrivées aléatoires de ces clients (arrivées dans la médiathèque), avec une personnalisation du client (tirage aléatoire de l'identité de l'abonné entrant dans le sous-ensemble du fichier

des abonnés susceptibles de venir, ce qui élimine, a minima, les abonnés déjà présents dans la médiathèque).

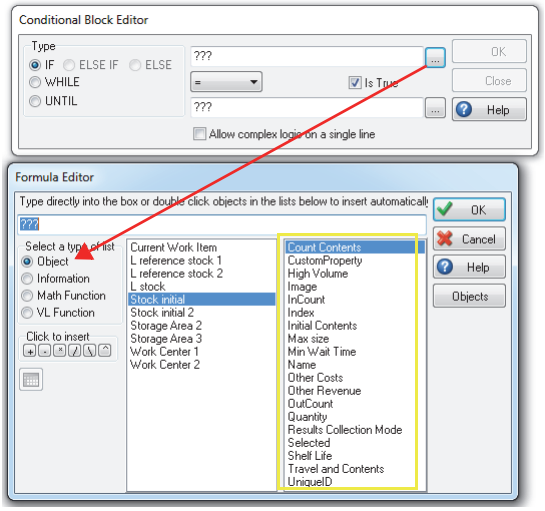
III FILE D'ATTENTE ET *VISUAL LOGIC*

III-1 Possibilités d'exécution de programmes en *Visual Logic* lors d'un mouvement en file d'attente



Il est possible d'exécuter un programme en *Visual logic* (voir ci-contre) lorsqu'un item arrive dans une file d'attente (VL On Entry) ou en sort (VL On Exit) ou lorsqu'il est appelé par une activité fictive chargée de retirer du stock les items dont le temps de séjour a atteint la durée maximale (fixe) de séjour (VL On Expire). Les deux premières actions possibles sont intéressantes pour mettre à jour un inventaire permanent décrit dans un tableau de l'*Information Store* (ce qu'illustre l'exemple Stock_&_Information_Store.S8) ou pour réviser les valeurs du *label* utilisé par l'option Prioritize qui définit la liste de sortie sur une base différente de celle de l'ordre d'arrivée (voir page 65).

III-2 Propriétés d'une file d'attente mobilisables par *Visual Logic*

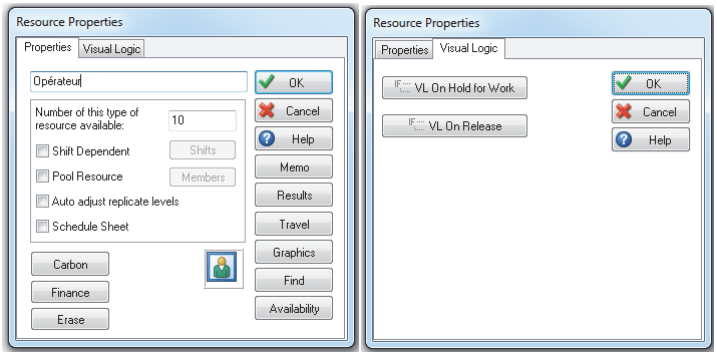


Les instructions de *Visual Logic* peuvent utiliser des propriétés de la classe «file d'attente». Il s'agit d'un ensemble de caractéristiques prédéterminées (voir liste encadrée en jaune, ci-contre), associées à n'importe quelle file d'attente et jouant le même rôle que les *labels*. Ces caractéristiques ont des valeurs par défaut utilisées lors de la création d'une file d'attente pendant la modélisation d'un processus. Certaines de ces caractéristiques sont modifiables, comme la capacité maximale d'une file d'attente, et d'autres pas, comme le dénombrement du nombre d'items dans la file d'attente au moment de l'exécution de l'instruction de *Visual Logic* utilisant cette caractéristique. Pour obtenir le résultat de l'écran suivant, il suffit de double-cliquer sur le nom de la file d'attente, puis de double-cliquer sur la caractéristique à utiliser.

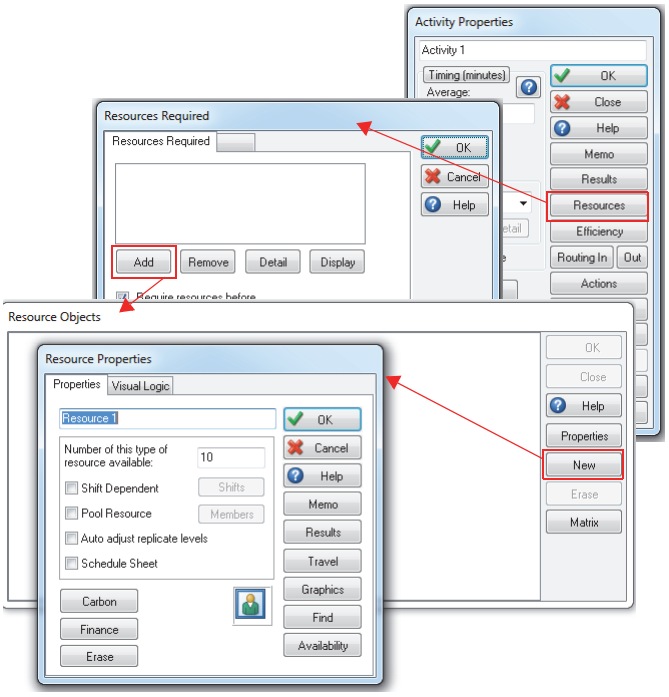
La caractéristique la plus souvent utilisée est Count Contents qui contient le nombre d'items en file d'attente au moment de l'exécution de l'instruction (ne pas confondre avec InCount qui vaut 1 si la file d'attente a été utilisée dans la simulation). Parmi les exemples d'utilisation de la propriété Count Contents, voir Choix_Production_stock_mini.S8, Synchronisation_stock_produits_Multiples.S8 (voir page 116) ou Arrivees_stochastiques_clients_connus.S8, dans lequel cette propriété est utilisée pour définir la borne supérieure d'une distribution uniforme, utilisée pour tirer de manière équiprobable un item d'une file d'attente dont le contenu évolue au cours de la simulation).

Chapitre V
Caractérisation des ressources

I Caractérisation des ressources



Une **ressource** est un facteur productif (opérateur, outillage…) indispensable à l'exécution d'une opération sur un item dans l'activité. L'usage des ressources a été introduit dans l'exemple de la page 53, avec la seconde version du cabinet médical. Nous avons souligné alors que la **création** d'une ressource ne s'impose que si celle-ci est partagée par plusieurs activités. Dans l'exemple du cabinet médical, l'activité «Secrétariat» comporte un bureau, un fauteuil, un téléphone, un micro-ordinateur et une secrétaire; aucun de ces éléments n'étant partagés, l'activité comporte tous ces éléments et est réputée opérationnelle sur toute l'amplitude de la journée (sauf mobilisation des plages horaires, voir page 53). La fenêtre des propriétés d'une ressource est fournie ci-dessus.



La **création** d'une ressource est normalement antérieure à son utilisation (usage du bouton Resource). Elle peut également se faire, comme illustré ci-contre, lors de son affectation à une activité, en utilisant le bouton Add de la fenêtre ouverte par le bouton Resources, ce qui provoque l'affichage de la liste des ressources existantes, qui peut être augmentée en utilisant le bouton Add de cette dernière fenêtre.

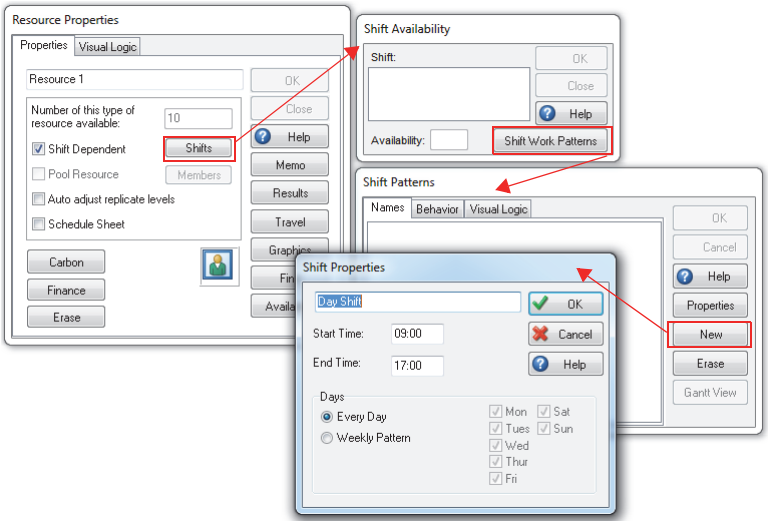
La **représentation** de la ressource se fait par une icône, l'icône par défaut pouvant être modifiée dans la fenêtre ouverte avec le bouton Graphics.

Il est possible de définir des **temps de transport** d'une ressource pour chaque couple origine-destination créé dans la modélisation du processus, dans la fenêtre ouverte avec le bouton Travel.

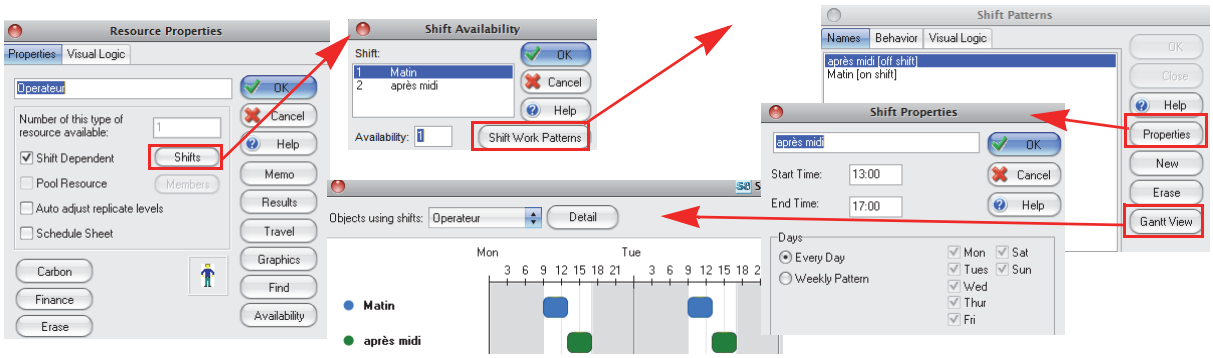
La **disponibilité** d'une ressource se définit dans la fenêtre des propriétés d'une

ressource; elle peut être fixe ou variable au cours de la journée. Dans le cas d'une disponibilité **fixe** (l'option Shift Dependent n'étant pas cochée), un effectif de 10 est alors proposé par défaut: Number of this type of resource available: 10), valeur bien évidemment modifiable. Cette disponibilité peut être **variable** dans la journée Shift Dependent.

Dans ce dernier cas, sa définition se passe par l'usage des Équipes (***shifts***). Une équipe se caractérise par sa présence sur une ou plusieurs plages de temps (***Shift Work Pattern***). Ces plages sont créées comme expliqué ci-contre.

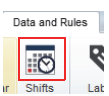


L'exemple Resources_Shift_1.S8 illustre l'utilisation d'équipes avec 2 centres de production mobilisant chacun un opérateur, sachant que le nombre d'opérateurs disponibles est de 1 le matin et de 2 l'après-midi.

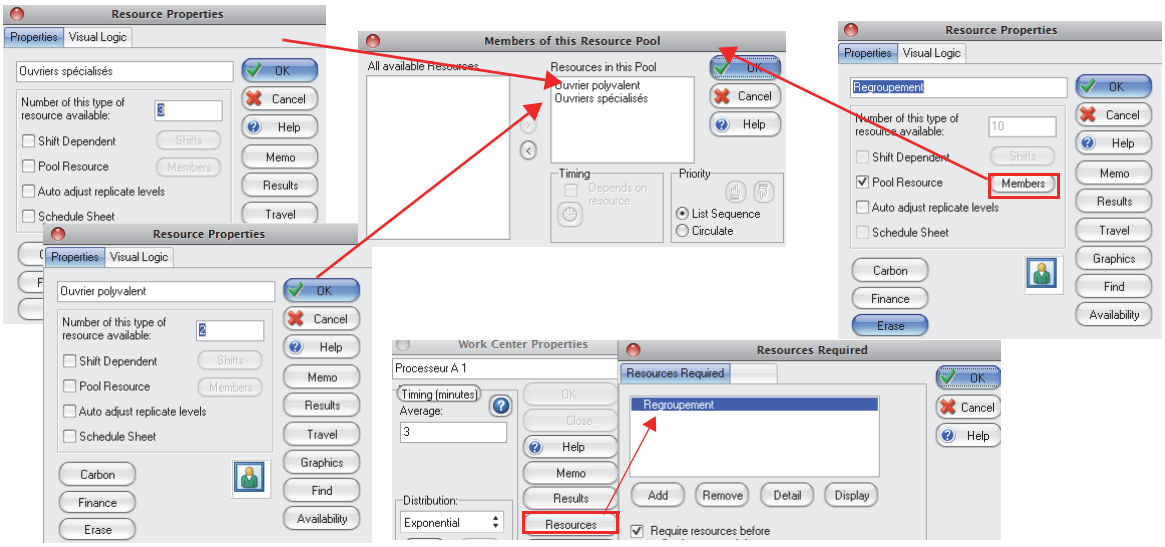


L'exécution d'une opération dans une activité, peut mobiliser une (ou plusieurs) unité(s) appartenant à des **ressources substituables** pour cette opération. Par exemple, après avoir défini une ressource «ouvrier qualifié» comportant 3 personnes et une ressource «ouvrier spécialisé» comportant 2 personnes, on peut avoir besoin de définir une ressource «Regroupement» réunissant les deux ensembles précédents, si l'opération d'un processeur A peut s'exécuter aussi bien par un ouvrier qualifié, qu'un ouvrier spécialisé, tandis que celle d'un processeur B ne peut s'exécuter qu'avec un ouvrier spécialisé, ce qu'illustre l'exemple Pool.S8. L'exemple Ressource_Polylavence_pool décrit une situation plus complexe de compétences multiples que peuvent posséder des opérateurs, ces opérateurs pouvant avoir des ensembles différents de compétences, et dans laquelle chaque tâche à exécuter nécessite l'une de ces compétences. Cette modélisation est importante pour pouvoir apprécier les gains liés à une amélioration de la **polyvalence** du personnel.

L'analyse des caractéristiques des équipes et leur modification peuvent aussi se faire en utilisant le bouton Shifts de l'onglet Data Rules du menu principal..



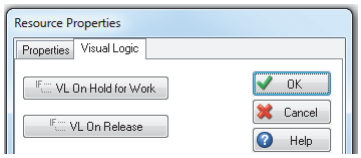
On reviendra à la page 91 comment les ressources sont mobilisées par une activité.



II RESSOURCE ET *VISUAL LOGIC*

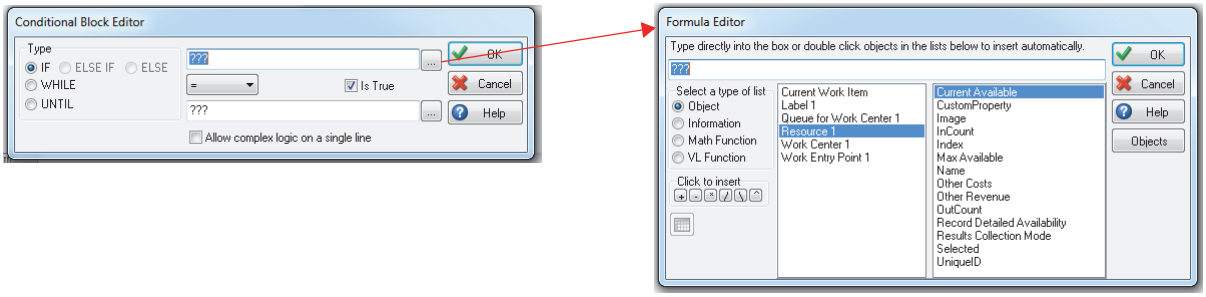
II-1 Possibilités d'exécution de programmes en *Visual Logic* lors de l'utilisation d'une ressource

Un programme en ***Visual Logic*** défini dans la fenêtre ouverte par l'onglet Visual Logic peut être exécuté lors de la mobilisation d'une ressource (VL On Hold for Work) ou de sa libération (VL On Release).



II-2 Propriétés d'une ressource mobilisables par *Visual Logic*

Les instructions de *Visual Logic* peuvent utiliser des propriétés de la classe «Ressource». Il s'agit d'un ensemble de caractéristiques prédéterminées, associées à n'importe quelle ressource et jouant le même rôle que les *labels*. Ces caractéristiques ont des valeurs par défaut utilisées lors de la création d'une ressource pendant la modélisation d'un processus. Elles sont utilisables dans les programmes écrits en *Visual Logic*. La liste de ces caractéristiques est fournie dans la fenêtre suivante.



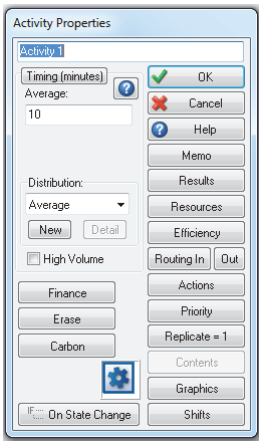
Un exemple intéressant d'utilisation de la propriété *Max Available* d'une ressource peut être trouvé avec le modèle Work_Center_Shifts_Stock_in_individuel_1.S8.

Chapitre VI

CARACTÉRISATION DES ACTIVITÉS (*POSTES DE TRAVAIL*)

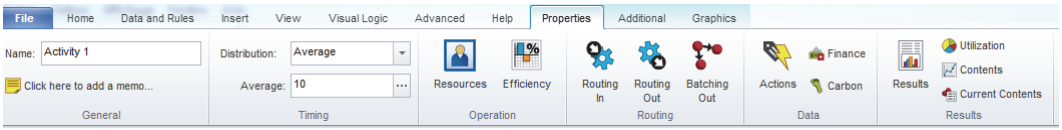
L'activité est terme générique permettant de désigner une machine ou un poste de travail utilisé pour effectuer un traitement sur un item. Dans les services, la spécialisation d'un lieu pour effectuer un traitement est loin d'être générale comme c'est le cas dans la production de biens, un même lieu jouant successivement des rôles de stocks (où aucun traitement n'est possible) ou de poste de travail. Dans ce contexte, l'appellation activité est sans doute préférable à celle de poste de travail et celle de file d'attente peut être plus intelligible que celle de stock. L'équipe de Simul8 après avoir longtemps utilisé les termes de ***storage bins*** et de ***work centers***, utilise depuis 2013 les termes de files d'attente et d'activités.

Les activités sont créés avec l'outil Activity. La **fenêtre de saisie** des propriétés d'une activité est la suivante. On examinera successivement les différentes composantes du paramétrage d'une activité, définies en appuyant sur les différents boutons.



- Alimentation de l'activité par le bouton Routing In, § I, page 72.
- Sortie de l'activité par le bouton Out, § II, page 87.
- Temps d'exécution du travail à effectuer par l'activité dans le cadre *Distribution* de la fenêtre ci-dessus, § III, page 91.
- Ressources mobilisées pour exécuter le travail à effectuer par l'activité par la partie bouton Resources, § IV, page 91.
- On regroupera dans un paragraphe (§ V, page 93) quelques spécifications complémentaires définies par les boutons Label Actions, Replicate = 1, Priority et Efficiency. On examinera également quelques points liés à l'utilisation de *Visual Logic*.
- On examinera enfin la caractérisation d'un convoyeur Conveyor (que l'on peut créer en ouvrant la partie Advanced Building Blocks de la barre latérale gauche de construction du modèle), qui est une activité particulière ayant des caractéristiques spécifiques (§ VI, page 95).

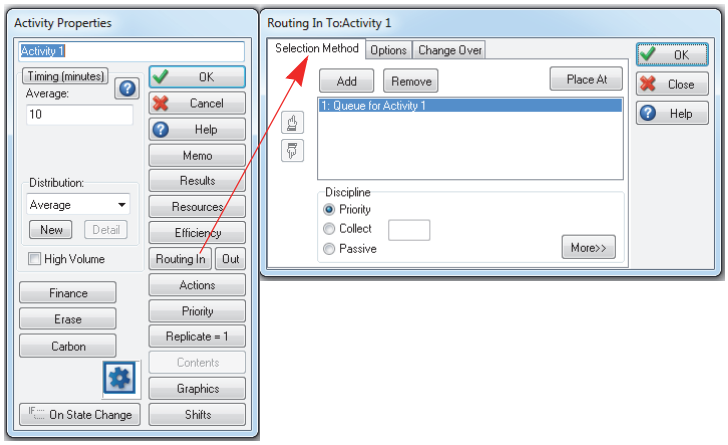
Ajoutons pour terminer qu'un simple clic sur l'icône d'une activité fait apparaître le bandeau supérieur suivant dans lequel on retrouve les options offertes dans la fenêtre ouverte par un double-clic sur l'icône de l'activité.



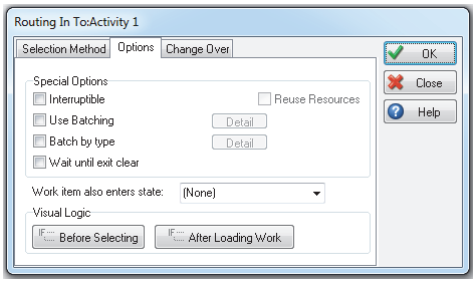
I Alimentation de l'activité: bouton *Routing in*

Les spécifications de l'alimentation sont définies par les fenêtres ouvertes après avoir appuyé sur le bouton Routing In, qui donne accès à trois onglets.

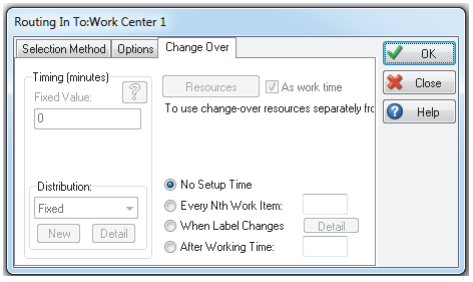
- L'onglet Selection Method permet de définir la règle de prélèvement du ou des items requis par l'opération exécutée par l'activité. Certaines de ces règles n'ont de sens que si plusieurs items sont requis et/ou plusieurs files d'attente existent en entrée de l'activité.



- L'onglet Options permet tout d'abord de définir des actions à exécuter (programmes en *Visual logic*) soit avant tout prélèvement (Before Selecting), ce qui permet en particulier de modifier certains paramètres de l'activité, soit immédiatement après le prélèvement mais avant que ne commence le traitement (After Loading Work); on reviendra sur ce point au § I-1, page 74. Il permet également de définir des conditions sur la valeur que doit posséder un *label* d'un item à prélever (Batch by type) ou une fourchette de la taille du lot d'items interchangeables à prélever (Use Label Batching). Ces deux options, exclusives l'une de l'autre, seront analysées au § I-1, page 74. Deux autres options sont offertes mais ne seront pas analysées dans ce document: la préemption (Interruptible), c'est-à-dire la possibilité d'interrompre un traitement en cours pour effectuer un traitement prioritaire, avant de reprendre le traitement en cours et la possibilité de bloquer les prélèvements d'items (Wait until exit clear) si le chemin de sortie n'est pas libre.



- L'onglet Change Over permet d'ajouter au temps de traitement de l'opération à exécuter, un temps de réglage des temps complémentaires à ajouter tous les ***n*** items traités (Every Nth Work Item:) ou en cas de changement de valeur d'un *label* (When Label Changes) ou après traitement (After Working Time:). Nous reviendrons sur ces possibilités à la section III, page 91.



La définition des méthodes de sélection et l'utilisation des options ne se combinent pas librement. Interviennent de surcroît certaines caractéristiques du problème de sélection à l'entrée: quand l'activité se libère, il va chercher un (ou plusieurs) item(s) homogènes ou hétérogènes dans un (ou plusieurs) files d'attente[1]. L'analyse de ces entrées oblige à distinguer plusieurs cas obtenus en combinant trois caractérisations de base.

- Pour le prélèvement par cette activité, les items contenus dans la file d'attente ou les files d'attente sont considérés comme étant de **même nature** et donc interchangeables, les différences de valeurs prises par leurs *labels* étant sans objet pour le traitement à effectuer et on parlera alors de **files d'attente homogènes**. Ces items peuvent être considérés comme **potentiellement différents** (et donc non-interchangeables), les items à prélever dans la file d'attente devant avoir une valeur précise pour certains *labels* et on parlera alors de **files d'attente hétérogènes**. Il convient de bien comprendre que cette homogénéité ou hétérogénéité des items d'une file d'attente dépend de l'activité qui y prélève des items, car c'est cette activité qui considère ou non les items d'une file d'attente comme interchangeables.
- L'opération à réaliser par l'activité nécessite le prélèvement d'un ou plusieurs items.
- L'activité est reliée à une ou plusieurs files d'attente en entrée. Plusieurs cas doivent être distingués, dans la modélisation de base proposée par Simul8.
  - 1 file d'attente - 1 item à prélever. Si l'activité prélève **un seul item** dans une seule file d'attente, cette file d'attente peut être aussi bien homogène qu'hétérogène ;
  - *m* file d'attente - 1 item à prélever. Il en est de même si le prélèvement de cet item unique peut se faire dans plusieurs files d'attente, le problème étant de déterminer dans quelle file d'attente l'item sera puisé ; ce problème de sélection relève d'une logique de type **OU**.
  - 1 File d'attente - *n* items à prélever. Lorsque *n* items (*n* étant prédéterminé) sont à prélever dans une seule file d'attente, celle-ci est nécessairement homogène.
  - *m* Files d'attente - 1 item à prélever. Lorsque plusieurs items sont à prélever dans plusieurs files d'attente, le nombre n_i d'items à prélever dans chaque file d'attente *i* étant prédéterminé avant que l'activité ne commence à prélever ces items, chaque file d'attente est homogène mais se différencie des autres files d'attente par la valeur prise par un même label. Dans ce cas, le problème de sélection relève d'une logique de type **ET** et se caractérise toujours par l'usage de l'option ***Collect*** dans le *Routing in*. Ce cas correspond soit à une opération de production conduisant à la création d'un nouvel item (assemblage), généralement sur la base d'une **nomenclature fixe**, soit à une opération de distribution conduisant au regroupement d'items en un lot (*picking*) en utilisant la **nomenclature variable** implicitement associée à une commande (voir page 80).
  - Un cas particulier de la modélisation sous Simul8 se caractérise par le prélèvement d'un nombre variable d'items dans une file d'attente homogène ou dans plusieurs files d'attente homogènes rigoureusement interchangeables du point de vue de l'opération réalisée par l'activité.

Cette combinatoire conduit au tableau suivant qui fait référence à des tableaux de synthèse commentant ce qu'il est possible de faire dans chaque cas.

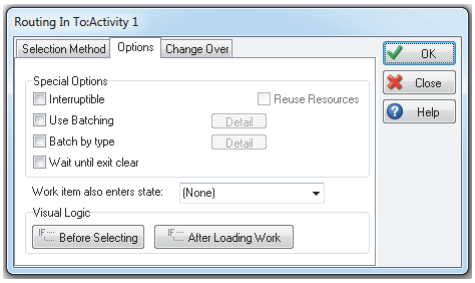
1. Exceptionnellement (voir page 88), il peut recevoir des items en provenance d'un autre activité qui attend passivement que l'on vienne chercher l'item qu'il vient de traiter

Tableau 3 : Typologie des cas de figure rencontrés en *Routing in*

Prélèvement de		File d'attente unique	Files d'attente multiples	
n items	dans une file d'attente		OU	ET (*Collect*)
$n = 1$	homogène ou hétérogène	Voir tableau 4, p. 81	Voir tableau 5, p. 83	2 files d'attente *hétérogènes* et $n_1 = 1$ *Match* sur valeur d'un label Voir tableau 6, p. 85
$n > 1$ fixe	homogène	Voir tableau 4, p. 81	-	Chaque file d'attente est homogène. Prélèvement de n_i unité dans la file d'attente i Voir tableau 6, p. 85
	hétérogène	-	-	-
$n_1 \leq n \leq n_2$	homogène	Voir tableau 4, p. 81	Voir tableau 5, p. 83	-
	hétérogène	-	-	-

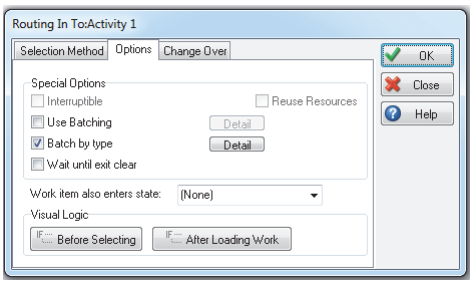
Nous allons détailler les possibilités offertes par chacun des onglets mais avant il faut ajouter que le prélèvement d'un item dans le (ou les) file(s) d'attente - amont est normalement possible dès que l'activité se libère et qu'il puisse trouver un item respectant les conditions de prélèvement requises. Le prélèvement peut éventuellement être bloqué par l'intermédiaire d'un programme en Visual Logic (voir § VII-1, page 96) exécuté avant tout prélèvement (Before Selecting)

I-1 Possibilités offertes par l'onglet «Options»



L'onglet Options sert à introduire une condition de valeur d'un *label* que doit respecter l'item entrant (§ I-1.1) ou à spécifier le nombre d'items de même nature à prélever si ce nombre est supérieur à 1 (§ I-1.2) ou à introduire une procédure (programme en *Visual Logic*) à exécuter avant tout prélèvement (*Before selecting,* Before Selecting) ou juste après prélèvement mais avant toute exécution du travail par l'activité (*After Loading Work,* After Loading Work). L'utilisation de ces possibilités de programmation sera étudiée au § VII-1, page 96.

I-1.1 Prélèvement d'item(s) ayant une valeur prédéterminée d'un *label* donné (*Batch by type*)



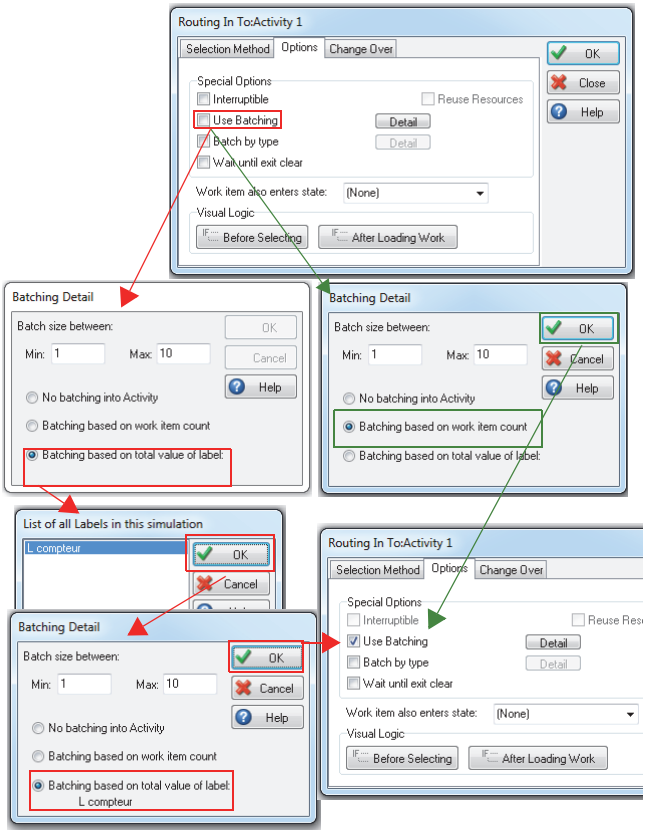
En cochant l'option *Batch by type*, on introduit une condition à respecter par l'item, nécessairement **unique**, prélevé par l'activité. L'utilisation de cette option est incompatible avec l'option *Use Label Batching* (voir ci-

après). Cette option est utilisable que l'activité soit reliée à une ou plusieurs files d'attente en entrée. Elle implique que ces files d'attente soient **hétérogènes**.

L'utilisation de cette option a été illustrée pour le prélèvement d'un seul item avec l'exemple du cabinet médical (en imposant au patient de se rendre au cabinet du docteur avec lequel il avait rendez-vous, voir page 33). Le ***cas* 1** étudié dans le tableau 4 de la page 81, explicite les méthodes de sélection (voir § I-2, page 77) compatibles avec cette sélection, ainsi que la possibilité de ne pas sélectionner suivant la logique du Premier-Entré - Premier-Sorti; il illustre chaque combinaison possible par une simulation que vous pouvez directement ouvrir.

Si l'activité est reliée à plusieurs files d'attente, le prélèvement est implicitement de type **OU**, c'est-à-dire que l'unité prélevée est cherchée dans l'une de ces files d'attente, l'exploration des files d'attente se faisant dans l'ordre défini par la méthode de sélection, toutes les méthodes n'étant pas compatibles avec cette option *Batch by Type*. Le ***cas* 4** étudié dans le tableau 5 de la page 83, explicite les méthodes de sélection compatibles avec cette sélection, ainsi que la possibilité de ne pas sélectionner suivant la logique du Premier-Entré - Premier-Sorti; il illustre chaque combinaison possible par une simulation que vous pouvez directement ouvrir.

I-1.2 Prélèvement d'un nombre variable d'items (*Use Label Batching*)



En cochant l'option *Use Label Batching*, l'activité prélève un nombre ***n*** d'items compris entre deux bornes n_1 et n_2, dans **une file d'attente *homogène*** (ou dans plusieurs files d'attente homogènes, comme on le verra ensuite). Deux possibilités sont offertes. On peut vouloir faire porter la contrainte sur le nombre d'items pouvant accéder simultanément à l'activité (Batching based on work item count), ce qu'illustre le cheminement en vert de la figure ci-contre. On peut aussi faire porter cette contrainte sur une somme pondérée du nombre d'items par un coefficient de pondération consigné dans un label possédé par les items (Batching based on total value of label), ce qu'illustre le cheminement en rouge de la figure ci-contre. Prenons l'exemple simplifié d'un boulanger fabriquant des croissants au beurre, à partir d'un stock (file d'attente) de croissants à cuire alimenté par une activité (ou un point d'entrée). Supposons que la cuisson porte sur un lot compris entre 30 (on peut lancer une

opération de cuisson si on dispose d'au moins 30 croissants) et 50 croissants (on ne peut traiter simultanément plus de 50 croissants).

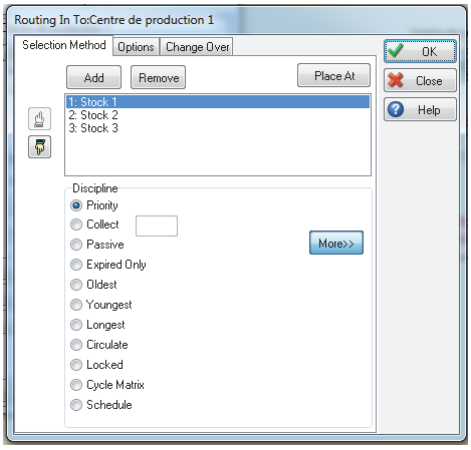
- L'exemple Routing_in_Use_Label_Batching_0.S8 illustre l'utilisation de la contrainte de lot portant sur le simple dénombrement du nombre d'items (Batching based on work item count).
- L'exemple Routing_in_Use_Label_Batching_1.S8 illustre l'utilisation de la contrainte de lot portant sur la somme pondérée des items du lot (Batching based on total value of label); on procède comme suit: après avoir coché l'option Use Label Batching une fenêtre permet de sélectionner le label numérique à utiliser dans le calcul; ici, c'est le label «L compteur» qui est utilisé et qui est affecté à tous les items entrant, avec la valeur 1. Ces deux exemples illustrent le ***cas* 2** étudié dans le tableau 4 de la page 81
- L'exemple Routing_in_Use_Label_Batching_2.S8 généralise l'utilisation de la contrainte de lot portant sur la somme pondérée des items du lot, en prélevant les items dans plusieurs files d'attente et en caractérisant ces items par des coefficients de pondération différents. Il illustre le ***cas* 5** étudié dans le tableau 5 de la page 83. Dans cet exemple, on peut mélanger des croissants normaux avec des mini-croissants qui prennent deux fois moins de place, la contrainte de capacité étant exprimée en nombre de croissants normaux. On peut ajouter que ces deux stocks peuvent être fusionnés en un seul sans problème. D'un certain point de vue, la file d'attente peut être alors considérée comme hétérogène mais cette hétérogénéité est de type volumétrique ou massique et utilisée uniquement par des activités où le critère volumétrique ou massique pour constituer un lot à traiter est à prendre en compte. Le problème traité ici se pose souvent dans la modélisation de transport. Par exemple si l'on s'intéresse au chargement d'un camion de capacité 20 m^2, les items à charger (cartons, caisses…) peuvent avoir des volumes unitaires différents ($L\ compteur_i$). Il importe alors que le volume embarqué ne dépasse pas la capacité et que le camion soit suffisamment chargé. Dans ces conditions, la contrainte devient $n_1 \leq \sum_i L\ compteur_i \leq n_2$.

Deux remarques complémentaires peuvent être faites:

- La taille du lot est variable, elle dépend du nombre d'items en file d'attente au moment où le traitement peut commencer; elle est fixe si les valeurs Min et Max coïncident.
- Les items ne sont pas fusionnés; ils conservent donc leurs caractéristiques initiales (sauf modification via Actions)

I-2 Possibilités offertes par l'onglet «*Selection Méthods*»

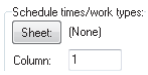
L'activité peut être reliée à une ou plusieurs files d'attente, certaines méthodes de prélèvement n'ont de sens que si l'activité est reliée à plusieurs files d'attente en entrée. Les méthodes décrites ici ne se préoccupent pas du caractère homogène ou hétérogène des files d'attente à l'exception de l'option *Collect* que l'on présentera en dernier. On verra à la page 111, comment traiter avec l'aide d'une activité fictive certains problèmes plus complexes d'alimentation d'une activité «réelle».



I-2.1 Premier jeu de méthodes de sélection en *Routing In*

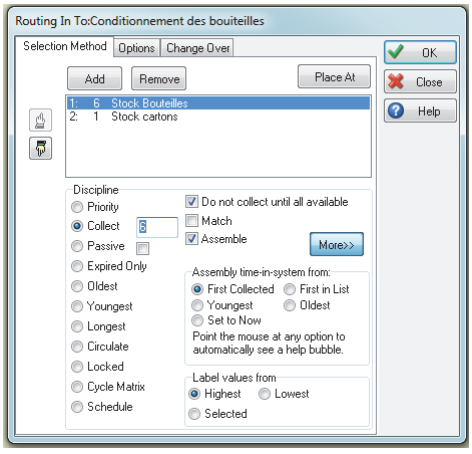
- L'option Priority indique que l'ordre dans lequel on doit examiner les files d'attente pour prélever un item est celui qui est affiché dans la fenêtre, si l'item ne peut être fourni par une file d'attente on passe à la suivante (si l'activité est reliée à plus d'une file d'attente en entrée). Cette option est utilisable dans tous les cas de figure et est illustrée (stock multiple) par l'exemple Routing_in_priority.S8 et l'exemple Routing_in_Priority_Label.S8.
- L'option Passive correspond au cas particulier d'une activité qui est alimentée directement par une autre activité au lieu de prélever lui-même les items dans une ou plusieurs files d'attente. Dans ce cas, l'activité situé en amont se libère à la double condition d'avoir terminé son opération et de pouvoir envoyer l'item dans l'*activité* passive, ce qui suppose qu'elle soit libre, sinon l'activité - amont est **bloquée**. Une séquence de deux activités peut être utile pour exécuter certaines opérations complexes difficiles à modéliser de manière simple avec une seule activité; dans ce cas, l'une des deux activités est normalement fictive (temps opératoire nul); si aucune des deux n'est fictive, cette option *Passive* est nécessaire pour la seconde activité. D'une manière générale, il est conseillé d'éviter deux activités se suivant sauf si l'une d'entre elles est fictive. Cette méthode exclut, bien évidemment, la possibilité de faire appel aux options de l'onglet Options, sauf celles de programmation. L'utilisation de cette méthode est illustrée avec l'exemple Routing_in_Passive.S8.
- L'option Oldest permet de prélever l'item qui se trouve depuis le plus longtemps soit dans l'une des files d'attente (Use Queue Time), soit dans le système productif (Use Queue Time); elle n'est utilisable que dans le cas de prélèvement d'un seul item, l'activité pouvant être connectée à plusieurs files d'attente en entrée. L'exemple Routing_in_oldest.S8 illustre cette méthode sans condition de valeur prise par un label, contrairement à l'exemple Routing_in_oldest_Label.S8.
- L'option Youngest permet de prélever l'item qui se trouve depuis le moins longtemps soit dans l'une des files d'attente (Use Queue Time), soit dans le système productif (Use Queue Time); voir Routing_in_youngest.S8 pour un usage simple de cette option et l'exemple Routing_in_youngest_Label.S8 qui combine cette option avec l'usage d'une valeur de *label* en sélection des items.

- L'option Longest permet de prélever l'item dans la file d'attente qui contient **le plus** d'items (voir l'exemple Routing_in_longest.S8 et l'exemple Routing_in_longest_Label.S8). Pour privilégier la référence la moins nombreuse dans une file d'attente hétérogène d'items possédant des références différentes (option *Shortest*, non offerte mais utile dans des problèmes d'approvisionnement) il faut exécuter une procédure (voir l'exemple Routing_in_label_Mini.S8).
- L'option Circulate permet de prélever l'item **successivement** dans chacune des files d'attente (voir l'exemple Routing_in_Circulate.S8 et l'exemple Routing_in_Circulate_Label.S8). Cette option est à privilégier en cas de choix du Use Batching dans l'onglet Options et si l'activité prélève des items dans plusieurs files d'attente : le choix de l'option Priority conduisant à ne passer à la seconde file d'attente que lorsque la première est vide, ce qui peut cependant correspondre à une nécessité de modélisation (gestion d'un approvisionnement selon la méthode des deux casiers, par exemple).
- L'option Ignore Starved (Ignore Starved) est cochée par défaut: si la file d'attente examinée est vide, on passe immédiatement à la suivante; ne pas cocher cette option conduit à bloquer le prélèvement pour «ce tour».
- L'option Locked : pour mémoire (car peu utilisé, uniquement via *Visual Logic*)
- L'option Cycle Matrix impliquant simultanément le *Routing in* et le *Routing Out* sera analysé à la page 114.
- L'option Schedule permet d'utiliser une feuille de classeur pour déterminer la file d'attente en entrée. Cette option permet de reproduire un ensemble de décisions observées.



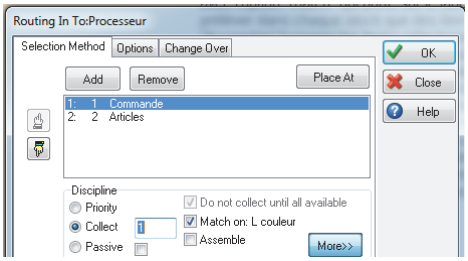
I-2.2 La méthode de sélection *Collect* en *Routing In*

La règle Collect permet de fabriquer une référence (opération d'assemblage ou de *picking*) à partir de **plusieurs items de nature différente** provenant de plusieurs files d'attente différentes, chacune d'entre elles étant homogène, les quantités étant prédéterminées avant tout prélèvement (**nomenclature fixe**).



Par exemple, pour confectionner un carton de 6 bouteilles, il faut prélever 6 bouteilles dans un premier stock (file d'attente) et 1 carton dans un second stock puis procéder à l'opération d'assemblage. Après avoir cliqué sur le bouton Routing In et choisit la règle Collect (Collect) ainsi que l'option spécifiant que les prélèvements dans les deux files d'attente ne peuvent se faire que si les quantités requises sont disponibles (Do not collect until all available); on sélectionne ensuite le premier stock dans la liste des files d'attente affichée (1: 1 Stock Bouteilles), on modifie le nombre d'items requis (Collect 6), ce qui donne à l'affichage (1: 6 Stock Bouteilles); pour le second stock, la valeur par défaut (1) n'a pas à être changée (2: 1 Stock cartons). En choisissant l'option Assemble, on indique qu'il n'y aura qu'un seul item en sortie (sinon, il y a conservation de tous les items entrants). Il faut ensuite indiquer que l'item résultant est un «carton de 6 bouteilles», ce qui se fait en ouvrant la fenêtre des

spécifications de sortie (bouton Out) puis en élargissant les options offertes (bouton More >>), ce qui permet de sélectionner l'item désiré (Exit Work Item Type: Carton de 6 bouteilles). L'item sortant est différent des items entrants si l'option *Assemble* a été cochée (voir l'exemple Routing_in_Collect_and_Assemble_1.S8). Cette méthode de prélèvement peut être utilisée pour constituer un lot à partir d'une seule file d'attente (voir l'exemple Routing_in_Use_Label_Batching_OR_Collect.S8).



La règle Collect permet également de prélever dans chacune des files d'attente pointées en entrée (habituellement deux), des items ayant tous une même valeur de label. Après avoir coché l'option Match, une fenêtre listant les labels existant s'ouvre ; on choisit le *label* à utiliser pour associer les items venant des différentes files d'attente, la valeur retenue pour ces prélèvements étant celle prise par le *label* du premier item de la file d'attente classée la première dans la liste affichée en haut de fenêtre. Là encore, on peut ou non décider de fusionner ces items en sortie (Assemble) et de prélever ou non plusieurs unités dans chacune des files d'attente (nombre saisi à droite de Collect). Cette utilisation de la règle *Collect*, qui correspond au **cas 7** du tableau 6 de la page 85, permet d'assurer la **synchronisation** entre un flux d'informations (commandes portant chacune sur un **article unique**) et un flux de matières hétérogènes (articles différents) tous gardés dans une **file d'attente unique**[1] et offre le moyen de simuler des interactions simples entre systèmes productifs et donc le fonctionnement de certaines **chaîne logistique** (une demande finale est satisfaite par un premier système productif, lequel doit s'approvisionner en composants auprès d'un autre système productif (voir l'exemple Routing_in_Collect_and_Match_1.S8, lequel utilise une activité fictive pour pouvoir choisir un client dans une liste de clients potentiels dynamiquement mise à jour). Le cas de commandes portant sur un article unique à prélever dans l'une des files d'attente d'un ensemble de files d'attente (**files d'attente multiples**) sera étudié à la page 116 (Synchronisation_stock_produits_Multiples.S8).

On peut adapter sans problème cette démarche en **gérant dynamiquement** le paramètre de la fonction *Collect*, pour prélever un nombre d'articles d'une référence donnée en une quantité définie par une commande (et donc susceptible de varier d'une commande à l'autre), chaque item étant gardé dans une file d'attente différente (voir l'exemple Routing_in_Collect_and_Assemble_2.S8). Il s'agit ici d'une forme de **nomenclature variable** dans laquelle les files d'attente ne sont pas banalisées et le contenu d'une commande est décrit par autant de *labels* qu'il y a d'items pouvant être commandés (tous ne l'étant pas nécessairement) ; ce cas de figure se trouve, par exemple dans les commandes passées dans les *fastfoods*. Si les files d'attente sont banalisées (partiellement ou en totalité) et si la variété des références pouvant être commandée est forte, on a intérêt à définir une commande comme un ensemble de lignes de bons de commande à traiter séquentiellement et non de manière groupée (voir page 116, l'exemple Synchronisation_stock_produits_Multiples.S8).

1. Si ces articles différents se trouvent dans des files d'attente différentes, le processus de synchronisation est un peu plus complexe (voir page 116).

I-3 Récapitulatif

Tableau 4 : *Routing in* file d'attente unique en entrée

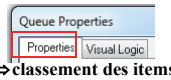
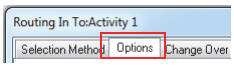
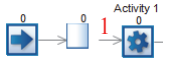
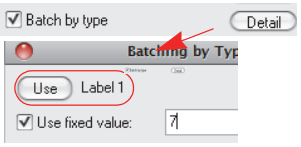
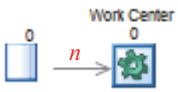
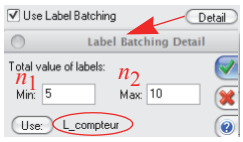
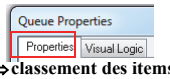
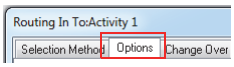
Configurations	Conditions de prélèvement définies dans		
	la file d'attente  **(⇒classement des items dans la liste de sortie)**	l'activité Routing In 	 ***(seules les méthodes pertinentes sont retenues)***
Cas 1 File d'attente unique en entrée et prélèvement de ***un seul*** item 	1. **Classement par défaut**: classement des items dans leur ordre d'arrivée dans la file d'attente Prioritize 2. **Modification de classement** sur *Label* via l'option *Prioritize* de la file d'attente ⇒ classement des items par ordre décroissant de valeurs du *label* choisi, avec partage des ex aequo sur le critère de leur ordre d'arrivée dans lea file d'attente. ***Exemple***: classement des items par valeur ↓ du *label* *L_Priorité*: Prioritize (L_Priorité) Reverse Priority (Low=first)	1. **Pas de condition** Batch by type 2. **Condition** sur *Label* via l'option *Batch by type* Exemple: on ne tire de la file d'attente que les items possédant la valeur 7 pour le *label Label 1* 	1. Priority ⇒ choix du premier item de la liste de sortie de la file d'attente respectant la condition de l'option. Voir Routing_in_priority.S8 (pas de condition) et Routing_in_Priority_Label.S8 (condition sur label) 2. More>> Oldest Use Queue Time ⇒ choix du premier item arrivé dans la file d'attente et respectant la condition de l'option; s'il n'a pas la priorité la plus élevée, l'item le plus ancien ayant la priorité la plus élevée lui est préféré. Voir Routing_in_oldest.S8 (pas de condition) et Routing_in_oldest_Label.S8 (condition sur label). 3. More>> Oldest Use Queue Time : idem que 2 en remplaçant le critère du temps d'attente dans la file d'attente par celui du temps d'attente cumulé depuis l'introduction dans le système. 4. More>> Youngest Use Queue Time : choix du dernier item arrivé dans la file d'attente et respectant la condition de l'option; s'il n'a pas la priorité la plus élevée, l'item le plus récent ayant la priorité la plus élevée lui est préféré. Voir Routing_in_youngest.S8 (pas de condition sur label) et Routing_in_youngest_Label.S8 (condition sur label). 5. More>> Youngest Use Queue Time : idem que 2 en remplaçant le critère du temps d'attente dans la file d'attente par celui du temps d'attente cumulé depuis l'introduction dans le système.
	3. **Condition** sur durée d'attente maximale dans la file d'attente (*Shelf Life*); exemple avec 2' d'attente maximale Shelf Life: 2	1. **Pas de condition** (Batch by type interdit)	6. Expired Only ⇒ prélèvement du premier item de la liste de sortie ayant séjourné un temps supérieur à la durée d'attente maximale en file d'attente; utilisé normalement par une *activité* fictive (temps opératoire nul)
Cas 2 File d'attente unique en entrée et prélèvement de ***n*** items avec $n_1 \le n \le n_2$  Traitement simultané des *n* items par l'activité	1. **Classement par défaut**: classement des items dans leur ordre d'arrivée dans la file d'attente Prioritize 2. **Modification de classement** sur *Label* via l'option *Prioritize* de la file d'attente ⇒ classement des items par ordre décroissant de valeurs du *label* choisi, avec partage des ex aequo sur le critère de leur ordre d'arrivée en file d'attente. ***Exemple*** : classement des items par valeur ↓ du *label* *L_Priorité*: Prioritize (L_Priorité) Reverse Priority (Low=first)	• Chaque item possède un *label* (*L_compteur* dans exemple) ayant une valeur positive (correspondant au poids d'un item ou son volume ou …) ; si cette valeur est toujours 1, utilisez Batching based on work item count • Définition de la contrainte via *Use Label Batching* $n_1 \le \sum_i L\_compteur_i \le n_2$ Exemple n_1=5 et n_1=9 :  • Remarques: - L'activité ne peut travailler que si au moins n_1 items sont dans la file d'attente. - Les *n* items ne sont pas fusionnés en sortie. - Exclusif du *Batch by type*.	1. Priority ⇒ choix du premier item de la liste de sortie. Voir Routing_in_Use_Label_Batching_1.S8 (avec ou sans *Prioritize*)

Tableau 4 : *Routing in* file d'attente unique en entrée (Suite)

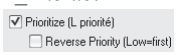
<table>
<tr><td rowspan="2">Configurations</td><td colspan="3">Conditions de prélèvement définies dans</td></tr>
<tr><td>la file d'attente
 

(⇒classement des items dans la liste de sortie)</td><td colspan="2">l'activité 

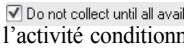


(seules les méthodes pertinentes sont retenues)</td></tr>
<tr><td rowspan="2">Cas 3
File d'attente unique en entrée et prélèvement de n items avec n fixe
 

Traitement simultané des n items par l'activité
Exemple : $n = 5$</td><td rowspan="2">1. Classement par défaut : classement des items dans leur ordre d'arrivée dans la file d'attente 

2. Modification de classement sur Label via l'option Prioritize de la file d'attente ⇒ classement des items par ordre décroissant de valeurs du label choisi, avec partage des ex aequo sur le critère de leur ordre d'arrivée en file d'attente. Exemple : classement des items par valeur ↓ du label L_Priorité:
 
</td><td>1. Reprise du Use Label Batching, avec $n = n_1 = n_2 = 7 \Rightarrow n$ items en sortie
</td><td>1. 

⇒ choix du premier item de la liste de sortie. Voir Routing_in_Use_Label_Batching_OR_Collect.S8 (avec ou sans Prioritize).</td></tr>
<tr><td>2. Pas d'option possible ici</td><td>2. Utilisation du Collect, avec saisie de n: 

Voir Routing_in_Use_Label_Batching_OR_Collect.S8 (avec ou sans Prioritize).
• Remarques :
- 

⇒ début du travail de l'activité conditionné par la disponibilité des n items
 

⇒ prélèvement au plus tôt dans la file d'attente mais début du traitement conditionné par la disponibilité des n items
- 

⇒ les n items ne sont pas fusionnés en sortie
 

⇒ les n items sont fusionnés en sortie avec valeur des labels définie par la + forte ou la + faible des valeurs trouvées 

- Voir Routing_in_Use_Label_Batching_OR_Collect.S8 (avec ou sans Prioritize).</td></tr>
</table>

Tableau 5 : *Routing in* - Files d'attente multiples en entrée, liés par la relation OU

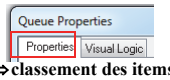
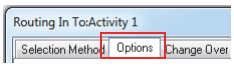
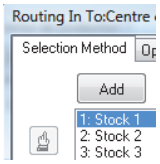
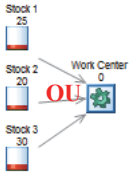
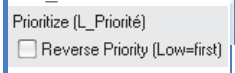
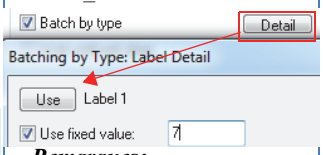
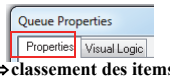
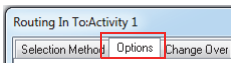
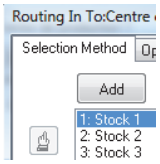
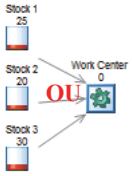
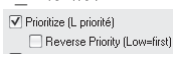
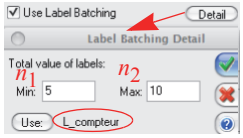
Configurations	Conditions de prélèvement définies dans		
	la file d'attente  **(⇒classement des items dans la liste de sortie)**	l'activité Routing In 	
Cas 4 files d'attente multiples en entrée et prélèvement de ***n* = 1** item 	1. **Classement par défaut**: classement des items dans leur ordre d'arrivée dans la file d'attente  2. **Modification de classement** sur *Label* via l'option *Prioritize* de la file d'attente ⇒ classement des items par ordre décroissant de valeurs du *label* choisi, avec partage des ex aequo sur le critère de leur ordre d'arrivée en file d'attente. ***Exemple*** : classement des items par valeur ↓ du *label* *L_Priorité*: 	1. **Pas de condition**  2. **Condition** sur *Label* via l'option *Batch by type* ***Exemple***: on ne tire de la file d'attente choisie (voir dernière colonne) qu'un item possédant la valeur 7 pour le *label* *Label_1*  ***Remarques:*** - exclusif du *Use Label Batching* - non utilisable si *n* > 1	1. Priority ⇒ choix du premier item de la liste de sortie de la file d'attente 1; file d'attente vide, on passe au 2e, etc. Voir Routing_in_priority.S8 (pas de condition) et Routing_in_Priority_Label.S8 (condition sur label). 2. More>> Circulate ⇒ idem que précédemment mais en partant de la file d'attente suivant la dernière utilisée, dans l'ordre de la liste (ici, partir de la file d'attente 2 si le denier item a été tiré de la file d'attente 1, partir de la file d'attente 1 si le dernier item a été tiré de la file d'attente 3). Voir Routing_in_Circulate.S8 (pas de condition sur *label* en option) et Routing_in_Circulate_Label.S8 (condition sur *label* en option). 3. More>> Oldest Use Queue Time ⇒ choix de l'item ayant le temps d'attente maximale parmi tous les items de toutes ces files d'attente ⇒ ignore le classement de la liste de sortie des items de la file d'attente. Voir Routing_in_oldest.S8 (pas de condition) et Routing_in_oldest_Label.S8 (condition sur label). 4. More>> Oldest Use Queue Time : choix de l'item ayant temps d'attente maximal parmi tous les items de ces files d'attente depuis son introduction dans le système productif ⇒ ignore le classement de la liste de sortie des items de la file d'attente. 5. More>> Youngest Use Queue Time : idem que 3, avec temps d'attente minimal et non maximal. Voir Routing_in_youngest.S8 (pas de condition sur label) et Routing_in_youngest_Label.S8 (condition sur label). 6. More>> Youngest Use Queue Time : idem que 5, avec temps d'attente minimal et non maximal 7. More>> Longest : choix de l'item dans la file d'attente la plus longue. Voir Routing_in_longest.S8 (pas de condition sur label) et Routing_in_longest_Label.S8 (condition sur label).
	3. **Condition** sur durée d'attente maximale dans la file d'attente (*Shelf Life*); exemple avec 2' d'attente maximale 	**1. Pas d'option possible** si condition sur durée d'attente maximale	8. Expired Only ⇒ prélèvement du premier item de la liste de sortie ayant séjourné un temps supérieur à la durée d'attente maximale en file d'attente; utilisé normalement par une *activité* fictive (temps opératoire nul)

Tableau 5 : ***Routing in*** **- Files d'attente multiples en entrée, liés par la relation OU (Suite)**

<table>
<tr><td rowspan="3">Configurations</td><td colspan="3">Conditions de prélèvement définies dans</td></tr>
<tr><td rowspan="2">la file d'attente
 

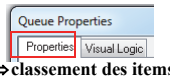
(⇒classement des items dans la liste de sortie)</td><td colspan="2">l'activité Routing In</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Cas 5
Files d'attente multiples en entrée et prélèvement de
n items avec
$n_1 \le n \le n_2$
 

Traitement simultané des n items par l'activité</td><td>1. Classement par défaut: classement des items dans leur ordre d'arrivée dans la file d'attente
 

2. Modification de classement sur Label via l'option Prioritize de la file d'attente ⇒ classement des items par ordre décroissant de valeurs du label choisi, avec partage des ex aequo sur le critère de leur ordre d'arrivée en file d'attente. Exemple: classement des items par valeur ↓ du label L_Priorité:
 
</td><td>• Chaque item possède un label (L_compteur dans l'exemple) ayant une valeur positive (correspondant au poids d'un item ou son volume ou…), éventuellement toujours la même (1, par exemple)
• définition de la contrainte via Use Label Batching
$n_1 \le \sum_i L\_compteur_i \le n_2$
Exemple n_1=5 et n_1=9 :
 

• Remarques :
- L'activité ne peut travailler que si au moins n_1 items sont dans la file d'attente.
- Les n items ne sont pas fusionnés en sortie.
- Exclusif du Batch by type.</td><td>1. Priority ⇒ choix du premier item de la liste de sortie de la première file d'attente, jusqu'à épuisement de cette file d'attente ou atteinte de n_2; sinon, on prélève dans la file d'attente suivante, etc. Voir Routing_in_Use_Label_Batching_2.S8 (avec ou sans Prioritize)
2. More>> Circulate ⇒ idem que précédemment mais en partant de la file d'attente suivant la dernière utilisée, dans l'ordre de la liste (ici, partir de la file d'attente 2 si le denier item a été tiré de la file d'attente 1, partir de la file d'attente 1 si le denier item a été tiré de la file d'attente 3).</td></tr>
</table>

Tableau 6 : *Routing in* - files d'attente multiples en entrée liées, par la relation ET

<table>
<tr><td rowspan="3">Configurations</td><td colspan="3">Conditions de prélèvement définies dans</td></tr>
<tr><td>la file d'attente
 

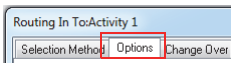
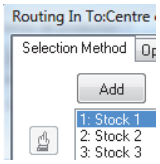
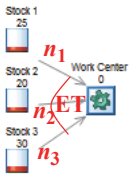
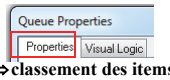
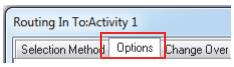
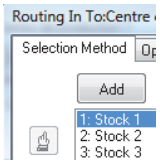
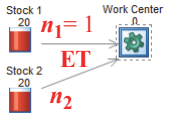
(⇒classement des items dans la liste de sortie)</td><td colspan="2">l'activité Routing In</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Cas 6
Files d'attente multiples en entrée et prélèvement de n_i items des files d'attente ii
 
</td><td>• ***Impossibilité*** d'utiliser Prioritize ⇒ classement par défaut des items dans leur ordre d'arrivée en file d'attente</td><td>• ***Impossibilité*** d'utiliser les options *Batch by type* ou *Use Label Batching*</td><td>1. ***Collect* sans *Match*** Collect Match : prélèvement impératif de $n_i \geq 0$ items des files d'attente i avant que l'activité ne commence le traitement
- *Exemple*: $n_1 = 3$ items de la file d'attente 1, $n_2 = 2$ items de la file d'attente 2 et $n_3 = 5$ items de la file d'attente 3
Par défaut: $n_i = 1$
• Modification des valeurs par défaut: se positionner sur la ligne de la file d'attente i (exemple, $i = 3$) 1: 3 Stock 1 / 2: 2 Stock 2 / 3: 1 Stock 3, saisie du nombre (exemple, 5) Collect 5 ⇒ 1: 3 Stock 1 / 2: 2 Stock 2 / 3: 5 Stock 3. Illustration avec Routing_in_Collect_and_Assemble_1.S8
Compléments:
- Do not collect until all available ⇒ les items ne sont prélevés dans les files d'attente que lorsque tous les items nécessaires sont disponibles
- Do not collect until all available ⇒ les items sont prélevés dans les files d'attente dès qu'ils sont disponibles mais le traitement ne commence que lorsque tous les items sont réunis.
- Assemble par défaut, tous les items sont fusionnés ⇒ item unique en sortie (sauf usage du *Batching* en *Routing Out*). Pour modifier cette option: More>>, puis décocher Assemble ⇒ Assemble ⇒ les items entrants ne sont pas fusionnés et gardent leurs caractéristiques initiales en sortie de l'activité. Les valeurs des *labels* de l'item fusionné sont défini par l'une des 3 options du bas de la fenêtre; l'option Selected de « *label value from* » affecte les valeurs du dernier item prélevé dans la file d'attente sélectionnée dans la liste des files d'attente entrantes.
- Possibilité de modifier les n_i sous *Visual Logic* avant tout prélèvement d'items Before Selecting via la commande *Set Collect Number* ⇒ permet d'utiliser des nomenclatures variables selon une commande passée. Illustration avec Routing_in_Collect_and_Assemble_2.S8 dans lequel les files d'attente multiples sont spécialisées. Le cas, plus complexe, de files d'attente multiples banalisés est traité aux cas 8 et 9.</td></tr>
</table>

Tableau 6 : *Routing in* - files d'attente multiples en entrée liées, par la relation ET (Suite)

Configurations	Conditions de prélèvement définies dans		
	la file d'attente  **(⇒classement des items dans la liste de sortie)**	l'activité Routing In  	
Cas 7 Files d'attente multiples en entrée et prélèvement d'items *partageant une même valeur de label*  Utilisé normalement avec 2 files d'attente seulement (le premier correspondant à des réquisitions, le second à des produits)	• ***Impossibilité*** d'utiliser Prioritize ⇒ classement des items dans leur ordre d'arrivée en file d'attente • ***Impossibilité*** d'utiliser Min Wait Time: dans l'une des files d'attente entrantes	• ***Impossibilité*** d'utiliser l'option *Batch by type* (mais sans intérêt avec la méthode de sélection *Match*) et *Use Label Batching*	1. ***Collect* avec *Match*** Collect Match: prélèvement d'items partageant tous la valeur d'un *label* du premier item de la première file d'attente de la liste (⇒ $n_1 = 1$ et, normalement, deux files d'attente) - *Exemple*: $n_1 = 1$ item de la file d'attente 1 (seule valeur possible), $n_2 = 2$ items de la file d'attente 2 Pour la saisie des n_i, voir ci-dessus (cas 6) Sélection du label: en cochant Match, une fenêtre de sélection des *labels* disponibles s'ouvre; choix du *label* Label_1 ⇒ Match on: Label_1 ⇒ les n_2 items prélevés dans la file d'attente 2 partagent tous nécessairement la valeur prise par le *label* Label_1 du premier item de la première file d'attente dans la liste. Illustration avec Routing_in_Collect_and_Match_1.S8 ***Compléments:*** - Les items sont prélevés dans les files d'attente dès qu'ils sont disponibles mais le traitement ne commence que lorsque tous les items demandés sont réunis. - Assemble par défaut, tous les items sont fusionnés ⇒ item unique en sortie (sauf usage du *Batching* en *Routing Out*). Pour modifier cette option: More>>, puis décocher Assemble ⇒ Assemble ⇒ les items entrants ne sont pas fusionnés et gardent leurs caractéristiques initiales en sortie de l'activité. Les valeurs des *labels* de l'item fusionné sont défini par l'une des 3 options du bas de la fenêtre; l'option Selected de « *label value from* » affecte les valeurs du dernier item prélevé dans la file d'attente sélectionnée dans la liste des files d'attente entrantes. - Possibilité de modifier dynamiquement les n_i sous *Visual Logic* avant tout prélèvement d'items Before Selecting via la commande *Set Collect Number* ⇒ permet d'utiliser des nomenclatures variables selon une commande passée. Illustration avec Routing_in_Collect_and_Match_2.S8

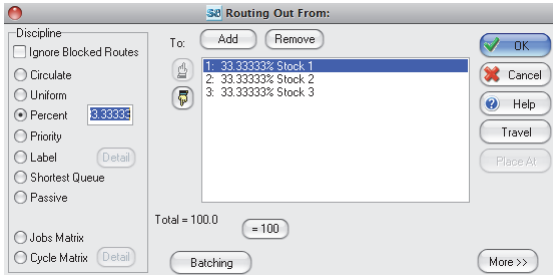
II DESTINATIONS EN SORTIE DE L'ACTIVITÉ

Après traitement par l'activité, l'item est orienté (ou les items sont orientés) vers une (ou plusieurs) file(s) d'attente. La définition de ces destinations en sortie se fait dans la fenêtre ouverte par le bouton Out. Si l'*activité* n'est connectée qu'à une seule file d'attente en sortie, le problème de l'orientation de l'item en sortie ne se pose pas; on ne s'intéressera donc qu'au cas des destinations multiples (§ II-1). Cette fenêtre permet de déclencher un certain nombre de traitements complémentaires, indépendants du nombre de destinations (§ II-2, page 88).

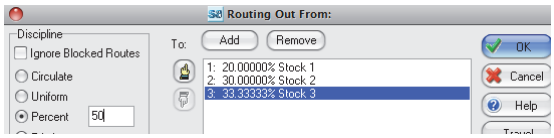
II-1 Détermination de la destination en sortie d'une activité

Le bouton Out liste les files d'attente de destination en sortie. S'il n'y en a qu'un, aucune ambiguïté n'existe et la règle de sortie par défaut est Priority. On traitera ici le cas où l'activité est reliée à plusieurs files d'attente en sortie. Ces destinations sont listées et numérotées dans la fenêtre ci-dessous. Cet ordre est important dans l'usage de certaines règles de choix de la destination et peut être modifié en sélectionnant une file d'attente de la liste puis en la faisant monter ( ) ou descendre ( ) dans la liste. Les **règles de sortie** (ou **disciplines**, pour reprendre la terminologie en usage dans la théorie des files d'attente) sont les suivantes:

- L'**affectation circulaire** (Circulate) conduit à envoyer le premier item arrivé dans la première file d'attente, le deuxième dans la deuxième file d'attente, le troisième item dans la troisième file d'attente, le quatrième item dans la première file d'attente, le 5e item dans la deuxième file d'attente, etc. Voir l'exemple Routing_out_Circulate.S8.
- Affectation **aléatoire équiprobable** (Uniform) où chaque direction possible a la même probabilité d'être utilisée par un item sortant; voir l'exemple Routing_out_Uniform.S8.
- Affectation suivant la plus faible **longueur des files d'attentes** (Shortest Queue) où l'item est dirigé vers la file d'attente la plus courte; voir l'exemple Routing_out_Shortest.S8).
- Affectation **aléatoire non équiprobable** (Percent) s'effectue en affectant une probabilité à chaque destination possible; la sélection de cette option conduit par défaut à une équiprobabilité des directions en sortie; la saisie la probabilité de la première destination conduit à l'écran ci-contre.



On modifie alors la probabilité et sélectionne la destination suivante en lui affectant une probabilité, etc. Le changement de destination valide la saisie précédente. On obtient alors, l'écran ci-contre; voir l'exemple Routing_out_Percent.S8.



- Affectation définie par une valeur de **label** de l'item sortant (Label). La direction de sortie est définie par la valeur d'un *label* qui est définie antérieure-

ment. Cette valeur peut être certaine, comme c'est le cas de la destination en sortie du secrétariat (voir page 31). La valeur du *Label* peut aussi être une occurrence d'une variable aléatoire, ce qu'illustre l'exemple Routing_out_Label_1.S8. Cette variable aléatoire peut dépendre d'une caractéristique d'un item; par exemple la probabilité qu'un patient arrivant dans un service d'urgences ait à se déplacer avec l'aide de brancardiers dépend de la gravité de son état lors de son arrivée aux urgences. L'utilisation de *Visual Logic* est alors indispensable; dans l'exemple Routing_out_Label_2.S8 des niveaux de gravité croissant (de 1 à 3) de l'état du patient à l'arrivée sont définis ($p_1 = 30\%$; $p_2 = 50\%$; $p_3 = 20\%$) avec des probabilités d'avoir besoin d'un brancardage respectivement de 10%, 40%, 100%. Pour des problèmes plus complexes, il est judicieux d'utiliser les possibilités de paramétrisation offertes par l'usage de l'*Information Store*, ce qu'illustre l'exemple Routing_out_Label_3.S8) s'appuyant sur les mêmes données numériques. L'exemple Parametrisation_distribution_probability_profile.S8 montre comment définir la valeur d'un *label* utilisé pour définir la direction qu'un item doit emprunter à la sortie d'une activité à partir d'une distribution de probabilités dont les paramètres, dépendant d'un autre *label* de cet item, sont lus dans l'*Information Store*. L'exemple Distribution_Discrete_et_Information_Store.S8, que l'on commentera à la page 100, traite simultanément du problème de l'orientation d'un item et de celui de son temps de traitement à partir de la valeur prise par un label, valeur définie aléatoirement, et d'informations stockées dans l'*Information Store*. L'exemple Open_shop.S8 traite d'une sortie sur la valeur d'un *label* qui dépend de celles de plusieurs autres *labels*; l'exemple Routing_out_2_Labels.S8 montre comment obtenir un résultat similaire en faisant appel à des activités fictives si le choix d'une file d'attente de destination dépend simultanément de valeurs prises par deux *labels*. Enfin, l'exemple Choix_Production_stock_mini.S8 montre comment décider quelle référence lancer en production, sachant que le critère retenu est celui de la référence possédée par les items se trouvant dans la file d'attente possédant le moins d'items (recherche d'un minimum sur plusieurs files d'attente).

- Affectation définie selon l'**ordre de la liste** (

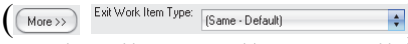
): on choisit la première file d'attente, sauf si elle est bloquée (capacité maximale atteinte) auquel cas on prend la seconde, etc.; voir l'exemple Routing_out_Priority.S8.
- **Passivité** de l'activité en sortie (

): dans ce cas, l'activité attend que l'item qu'elle vient de traiter soit demandé par une autre activité; de ce point de vue, l'activité se comporte alors comme une file d'attente.
- Les options *Jobs Matrix* et *Cycle Matrix* seront abordées à la page 114.

On verra à la page 111, comment traiter avec l'aide d'une activité fictive, certains problèmes plus complexes d'orientation des items en sortie d'une activité «réelle».

II-2 Traitements complémentaires en sortie d'une activité

Le *Routing out* permet d'effectuer non seulement l'orientation de l'item (ou des items) en sortie mais aussi d'exécuter un certain nombre de traitements complémentaires.

Par défaut, l'item sortant est de la même catégorie que l'item entrant, ce qui est implicitement défini au bas de la fenêtre du *Routing Out* (

). Rappelons que pour rendre la simulation plus convaincante visuellement, il est possible de **changer l'icône** d'un item comme on l'a vu à la page 57. Il est également possible de **changer de type d'item** si le traitement réalisé dans l'activité conduit à un tel chan-

gement. Pour ce faire, on sélectionne la classe désirée dans le menu déroulant 

. L'exemple Routing_in_Collect_and_Assemble_1.S8 illustre la confection de cartons de 6 bouteilles, à partir d'emballages et de bouteilles.

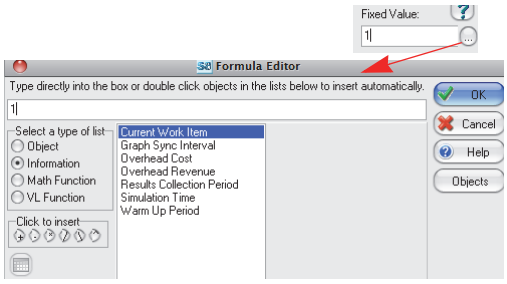
Par défaut, si un seul item rentre dans l'activité, un seul item en sort après traitement. Si plusieurs items sont utilisés par le traitement, on a autant d'items en sortie, sauf si la méthode de sélection *Collect* est utilisée avec l'option *Assemble* (voir pages 79, 85 et 86). Dans la modélisation de certains processus, il est nécessaire de remplacer l'item entrant (carton de 6 bouteilles, par exemple) par plusieurs items en sortie (6 bouteilles à mettre en rayon, par exemple), l'activité ayant pour vocation d'éclater un lot de marchandises. Moins intuitif est le cas d'une information émise (plat commandé dans un restaurant, prélèvement sanguin…), obligeant à émettre 2 items en sortie de l'activité, le premier de nature informationnel (commande, prélèvement) et le second correspondant à l'émetteur de l'information (client du restaurant, patient d'un box d'urgence) vers 2 files d'attente (ce qui implique l'usage du *Circulate* en sortie); l'émetteur reste sur place en attendant de recevoir ultérieurement le résultat de sa demande (plat commandé, analyse sanguine…), ce qui mettra en jeu une autre activité.

On peut modifier la **taille du lot en sortie**, par défaut égale à 1, à l'aide du bouton 

qui ouvre la fenêtre suivante permettant de spécifier le nombre d'articles en sortie. La taille du lot peut être:

- un **nombre** fixe **saisi** directement dans la fenêtre de saisie

en remplacement du 1 par défaut;
- la valeur prise par un *label* de l'item traité; pour ce faire, placer le curseur sur la fenêtre de saisie et cliquer, le bouton  apparaît à droite de cette fenêtre de saisie; l'activation du bouton provoque l'ouverture de la fenêtre de saisie de l'éditeur de formule; cette valeur de label ne doit pas être définie dans le *Label Actions* de cette activité car ces toutes actions sont exécutées individuellement pour chacune unité du lot sortant de cette activité;



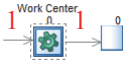
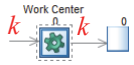
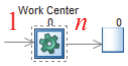
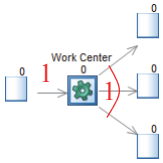
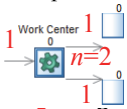
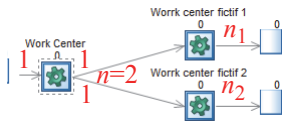
- la valeur prise par une variable ou un élément de tableau stocké dans l'***Information Store*** (ces éléments se trouvant ajoutés à la liste de l'éditeur de formule ci-dessus); dans ce dernier cas, on utilise généralement des valeurs de *labels* prises par l'item détenu par l'activité, pour repérer la ligne et/ou la colonne dans laquelle se trouve l'information pertinente pour définir la taille du lot.

L'exemple Routing_Out_Fractionnement_1.S8 illustre une mise en linéaire, dans un supermarché, de bouteilles arrivant dans des cartons. L'exemple Routing_Out_Fractionnement_2.S8 illustre le cas de patients dans des box d'un service d'urgence: après prélèvement d'un échantillon par l'infirmière, l'échantillon est envoyé au laboratoire tandis que le patient attend dans le box (⇒ lot de 2 en sortie et *Circulate*); les résultats de l'analyse sont ensuite étudiés avec le patient par le médecin et l'infirmière (*Collect et Match* (voir page 86) sur le numéro de patient attendant dans le box).

On peut avoir besoin de désassembler un objet composite (conteneur plein comportant un kanban et un lot de 5 objets identiques, par exemple) en plusieurs objets homogènes (conteneur vide, kanban, lot de 5 pièces). La conception du *Routing Out* ne permet pas directement d'éclater un item en des lots de tailles différentes, envoyés vers des files d'attente différentes. Dans ce cas, on est obligé de faire appel à plusieurs activités fictives, un par lot différent de taille supérieur à 1. L'exemple Routing_out_Desassembler.S8 illustre la solution de ce problème sur un exemple de kanban.

Le tableau 7 résume les différents cas de figure rencontrés dans la modélisation des sorties d'items d'une activité.

Tableau 7 : Lotissement en sortie d'une activité

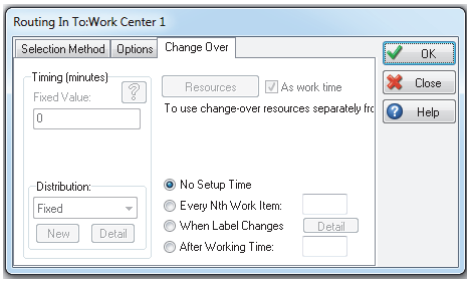
	Un seul item en sortie	Plusieurs items en sortie	
file d'attente unique en sortie	**En entrée de l'activité**: 1 seul item prélevé à partir d'un ou plusieurs files d'attente (ou plusieurs items fusionnés par l'option *Assemble*)  *Batching* **par défaut** (*n*=1) en *Routing Out* **Règle de sortie:** quelconque	**En entrée de l'activité**: *k* items prélevés à partir d'un ou plusieurs files d'attente (se rencontre avec *Use label Batching* et *Collect* sans cocher *Assemble*)  *Batching* **par défaut** (*n*=1) en *Routing Out* (cas rencontré avec *Use label Batching* et *Collect* sans cocher *Assemble*) **Règle de sortie**: quelconque	**En entrée de l'activité**: 1 seul item prélevé à partir d'un ou plusieurs files d'attente (ou plusieurs items fusionnés par l'option *Assemble*)  *Batching n* en *Routing Out* **Règle de sortie**: quelconque
Files d'attente multiples en sortie	**En entrée de l'activité**: 1 seul item prélevé à partir d'un ou plusieurs files d'attente, ou plusieurs items fusionnés par l'option *Assemble*  *Batching* **par défaut** (*n*=1) en *Routing Out* **Règle de sortie:** quelconque	**En entrée de l'activité**: 1 seul item prélevé à partir d'un ou plusieurs files d'attente, ou plusieurs items fusionnés par l'option *Assemble*  *Batching n*=2 en *Routing Out* **Règle de sortie:** *Circulate* **Remarque**: *n* est égal au nombre de files d'attente en sortie	**En entrée de l'activité**: 1 seul item prélevé à partir d'un ou plusieurs files d'attente, ou plusieurs items fusionnés par l'option *Assemble*  *Batching n*=2 en *Routing Out de l'activité et batching* n_1 *et* n_2 *en Routing Out des activités fictives 1 et 2* **Règle de sortie:** *Circulate*

Quelques options complémentaires peuvent être activées à partir de cette fenêtre ouverte par le bouton Out.

- Possibilité de **bloquer la sortie** du (ou des) item(s) tant que l'activité ne peut pas recommencer à travailler en raison d'une absence de travail à effectuer; l'activité joue alors le rôle d'une file d'attente (☑ Hold work here until more input work is ready).
- **Fixation du temps de transport** vers une file d'attente. En cliquant sur une file d'attente figurant dans la liste de cette fenêtre, le bouton Travel devient actif; il permet de définir le temps de transport mis par l'item pour atteindre cette file d'attente, ce qui permet de s'affranchir des contraintes liées à la vitesse de déplacement identique pour tous les déplacements d'items (voir l'exemple Temps_de_transport.S8). Rappelons que dans beaucoup de simulations, les temps de déplacement sont négligeables par rapport aux temps de traitement, ce qui conduit à les neutraliser comme on l'a vu à la page 50.

III Définition du temps opératoire et du temps de lancement d'une activité

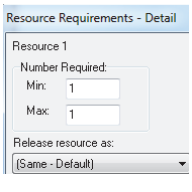
Les distributions de probabilités utilisables pour définir le temps d'exécution d'une opération ont été étudiées au § II-2, page 35).



Dans certains cas, il est souhaitable d'ajouter à ce temps de traitement un **temps de lancement** (inexistant par défaut); celui-ci se définit par une autre distribution de probabilités définie dans l'onglet Change Over de la fenêtre ouverte par le bouton Routing In. Dans cet onglet, l'option par défaut est No Setup Time. Si l'on choisit l'option Every Nth Work Item: (qui désélectionne l'option No Setup Time), il faut saisir le nombre d'items qui déclenche un nouveau temps de lancement, qui est une occurrence de la distribution de probabilités définie dans la partie gauche de la fenêtre. Ce temps de lancement additionnel peut être déclenché par le changement de valeur d'un *label* (When Label Changes) choisi dans la liste des *labels* existants dans une fenêtre affichée lorsque cette option est choisie; ce label est souvent celui d'un numéro d'Ordre de Fabrication (OF), ce qui conduit à ajouter un temps de lancement à chaque changement d'OF, celui-ci étant fragmenté en plusieurs lots. On peut enfin ajouter un temps fixe de lancement par l'option After Working Time: qui déclenche de la même façon la définition d'une distribution de probabilité. Cette dernière option n'a de sens que si le temps opératoire de base défini dans la fenêtre générale de l'activité est un temps proportionnel à la valeur prise par un label.

IV Ressources mobilisées (boutons *Resources*, *Priority* et *Shifts*)

Les caractéristiques des ressources ont été définies au chapitre V, page 68, où nous avons souligné que la création d'une ressource ne s'impose que si la ressource est partagée par plusieurs activités. Plusieurs ressources (opérateur et outillage ou médecin et infirmière, par exemple) peuvent être simultanément requises par l'exécution d'une opération laquelle ne peut débuter que si toutes les ressources requises sont présentes; ces ressources sont libérées simultanément à l'achèvement de cette opération.



La **quantité** de chaque **ressource mobilisée** dans l'exécution d'une opération dans une activité est de 1 par défaut mais peut être modifiée (l'utilisation du bouton Detail de la fenêtre ouverte par le bouton Resources de l'activité ouvrant une fenêtre (voir ci-contre) dans laquelle on peut faire varier le paramètre entre 2 bornes initialisées à 1. L'opération ne peut commencer tant que l'effectif minimal n'est pas disponible. Le plus souvent, ces bornes sont identiques; dans le cas contraire, l'opération mobilisera la plus forte quantité *n* de ressource disponible comprise entre ces deux bornes *Min* et *Max*; le temps opératoire de l'opération étant implicitement spécifié pour l'effectif minimal, le temps opératoire utilisé dans la simu-

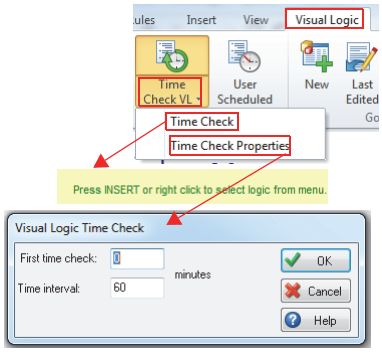
lation sera celui généré pour la loi définie pour cette activité, divisé par le rapport *n/Min*. Ajoutons enfin que ces différentes unités d'une ressource sont réputées totalement interchangeables.

Si plusieurs activités réclament au même moment une même ressource disponible en quantité insuffisante, le choix des bénéficiaires est aléatoire. L'option Priority des activités permet de hiérarchiser ces activités et donc de choisir les bénéficiaires en cas de pénurie. Un niveau de **priorité** est attribué aux activités potentiellement candidates (niveau modifiable au cours de la simulation via *Visual Logic*). Si plusieurs activités ont le même niveau de priorité et que le nombre d'unités disponibles de la ressource est insuffisant, le choix est aléatoire. L'exemple Priority_Resources_niveau_fixe.S8 illustre l'usage de cette option : les centres de production 1 et 2 ont la priorité 60, contre 50 pour le centre de production 3 ; tous ont besoin d'un opérateur et tous puisent dans la même file d'attente, il y a 3 opérateurs ce qui fait que l'activité à priorité la plus faible n'est utilisée que si les deux autres le sont déjà ; dans une seconde version, chaque activité a sa propre file d'attente et deux opérateurs seulement sont disponibles ; l'activité à la priorité la plus faible fonctionne qu'à la condition que l'un des deux autres n'ait pas de travail en attente, ce qui revient à pénaliser les items orientés vers l'activité à priorité la plus faible.

Lorsque, sur une période de temps donnée, le nombre d'activités susceptibles de mobiliser une ressource est supérieur à l'effectif disponible de cette ressource on observe une volatilité d'affectation conduisant à une utilisation discontinue des activités généralement inacceptable parce que ce comportement ne correspond pas à ce qui se passe sur le terrain (voir l'exemple Resources_Shift_1.S8). Il convient alors de faire varier le nombre d'activités disponibles en le faisant coïncider avec celui de la ressource partagée. La solution à retenir dépend de ce que les activités concernées prélèvent ou non les items dans une même file d'attente.

- Dans le cas d'une file d'attente commune aux activités partageant cette ressource il convient d'appliquer également le concept d'équipe aux activités concernées. Pour ce faire, il faut utiliser le bouton Shifts de la fenêtre de l'activité, ce qui ouvre une fenêtre similaire à celle ouverte pour les ressources et dans laquelle on associe à l'activité une ou plusieurs plages de temps. L'exemple Resources_Shift_2.S8 reprend l'exemple Resources_Shift_1.S8 précédent en neutralisant l'activité 2 en cas d'insuffisance de ressource. Si, pour chaque plage de temps, il y a coïncidence entre le nombre d'unités disponibles pour une ressource et celui des activités mobilisant cette ressource, il devient inutile de déclarer cette ressource dans la modélisation : l'association des plages de présence à chaque activité visée étant suffisante (ce qu'illustre la seconde modélisation de l'exemple précédent). L'appel à l'une de ces deux solutions implique l'usage d'une file d'attente commune, ce qui est souvent inacceptable dans les services : par exemple, pour payer, les clients d'un hypermarché se placent toujours dans la file d'attente d'une caisse ouverte, l'idée d'une file d'attente commune à l'ensemble des caisses n'ayant aucun sens tant du point de vue organisationnel, qu'économique.

- Pour garder une file d'attente spécifique par activité, il faut pouvoir désactiver l'accès à la file d'attente d'une activité inaccessible momentanément.
  - Une première solution consiste à utiliser des activités fictives associées à chaque groupe de destinations possibles, à définir le numéro de route vers l'une de ces activités fictives, en sortie du point d'entrée, par un *label* (bouton Out puis Label) dont la valeur est défini dans Actions comme le résultat (Set to:) d'une distribution de type *Fixed* dont la constante est égal au nombre d'unités disponibles de la ressource à mobiliser (à une constante additive près si l'effectif minimal n'est pas 1), ce qu'illustre l'exemple Work_Center_Shifts_Stock_in_individuel_1.S8.
  - Une solution alternative ne s'appuie pas sur l'utilisation d'*activités* fictives; elle consiste à valider ou neutraliser certains chemins allant du point d'entrée aux files d'attente en fonction du nombre d'unités disponibles. Il faut alors passer comme illustré ci-contre par l'onglet Visual Logic de la barre de menu général pour créer un programme en *Visual logic* exécuté périodiquement, ce qui implique que la périodicité choisie permette de détecter l'événement modifiant le nombre d'activités à neutraliser. On peut alors utiliser les instructions de création/suppression de flèches (link / unlink) permettant de modifier dynamiquement la cartographie du processus, ce qu'illustre l'exemple Resources_Shift_3.S8 ou celui, plus astucieux dans sa conception, Work_Center_Shifts_Stock_in_individuel_2.S8.



L'exécution d'une opération dans une activité, peut nécessiter une compétence que possèdent une ou plusieurs ressources interchangeables sur la compétence considérée, une ressource étant caractérisée par une ou plusieurs compétences. Cette problématique, au cœur de l'amélioration de la performance par une **polyvalence** accrue a été présentée à la page 69 et est illustrée par les exemples Pool.S8 et Ressource_Polylavence_pool.S8.

V AUTRES CARACTÉRISTIQUES D'UNE ACTIVITÉ

V-1 Action sur les *labels* (bouton *Label actions*)

Les actions sur les *labels* (Actions) ont déjà été présentées (voir exemple introductif du cabinet médical et page 57).

V-2 Duplication d'une activité (bouton *Replicate* ou *Duplication Wizard*)

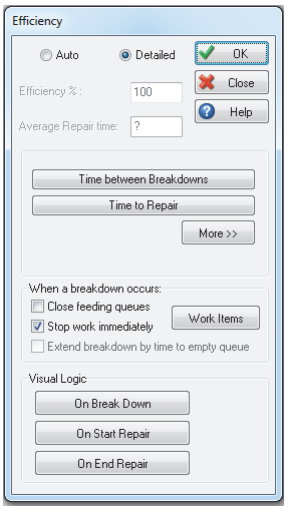
L'utilisation du bouton Replicate = 1 permet une duplication automatique d'activités de caractéristiques strictement identiques, prélevant dans les mêmes files d'attente et envoyant les items traités dans les mêmes files d'attente; on est donc dans une configuration de processeurs parallèles représentés «collectivement» dans le modèle de simulation (cas 3 de la figure 5 de la page 14), ces processeurs étant fusionnés en une activité caractérisée par un *Replicate* supérieur à 1. Voir

l'exemple de la simulation Replication_work-center.S8. Attention l'utilisation de l'option Batch By Type est ***incompatible*** avec l'option Replicate (bouton neutralisé).

Si tel n'est pas le cas (une file d'attente spécifique par activité, par exemple), il faut procéder à une duplication de l'ensemble d'objets sélectionnés, le plus simple étant de passer par l'option *Duplication Wizard* de l'onglet Home, pour conserver les liens, toutes les caractéristiques des composants dupliqués. Il peut être intéressant également de créer un composant fusionnant cet ensemble (voir page 110; l'exemple Duplication_composant.S8 illustre une duplication de composant).

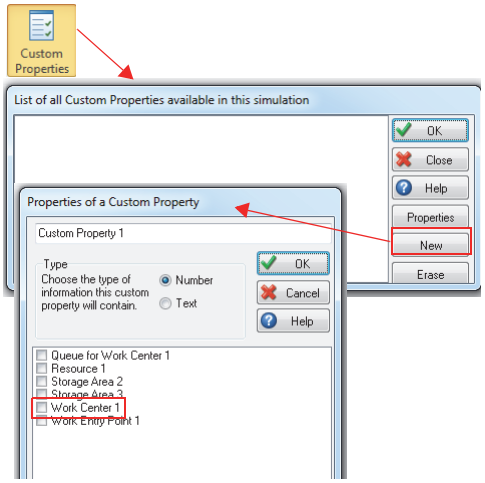
V-3 Fiabilité d'une activité (bouton *Efficiency*)

L'activité est ici nécessairement un poste de travail utilisant une machine (au sens large). Par défaut, ce poste est réputé fonctionner sans panne. On peut cependant simuler des **pannes** de ce centre de production par le bouton Efficiency ainsi que les temps de réparation. Cette possibilité permet d'intégrer des préoccupations de fiabilité et de maintenance curative. Le bouton Time between Breakdown souvre une fenêtre de définition de la loi de l'intervalle entre deux pannes et le bouton Time to Repair ouvre une fenêtre permettant de définir la distribution du temps de réparation. Notons qu'une opération de maintenance préventive se traitera par l'intermédiaire d'un item spécifique, mobilisant éventuellement des ressources particulières, s'intercalant parmi les autres items traités par ce poste de travail.

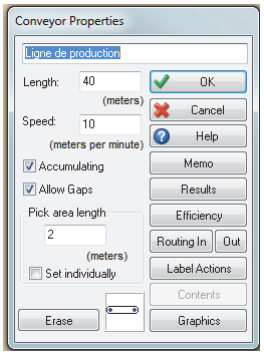


V-4 Possibilité de caractérisation d'une activité par des *labels*

Simul8 permet de caractériser une activité (mais aussi, une ressource ou une file d'attente) comme un item, par un ensemble de caractéristiques utilisables par Visual Logic dans certaines situations complexes et jouant le même rôle que les *labels* pour les items. Ces attributs, qualifiés de **Custom Object Properties**, sont créés dans la fenêtre (voir ci-contre) ouverte par le bouton du menu Visual Logic de la barre de menu général. Comme pour les labels, on a le choix du type (numérique ou littéral) et le nom est modifiable; cette fenêtre affiche la liste de tous les points de passage créés et il suffit de cocher le point retenu pour lui affecter cette propriété (ce qui correspond à l'affectation d'un label à un type d'items). La valeur d'un *custom object property* est assignée sous *Visual Logic* par une instruction de type Set et est utilisable, au même titre qu'un label, dans un programme en *Visual Logic*. Cette possibilité n'est pas disponible dans la version étudiante.



VI UN POSTE DE TRAVAIL PARTICULIER: LE CONVOYEUR



Le **convoyeur** est créé par l'outil Conveyor de la barre latérale de création du modèle physique, en utilisant le bouton Advanced Building Blocks. C'est un équipement un peu particulier jouant à la fois un rôle d'activité et de file d'attente. Sous Simul8, il peut prélever directement un item dans une file d'attente, à condition que l'option *Priority* soit cochée dans le *Routing in* du convoyeur. Pour gérer des processus plus complexes, un poste de travail (activité) doit alimenter le convoyeur pour lequel l'option *Passive* a été cochée dans le *Routing in*. Voir l'exemple Convoyeur.S8. Le convoyeur a une longueur et les items qu'il traite avancent à une certaine vitesse: le temps de traitement en découle (il est d'une minute dans l'exemple et correspond au temps de cycle de la ligne).

Le convoyeur déplace un nombre d'items égal au quotient de sa longueur (L = 40 mètres) par le produit de l'intervalle séparant 2 arrivées successives (I = 0,5 minute), par la vitesse du convoyeur (V = 10 mètres à la minute), ce qui donne une capacité $K = L/(I \cdot V) = 8$ items. L'intervalle J entre 2 débuts de positions d'item sur la ligne est $J = L/K = I \cdot V = 5$ mètres. Si cet intervalle J est inférieur à la taille T de l'item (définie dans la fenêtre de propriété d'une classe d'items; voir page 57), la capacité de la ligne est $K = L/T$; par exemple, si T = 8 mètres, la capacité de la ligne est K = 5 items; dans ce cas, l'intervalle entre deux entrées successives dans le convoyeur est $I' = T/V = 0{,}8$ minute. Autrement dit, $K = Min(L/(I \cdot V), L/T)$ et le débit (ou temps de cycle) du convoyeur est $Max(I, T/V)$.

Si le convoyeur n'est pas alimenté par une activité mais directement par une file d'attente, il prélève les items dans cette file d'attente de façon à saturer sa capacité (à la valeur L/T; ce qui donne 40 unités si T = 1 mètre), avec un intervalle entre deux entrées successives dans le convoyeur de $I' = T/V = 0{,}1$ minute. Si les arrivées sont plus rapides il y a accumulation dans la file d'attente; si les arrivées sont moins rapides, le convoyeur ne marche pas à sa capacité maximale.

Dans tous les cas de figure, le débit en sortie de la ligne est, bien évidemment égal à celui observé à l'entrée de la ligne.

L'option *Accumulating* (Accumulating) permet au convoyeur de continuer à fonctionner quand un item arrive «en fin de course» et l'option *Allow Gaps* (Allow Gaps) permet au convoyeur de ne pas s'arrêter si, au moment où le premier poste se libère, aucun item n'est prêt à prendre la place. En sortie du convoyeur, on peut trouver aussi bien une file d'attente qu'un centre de production (activité).

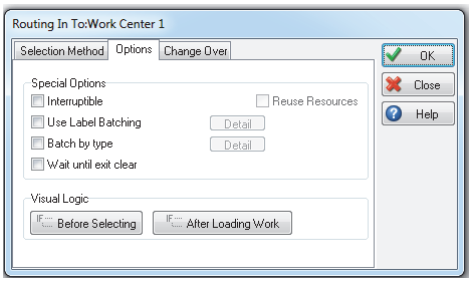
VII ACTIVITÉ ET *VISUAL LOGIC*

VII-1 Possibilités d'exécution de programmes en *Visual Logic* lors de l'utilisation d'une activité

L'activité offre sept possibilités d'inclusion de programme en Visual Logic. Ces possibilités doivent être placées dans une perspective de chronologie d'événements.

Deux possibilités sont offertes au niveau du 

:



- Le programme créé dans la fenêtre ouverte par le bouton 

est exécuté avant tout prélèvement d'item ; on l'utilise, en particulier, pour bloquer le prélèvement d'items lorsque certaines conditions sont réunies (voir, par exemple, l'utilisation qui en est faite dans Synchronisation_stock_produits_Multiples.S8).
- Le programme créé dans la fenêtre ouverte par le bouton 

est exécuté juste après le prélèvement mais avant que ne débutent le traitement et les actions sur label. Ce programme permet, par exemple,
  - de modifier les caractéristiques de la distribution du temps opératoire d'une activité (voir l'exemple DOCTEUR_Information_Store_1.S8),
  - de modifier certains paramètres utilisés en *Routing out* (voir l'exemple Distribution_Discrete_et_Information_Store.S8),
  - de modifier des données stockées dans l'*Information Store*, notamment pour bloquer ou débloquer l'accès à l'activité (ce qu'illustre l'exemple Box.S8).

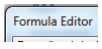
Au niveau de la fenêtre principale de définition des propriétés d'une activité, on trouve une possibilité d'exécution d'un programme en *Visual Logic* déclenché par un changement d'état de l'activité (

). Ce concept de changement d'état n'est pas très clair, aucune documentation n'étant disponible sur ce point (et quelques autres) ; il est donc préférable de ne pas l'utiliser.

La quatrième possibilité d'exécuter un programme en *Visual Logic* écrit dans la fenêtre ouverte par le bouton 

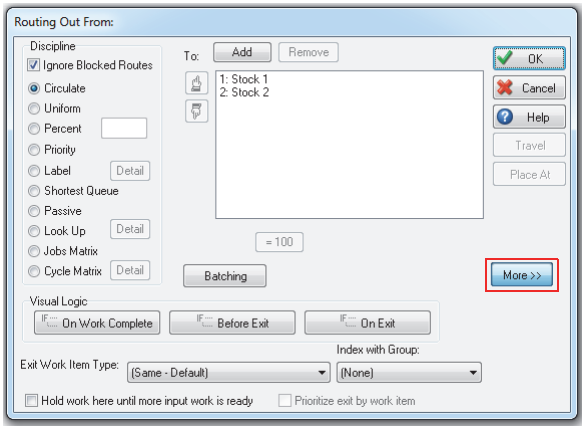
de la fenêtre ouverte par le bouton 

de la fenêtre principale de l'activité (ou d'un point d'entrée, ces deux fenêtres étant les mêmes ; on a illustré ce cas de figure aux pages 39 et suivantes). Si l'instruction à déclarer porte sur une donnée de l'*Information Store*, on peut effectuer cette instruction de type set en utilisant le bouton 

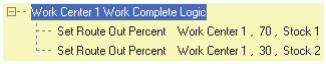
qui ouvre la fenêtre classique 

.

Une fois le travail exécuté dans l'activité, ce qui inclut l'exécution éventuelle de programmes *Visual Logic* présentés au paragraphe précédent, trois nouvelles possibilités d'exécution de programme sont offertes au niveau du *Routing Out* (Out), après avoir utilisé le bouton More >> pour faire apparaître le bas de la fenêtre :



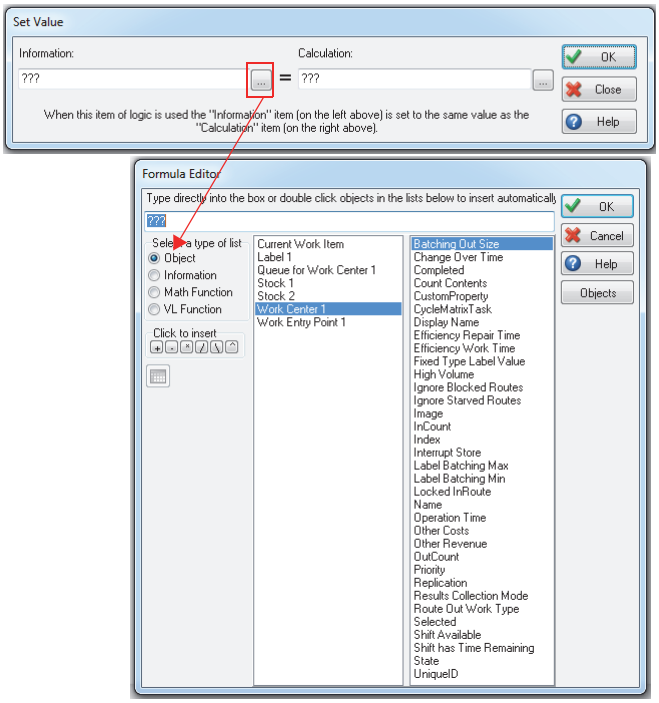
- On Work Complete On ne peut plus à ce niveau modifier les valeurs de *label* mais on peut encore agir sur les probabilités associées aux directions de sortie par l'option Percent, étudiée page 87, ce qu'illustre la copie d'écran ci-contre.



- Il revient au même d'utiliser Before Exit et On Exit si l'item n'est pas empêché de se rendre dans la direction choisie (saturation d'une file d'attente, s'il s'agit d'une file d'attente ou activité en cours d'utilisation, s'il s'agit d'une activité). Le premier programme permet de déclencher une action si l'item ne peut quitter l'activité. Une utilisation intéressante du second programme est la modification d'un paramètre de l'*Information Store* permettant de bloquer ou libérer l'accès à cette activité ou une autre activité du modèle de processus, ce qu'illustre l'exemple Box.S8.

VII-2 Propriétés d'une activité mobilisables par *Visual Logic*

Les instructions de *Visual Logic* peuvent utiliser des propriétés de la classe «activités». Il s'agit d'un ensemble de caractéristiques prédéterminées, associées à n'importe quelle activité et jouant le même rôle que les *labels*. Ces caractéristiques ont des valeurs par défaut utilisées lors de la création d'une activité pendant la modélisation d'un processus. Elles sont utilisables dans les programmes écrits en *Visual Logic*. La liste de ces caractéristiques est fournie dans la fenêtre ci-contre.



Par exemple, la caractéristique *Fixed Type Label Value* permet de modifier la valeur utilisée dans l'option Batch by type du *Routing in*, pour sélectionner un item ayant une valeur donnée pour un label ; cette possibilité est au cœur du traitement de bons de commande de marchandises à prélever dans des files d'attente pour expédition au client (Synchronisation_stock_produits_Multiples.S8). La désignation des autres caractéristiques est assez explicite.

Chapitre VII

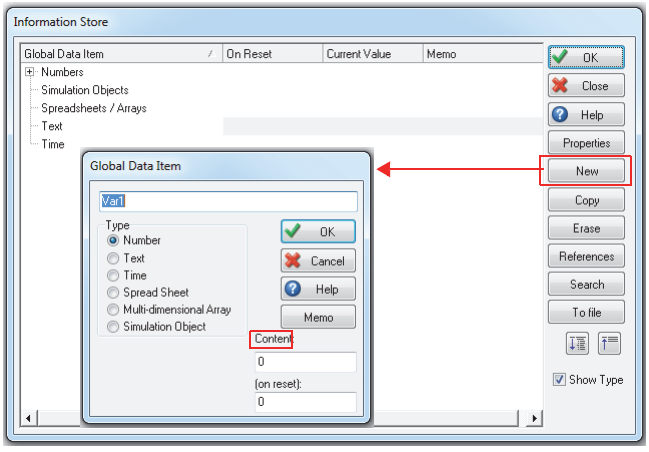
COMPLÉMENTS SUR LA MODÉLISATION SOUS SIMUL8

On commencera par approfondir l'utilisation de l'*Information Store* (§ I). On approfondira ensuite les principes d'utilisation de Visual Logic (§ II, page 101). On abordera enfin un quelques problèmes de modélisation complexes (§ III, page 109).

I *INFORMATION STORE*

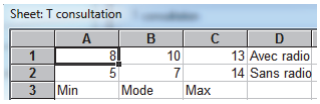
Après avoir expliqué comment sont initialisées les informations stockées dans l'*Information Store* (§ I-1), on s'intéressera à l'usage de ces informations dans la conception du modèle de simulation (§ I-2, page 100).

I-1 Initialisation des données de l'*Information Store*

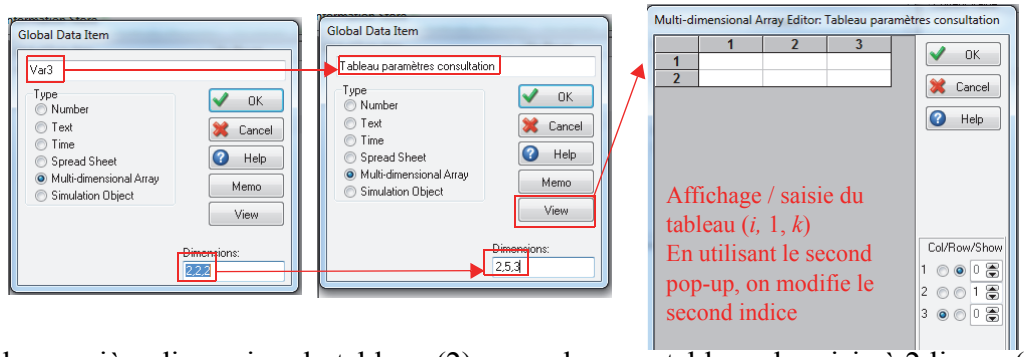


L'*Information Store* est un lieu de stockage d'un ensemble de paramètres et de tableaux exploitables par un modèle de simulation de Simul8. Un certain nombre de paramètres de la simulation sont obligatoirement fournis, comme *Simulation Time* qui indique depuis combien de temps la simulation est exécutée. Chaque paramètre et tableau portent un nom et doivent préalablement être déclarés par le bouton du menu Data_Rules. Les valeurs initialement contenues dans ces zones de stockage sont modifiables et accessibles tout au cours de la simulation par des programmes en *Visual Logic*.

- La déclaration d'une variable numérique se fait comme affiché ci-dessus. La valeur initiale peut être modifiée au cours de la simulation, la valeur courante de cette variable pouvant être examinée après arrêt de la simulation (Content, voir ci-dessus).
- De la même façon, on peut créer une variable littérale (Text) ou temporelle (Time) , ce qui présente un intérêt dans certains problèmes de modélisation.
- La déclaration de tableaux à 2 dimensions se fait par l'option Spread Sheet. Ce tableau peut être rempli dans la feuille de calcul affichée avec le bouton View. La copie d'écran illustre la création de ce type de tableau; on notera qu'ici les en-têtes de colonnes sont mis en dernière ligne, et celle des lignes, en dernière colonne. Ce type de tableau peut mélanger des informations de type numérique et de type littéral (voir l'exemple DOCTEUR_Information_Store_2.S8).



- La déclaration de tableaux à plusieurs dimensions se fait par l'option Multi-dimensional Array. Commençons par le cas d'un tableau à 3 dimensions, la généralisation à *n* dimensions étant alors évidente. Supposons, par exemple, qu'un patient admis aux urgences soit examiné deux fois par un urgentiste, que le temps de la consultation suive une distribution triangulaire dont les paramètres dépendent du niveau de gravité du patient à son admission. Le patient est donc caractérisé par un *label* «L Etape», initialisé à 1 et qui s'incrémente à chaque visite. La saisie des tableaux est relativement simple: en bas à droite de la fenêtre de saisie on a 2 colonnes de boutons radio; dans cet exemple le 3e bouton radio de la colonne 1 des boutons radio (*Col*) est validé, ce qui correspond à la 3e dimension du tableau (3) et le premier bouton de la colonne 2 des boutons radio (*Row*), ce qui correspond

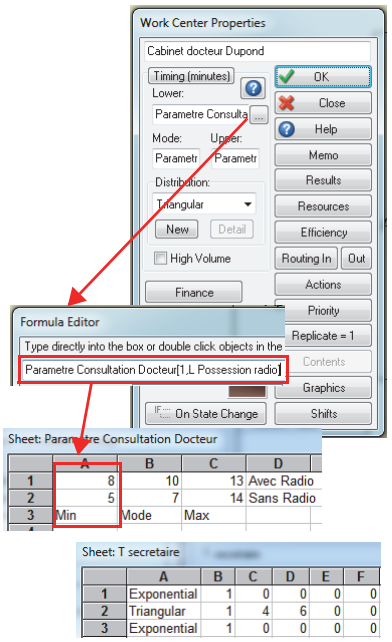


à la première dimension du tableau (2); on a donc un tableau de saisie à 2 lignes (étapes) et 3 colonnes (paramètres d'une distribution triangulaire); sur la seconde ligne des boutons radio, aucun bouton-radio n'est coché mais vous observez un pop-up avec le chiffre 1, ce qui correspond à la première occurrence de la dimension non retenue dans ce tableau, c'est-à-dire la gravité; le *pop-up* va de 1 à 5, ce qui permet de saisir successivement les 5 tableaux de paramètres. Si vous devez travailler avec un tableau à 4 dimensions, la généralisation de ce qui vient d'être décrit est immédiate, on saisit toujours un tableau à 2 dimensions, les occurrences des autres dimensions étant visualisées par 2 pop-ups sur les lignes où aucun bouton radio n'a été coché.

Les informations initialement saisies dans l'*Information Store* sont accessibles au cours de la simulation dans les programmes écrits en *Visual Logic* qui peuvent également les modifier. Ces variables et tableaux sont regroupés dans la partie «Information» de l'éditeur de formule (classement par ordre alphabétique). Attention, les valeurs courantes d'un tableau après l'arrêt d'une simulation sont reprises lors de la réinitialisation de la simulation (les variables, elles, sont réinitialisées à leurs valeurs définies lors de leurs créations). Dans certaines modélisations, il faudra prévoir l'exécution d'un programme en *Visual Logic*, avant le début de la simulation, pour effacer ou réinitialiser le contenu de certains tableaux (voir § II-3.2, page 108).

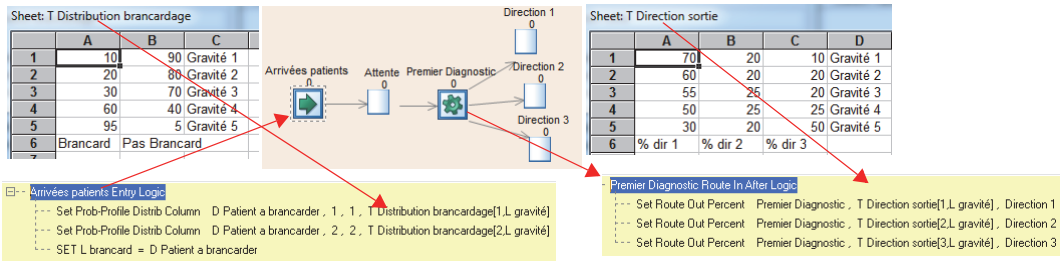
I-2 Utilisation de l'*Information Store* dans la conception du modèle de simulation

L'utilisation d'informations stockées dans l'*Information Store* présente deux avantages. Le premier est de permettre une paramétrisation du modèle de simulation lui permettant d'être plus concis et plus facile à maintenir. Le second avantage est de mettre à disposition de tous les programmes en *Visual Logic* susceptibles d'être exécutés au cours de la simulation, des informations partagées dont certaines portent sur des caractéristiques courantes des points de passage des items dans la modélisation, ce qui permet de déclencher des actions mettant sous contrôle le comportement global du système simulé. On verra cet usage au § II, page 101. Commençons par examiner les possibilités de paramétrisation offertes par l'*Information Store*.



Le premier intérêt est d'**éviter de multiplier les distributions de probabilités.** Commençons par celles de type ***Label based***, dès lors que la loi reste la même et qu'elle définit une variable quantitative, les paramètres de cette loi dépendant de la valeur prise par un (ou plusieurs) label(s) et étant stockables dans un tableau (Spread Sheet ou Multi-dimensional Array, si plusieurs *labels* sont nécessaires pour retrouver l'information). L'exemple DOCTEUR_Information_Store_0.S8 illustre cette possibilité de détermination de la loi du temps de travail d'un docteur, qui est une distribution triangulaire dont les paramètres dépendent du *label* «L Possession radio» (voir ci-contre). Il est possible, de la même façon, de définir une distribution Named Distribution ayant des paramètres lus dans un tableau de l'*Information Store*, en exploitant la valeur du *label* de l'item provoquant l'utilisation de cette distribution. Lorsque la loi ne reste pas la même, il faut définir la distribution et ses paramètres en passant par un programme en *Visual Logic*, en utilisant un tableau de paramètres stocké dans l'*Information Store*, ce qu'illustre le tableau ci-contre, utilisé pour déterminer le travail de la secrétaire dans notre exemple introductif dans la modélisation DOCTEUR_Information_Store_1.S8; l'exemple DOCTEUR_Information_Store_2.S8, est une variante de la solution précédente facilitant la généralisation du problème.

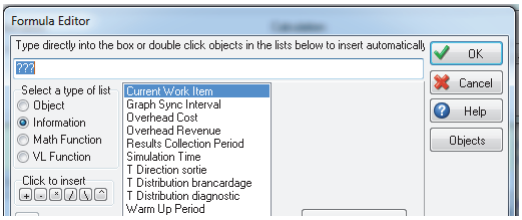
La ***paramétrisation des distributions des variables discrètes*** (Probability Profile) offerte par la lecture d'un tableau d'*Information Store* à partir de valeurs de *labels* et d'une programmation en *Visual Logic*, complète celle des distributions continues que l'on vient de voir. La démarche à suivre est décrite dans l'exemple Distribution_-Discrete_et_Information_Store.S8 dans lequel: a) les probabilités qu'un patient a d'être brancardé dépend du degré de gravité de son état en arrivant (*label* «L gravité»), informations stockées dans le tableau «T Distribution brancardage», ce qui permet une actualisation des paramètres de la distribution «D Patient à brancarder» par un programme *Visual Logic*; b) la direction de sortie de l'activité «Premier Diagnostic» est définie par l'option Percent, les probabilités d'emprunter une direction dépendant également de l'état de gravité et devant être mise à jour par un programme *Visual Logic*.



II PRINCIPES D'UTILISATION DE *VISUAL LOGIC*

La programmation sous *Visual Logic* permet de modéliser des processus plus complexes que ceux que l'on peut décrire par une simple combinaison des fonctionnalités proposées par les composantes de base de Simul8. Cette programmation est de type structuré ; elle s'appuie sur quatre classes d'instructions (§ II-2) et deux catégories d'information (§ II-1). On reviendra rapidement sur les événements susceptibles de déclencher l'exécution de programmes en Visual Logic (§ II-2).

II-1 Les informations exploitées par Visual Logic

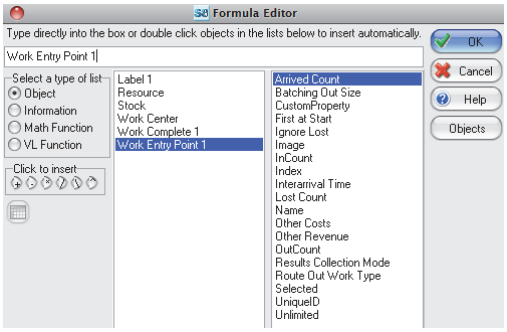


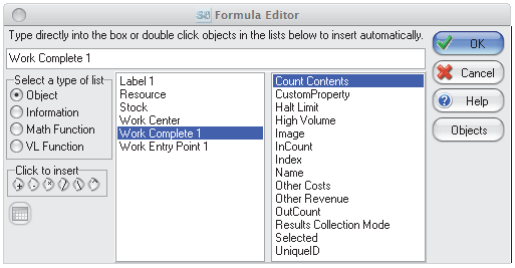
Un programme en *Visual logic* exploite deux catégories d'informations.

Il peut mobiliser des **informations partagées** (Information), les premières étant disponibles par construction, comme le temps écoulé depuis le début de la simulation (*Simulation Time*) et les autres correspondant à des variables ou des tableaux créés dans l'*Information Store* pour pouvoir simuler correctement le processus modélisé. Les informations initialement saisies dans l'*Information Store* sont accessibles au cours de la simulation dans les programmes écrits en *Visual Logic* qui peuvent également les modifier. Ces variables et tableaux sont regroupés dans la partie «Information» de l'éditeur de formule (voir ci-dessus, classement par ordre alphabétique). Attention, les valeurs courantes d'un tableau après l'arrêt d'une simulation sont reprises lors de la réinitialisation de la simulation (les variables, elles, sont réinitialisées à leurs valeurs définies lors de leurs créations).

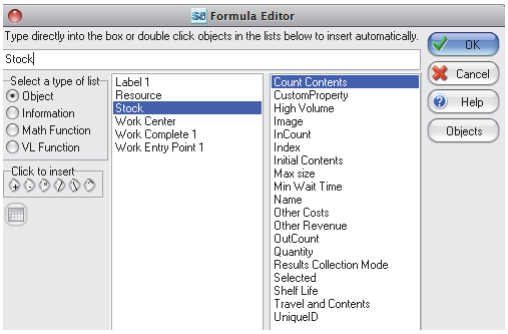
Un programme en *Visual logic* peut aussi utiliser des **propriétés des objets** créés pour modéliser le processus (Object), à savoir les points de passage des items dans le système productif (points d'entrée et de sortie, files d'attente et activités), les ressources, les *labels* et les distributions de probabilités. Nous avons déjà détaillé les propriétés des points de passage que l'on peut mobiliser dans les instructions écrites en Visual Logic ; faisons une rapide synthèse.

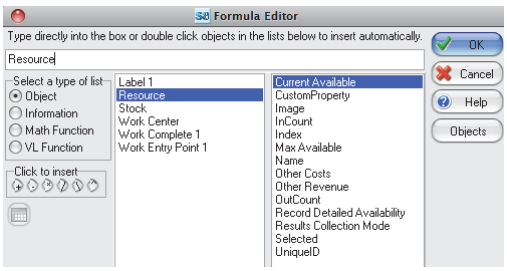
- La description des propriétés de la classe «Point d'entrée» est faite à la page 60; celle de la classe «Point de sortie» est faite à la page 61.



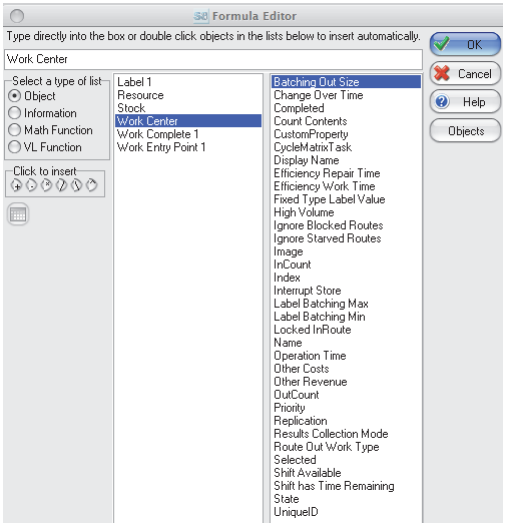


- La description des propriétés de la classe «file d'attente» est faite à la page 67.
- La description des propriétés de la classe «Ressources» est faite à la page 70.





- La description des propriétés de la classe «activités» est faite à la page 97.



Une instruction en *Visual Logic* peut mobiliser la valeur d'un *label* associée à un item. Dans un programme en *Visual Logic*, l'item peut être celui qui est traité par une activité ou l'un des items pointés dans une file d'attente[1]. Il peut être aussi un item pointé dans une autre file d'attente[2].

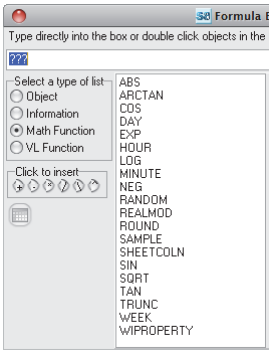
Enfin, une instruction en *Visual Logic* peut mobiliser des **distributions de probabilités**, normalement dans le second membre d'une instruction d'affectation d'une valeur (occurrence d'une variable aléatoire) à un *label* (Set to). Les distributions disponibles ont été présentées en détail aux pages 35 et suivantes.

II-2 Les instructions de *Visual Logic*

Comme dans tout langage de programmation, on y retrouve des opérateurs mathématiques (+, -,…) des fonctions mathématiques et un ensemble d'instruction permettant une programmation structurée.

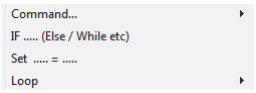
1. Ce cas est illustré dans l'exemple Initialisation_Stock_deterministe.S8.
2. Ce cas est illustré dans l'exemple Synchronisation_stock_produits_Multiples.S8, page 116.

II-2.1 Fonctions mathématiques de *Visual Logic*



Les seules fonctions vraiment utiles sont celles qui permettent le repérage temporel dans une unité différente de celle utilisée par Simulation time (par défaut la minute). Ces fonctions, HOUR, DAY et WEEK ont toutes pour argument Simulation Time et se réfèrent à un cycle de base : HOUR[Simulation Time] convertit Simulation Time en un numéro d'heure d'une journée de 24 heures, la première heure étant 0 et la dernière 23 ; DAY[Simulation Time] convertit Simulation Time en un numéro de jour d'une semaine de 7 jours, le lundi étant le jour 1 de la semaine et le dimanche, le 7e jour de la semaine. Un exemple d'utilisation de ces fonctions peut être trouvé avec Distribution_Time_Dependant_Jour_semaine_IS.S8. et Distribution_Time_Dependant_partition.S8 (utilisée pour assigner à un *label* le numéro de plage de temps d'une distribution *Time dependant*, pour faciliter l'analyse des résultats).

II-2.2 Instructions de programmation structurée



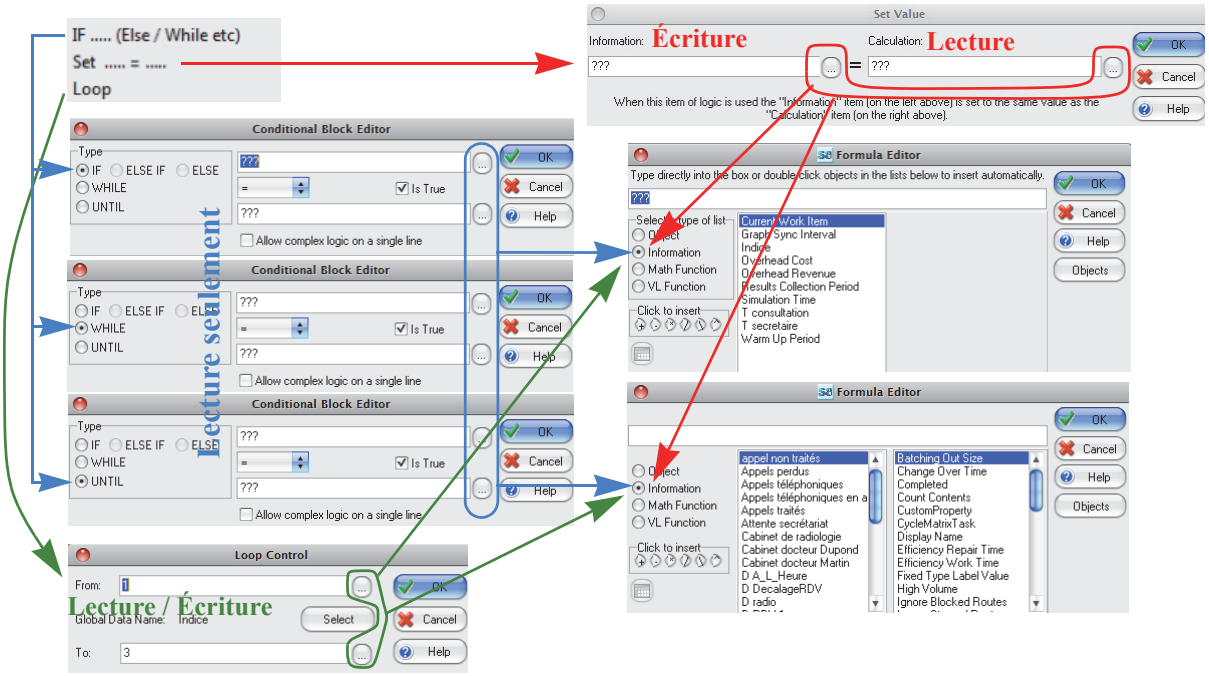
Visual Logic offre les instructions de base de toute programmation structurée, avec la possibilité d'assigner une valeur à un paramètre ou un *label* (instruction «SET…»), de conditionner cette assignation à une condition logique (instruction «IF… THEN.») et d'exécuter une ou plusieurs instructions dans le cadre d'un processus itératif (instruction «LOOP…»[1] ou «WHILE»[2]). Enfin, il est possible d'utiliser directement certaines commandes (Command) de *Visual Logic* pour intervenir directement sur des caractéristiques du processus modélisé qui ne sont pas représentées par des propriétés des points de passage (voir § II-2.3, page 104).

Le choix de l'une des instructions de base, dans le menu ouvert par un clic-droit sur la fenêtre d'écriture d'un programme, ouvre une fenêtre permettant de spécifier les éléments qui interviennent dans l'écriture d'une instruction. Cet élément peut être une information partagée (§ I-2, page 100) ou une propriété d'un point de passage, un *label* ou une distribution de probabilités (page 35) ou une expression faisant intervenir un ou plusieurs de ces éléments avec utilisation de fonctions mathématiques (§ II-2.1). Certains de ces éléments ne peuvent être utilisés qu'en

1. Cette instruction utilise nécessairement un indice préalablement déclaré dans l'*Information Store. L'instruction loop sert à exécuter plusieurs fois un ensemble d'instructions.* Dans l'exemple Peremption_variable.S8, un programme VL est exécuté toutes les 0,1 minute, il examine un à un les articles mis en file d'attente et change la valeur d'un *label* si la date courante (*Simulation Time*) est postérieure à une date limite de séjour en file d'attente (générée à l'entrée pour chaque item). Vous trouverez une autre application de LOOP dans DOCTEUR_Information_Store_2.S8, où un programme VL modifie la distribution du temps de traitement de la secrétaire en fonction de l'étape du client dans le processus (rendant inutile l'utilisation de distribution de type *Label Based*). Enfin, dans Stock_&_Information_Store, l'instruction LOOP est utilisée pour réinitialiser un tableau des positions de stock d'items pour les différentes valeurs possibles prises par un *label* particulier, consignées dans l'*Information Store*.

2. L'exemple Generation_Bon_de_Commande.S8 illustre l'utilisation combinée des instructions LOOP et WHILE pour définir le contenu de plusieurs lignes de bons de commande en s'assurant qu'une même référence n'est pas commandée sur deux lignes différentes d'un même bon de commande. L'exemple Generation_et_Traitement_BC.S8 reprend l'exemple précédent en le complétant par le processus de préparation de la commande à livrer.

lecture (on ne peut pas modifier le nombre d'items en file d'attente à un moment donné); d'autres sont utilisables en lecture et écriture (voir page suivante).

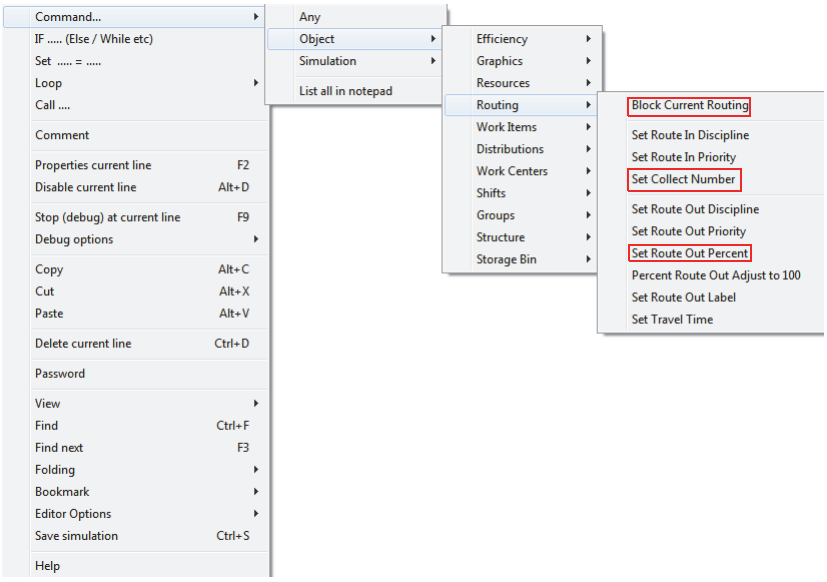


II-2.3 Les instructions *Command*

Les spécificités de la modélisation et de la simulation d'un processus productif ne peuvent être toutes prises en compte par les propriétés des points de passage d'un item. On peut vouloir, par exemple, modifier les caractéristiques d'une distribution statistique, bloquer l'accès à une activité ou modifier les coefficients d'une méthode de sélection Collect retenue pour une activité.

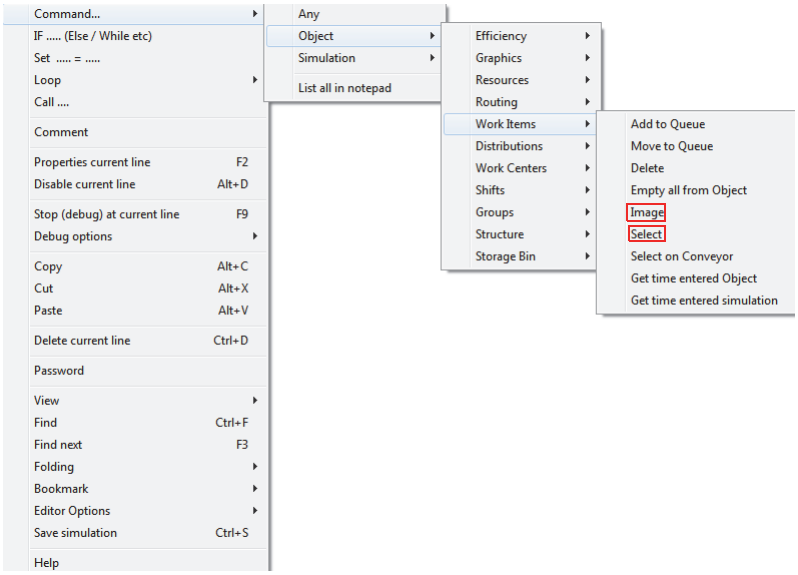
Ces instructions Command sont structurées de manière hiérarchique. On n'examinera pas toutes les classes de commande, on n'en examinera que celles que nous avons eu l'occasion d'utiliser. L'approche hiérarchique est parfois agaçante; si vous savez quelle commande mobiliser, choisissez *Any* qui ouvre un menu déroulant affichant toutes les commandes disponibles.

Dans la classe *Object > Routing*, nous avons eu l'occasion d'utiliser trois commandes :



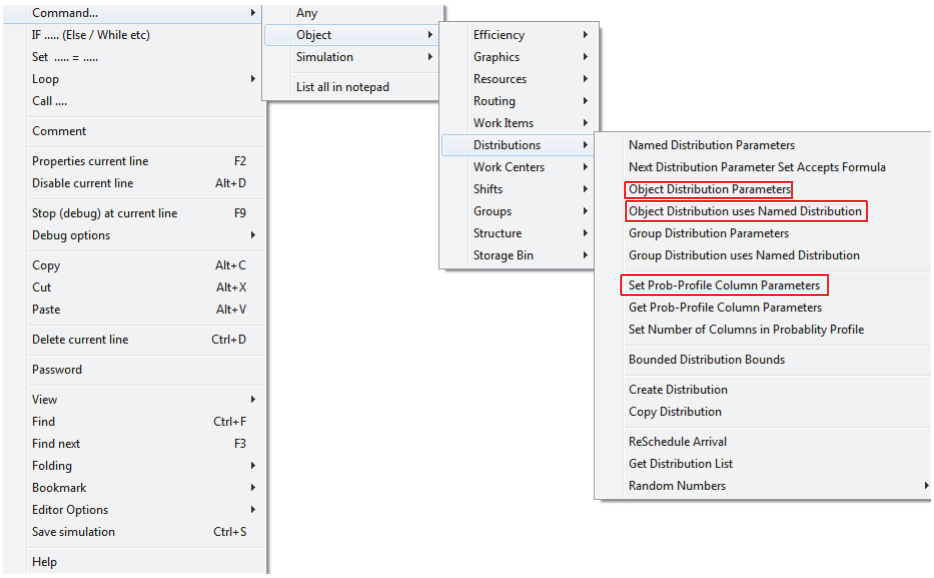
- La commande *Block Current Routing* est utilisée pour bloquer l'accès à l'activité qui lance l'exécution du programme (normalement en *Routing In, Before Selecting*, voir page 96) ; en général cette instruction est utilisée pour synchroniser un flux d'information et un flux de produits (prélèvement de produits dans une file d'attente, conditionné par l'existence préalable d'une commande). Cette commande est utilisée, notamment, dans Synchronisation_stock_produits_Multiples.S8 (page 116).
- La commande *Set Route Out Percent*, utilisée pour modifier les probabilités des directions en sortie d'une activité est utilisée dans Distribution_Discrete_et_Information_Store.S8.
- La commande *Set Collect Number*, utilisée pour modifier les nombres d'items prélevés dans plusieurs files d'attente par une activité est utilisée dans les exemples Routing_in_Collect_and_Match_2.S8 et Synchronisation_stock_produits_Multiples_Collect.S8.

Dans la classe *Object > Work Items*, nous avons eu l'occasion d'utiliser deux commandes :



- La commande Select permet de sélectionner l'item (normalement détenu dans une file d'attente) se trouvant en position k ; on dispose alors de l'ensemble des valeurs des *labels* de cet item (attention, si cette commande est exécutée dans une activité, les informations relatives à l'item en cours de traitement dans l'activité ne sont plus accessibles). Souvent cette instruction est utilisée avec $k = 1$ (voir, par exemple Synchronisation_stock_produits_Multiples.S8, page 116) mais ce n'est pas nécessairement le cas (voir, par exemple, Arrivees_stochastiques_clients_connus.S8, où un client est tiré aléatoirement dans un fichier de clients).
- La commande *Image* a été utilisée dans l'exemple Icone_Item_en_sortie.S8.

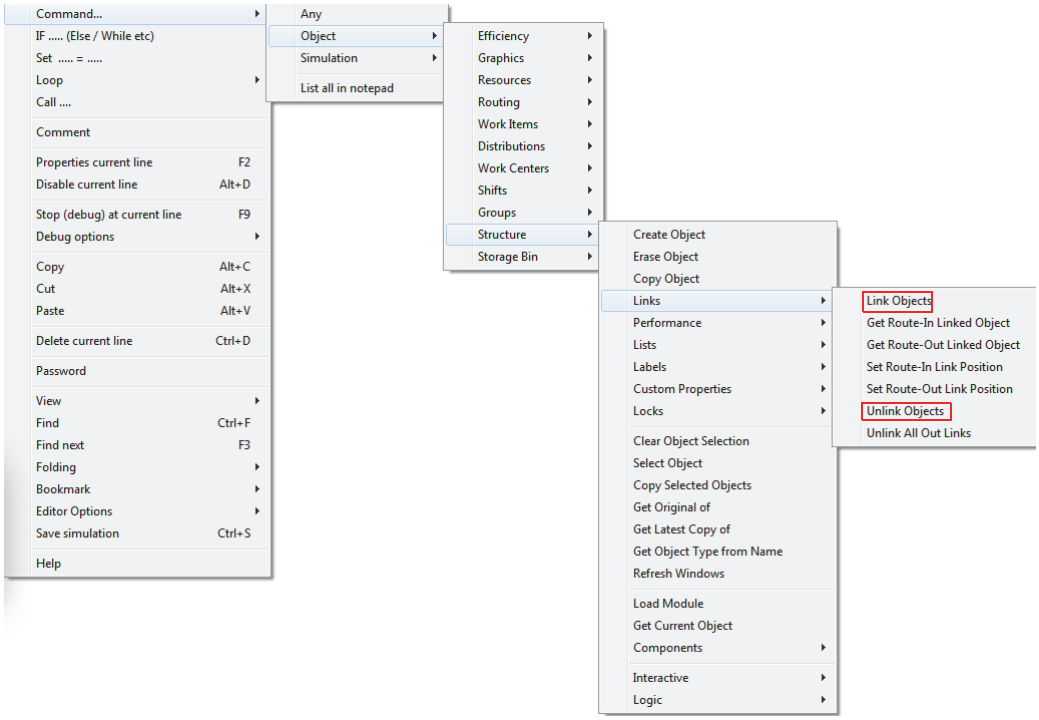
Dans la classe *Object > Distributions*, nous avons eu l'occasion d'utiliser trois commandes :



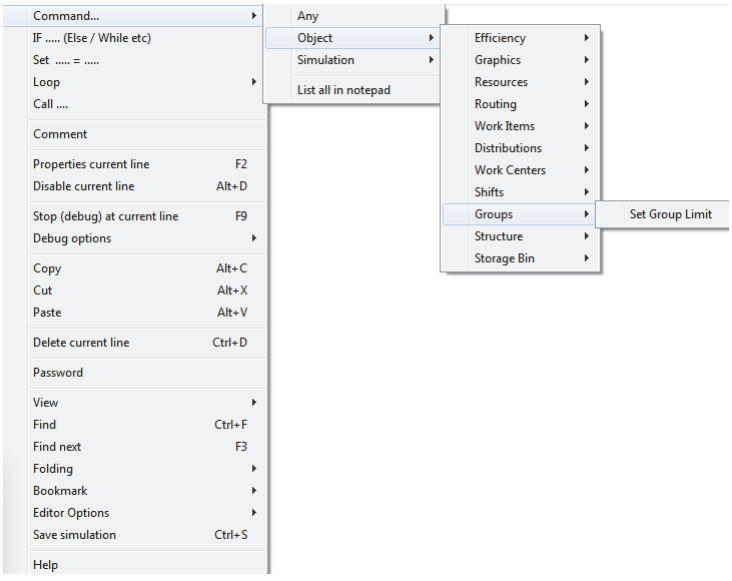
- La commande *Object Distribution uses Named Distribution* est utilisée dans l'exemple Distribution_Time_Dependant_Jour_semaine.S8.

- La commande *Object Distribution Parameters* est utilisée dans l'exemple DOCTEUR_Information_Store_1.S8, généralisé dans l'exemple DOCTEUR_Information_Store_2.S8.
- La commande *Set Prob Profile Column Parameters* est utilisée dans l'exemple Routing_out_Label_3.S8 pour paramétrer une distribution de type *Probability Profil* en fonction de la valeur prise par un *label* (avec utilisation d'un tableau d'*Information Store*).

Dans la classe *Object > Structure*, nous avons eu l'occasion d'utiliser les commandes *Link* et *Unlink* dans l'exemple Work_Center_Shifts_Stock_in_individuel_2.S8.



Dans la classe *Object > Groups*, nous avons eu l'occasion d'utiliser la seule commande proposée dans l'exemple Synchronisation_stock_produits_Multiples.S8, page 116.



II-3 Le déclenchement de programmes en *Visual Logic*

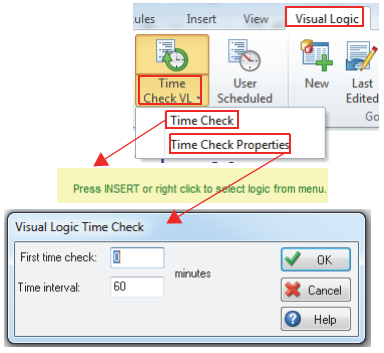
Un programme en Visual Logic peut être déclenché par un événement lié à l'utilisation d'un objet de base de la modélisation; il peut aussi être déclenché, indépendamment, à intervalle régulier.

II-3.1 Événements liés à l'utilisation d'un objet de base de la modélisation

Nous avons, dans les chapitres précédents, listé les événements susceptibles de déclencher un programme en *Visual Logic*.

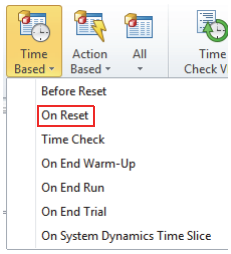
- Une possibilité au **point d'entrée** (Actions | Visual Logic); ce point est détaillé à la page 59.
- Trois possibilités dans une **file d'attente**, lors de l'entrée d'un item (VL On Entry) ou de sa sortie (VL On Exit et VL On Expire); ce point est détaillé à la page 67.
- Deux possibilités pour une **ressource**: lors de sa mobilisation (VL On Hold for Work) ou de sa libération (VL On Release); ce point est détaillé à la page 70.
- Six possibilités pour une ***activité***: deux possibilités au niveau du Routing In | Options (Before Selecting et After Loading Work); une dans la fenêtre du Actions (Visual Logic); trois dans la fenêtre du Out (On Work Complete, Before Exit et On Exit); ce point est détaillé à la page 96.

II-3.2 Événements liés au temps (*Visual Logic Time Check*)



Dans la simulation à événements discrets, le temps s'accroît de manière incrémentale en passant, à la fin d'un événement, au début de l'événement à venir se produisant au plus tôt dans la liste des événements à venir. On peut définir des événements se produisant à intervalle régulier (voir fenêtre), indépendants de ce qui passe dans la simulation, afin de pouvoir exécuter un programme en *Visual Logic*. Cette possibilité, introduite à la page 93, permet, d'**analyser périodiquement l'état du système** avant de prendre une décision. Trois exemples illustrent l'intérêt de ce type de problématique.

- L'exemple Peremption_variable.S8 montre comment marquer des items ayant dépassé un temps maximal de séjour dans une file d'attente, cette durée variant d'un item à l'autre, ce programme étant exécuté toutes les dixième de minute.
- L'exemple Work_Center_Shifts_Stock_in_individuel_2.S8 montre comment décider de l'ouverture ou de la fermeture d'activités en fonction d'une disponibilité variable au cours du temps d'une ressource partagée entre ces activités.
- L'exemple Decision_periodique.S8 montre comment on peut ajuster la capacité du système productif en fonction de la saturation: le système productif comporte 3 activités identiques et une 4e un peu moins performante; au démarrage de la simulation puis toutes les 30 minutes le système productif est analysé et si le cumul des items en file d'attente dépasse 10 unités, la 4e activité est mise en service pour la demi-heure suivante, sinon, elle est arrêtée.



Le tout début de la simulation joue un rôle tout d'abord, elle permet de **réinitialiser**:des tableaux mis à jour dynamiquement dans une simulation. Pour ce faire, on utilise l'option On Reset (voir ci-contre) dans l'onglet Visual Logic (Visual Logic) de la barre de menu principal pour créer le programme en Visual Logic. Notons qu'il est possible d'obtenir le même résultat en créant ce programme avec l'option Time Check, en conditionnant l'exécution du programme à la condition IF Simulation Time = 0 .

- On peut réinitialiser à zéro des **données** stockées dans l'***Information Store***, lorsque celles-ci sont susceptibles d'être modifiées au cours de la simulation. L'exemple Stock_&_Information_Store.S8 illustre ce type d'utilisation. La commande Clear Sheet permet d'effacer tout le contenu d'un tableau de l'*Information Store*; la syntaxe de cette commande étant Clear Sheet Nom_du_tableau[1,1]. La commande Clear Sheet Area permet d'effacer une partie du contenu d'un tableau de l'*Information Store*; la syntaxe de cette commande étant Clear Sheet Area Nom_du_tableau[x,y], z, u, où définissent les coordonnées de l'angle supérieur gauche du sous-tableau à effacer et z et u définissent le nombre de colonne et de lignes de ce sous-tableau à effacer; cette instruction permet de conserver des intitulés lignes et colonnes d'un tableau rempli dynamiquement en cours d'une simulation.
- On peut également réinitialiser à des seuils et valeurs imposés des **files d'attente** lorsque ceux-ci possèdent au démarrage de la simulation un ensemble d'items de caractéristiques définies dans un tableau de l'*Information Store,* ce qu'illustre l'exemple Initialisation_Stock_deterministe.S8 étudié à la page 65.

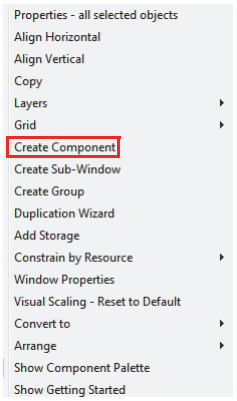
III COMPLÉMENTS SUR LA MODÉLISATION DE PROCESSUS

On commencera par examiner des objets complémentaires utilisés dans la modélisation, avant d'examiner le traitement des gammes dans la modélisation (§ III-2, page 113) et de terminer par quelques remarques importantes sur la façon d'aborder la modélisation de certains systèmes complexe (§ III-3, page 115).

III-1 Objets complémentaires de modélisation mobilisables

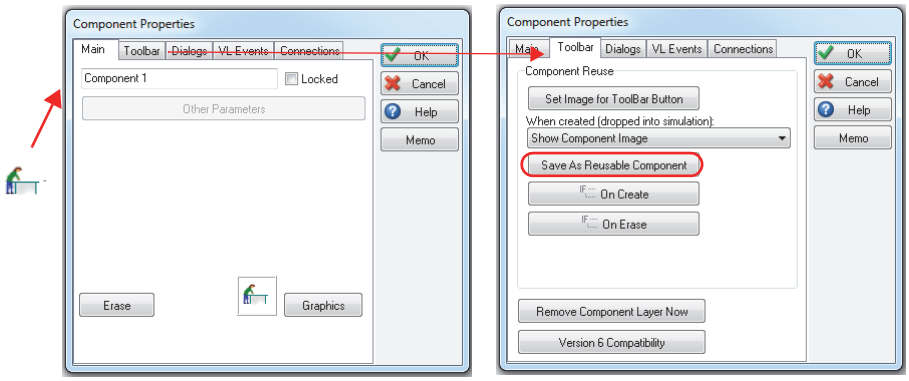
Nous ne reviendrons pas sur le concept de groupe, ensemble de files d'attente et activités, introduit à la page 63 et utilisé dans les exemples Synchronisation_stock_produits_Multiples.S8 (présenté en détail à la page 117) et Open_shop.S8. On analysera ici les composants (§ III-1.1) et les activités fictives qui jouent un rôle bien particulier dans la modélisation des processus sous Simul8 (§ III-1.2, page 111).

III-1.1 Création et utilisation de composants

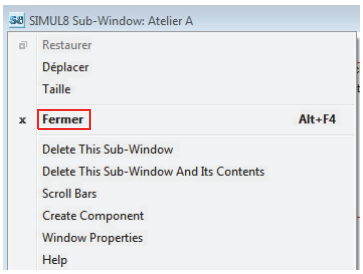


Un **Composant** (*component*) est un ensemble interconnecté d'activités et de files d'attente, sélectionnées pour constituer un tout jouant un rôle précis dans la simulation (c'est donc un **sous-processus**) à des fins diverses non exclusives: simplification du modèle permettant de mieux comprendre la circulation des items (**approche hiérarchique** des processus), gain de place, désir de masquer des activités fictives, facilité de duplication. La sélection de l'ensemble s'effectue en positionnant le pointeur de la souris à un endroit de la fenêtre puis et en déplaçant ce pointeur pour sélectionner l'ensemble voulu. Ensuite, on utilise le clic droit de la souris, ce qui fait apparaître un menu ci-contre. On choisit alors l'option Create Component qui remplace la sélection par l'icône (dont on peut modifier l'image). Ce composant doit être dupliqué en faisant appel à l'option *Duplication Wizard* de l'onglet Home, pour conserver les liens, toutes les caractéristiques des composants dupliqués et pour adapter automatiquement, dans les programmes de Visual Logic dupliqués, les désignations des files d'attente et activités de l'ensemble dupliqué et utilisées dans ces instructions. Voir l'exemple Duplication_composant.S8.

On peut également, après un double-clic sur l'icône, l'enregistrer dans la bibliothèque des «objets» de la simulation et de l'avoir à disposition sur la fenêtre (Component Properties, onglet Toolbar, bouton Save As Reusable Component); il est possible également de le mettre dans la bibliothèque générale de Simul8 afin de pouvoir l'utiliser dans d'autres simulations.



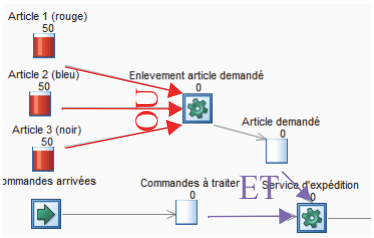
Pour visualiser le contenu d'un composant dans une fenêtre, il suffit d'un double-clic gauche sur l'icône du composant. Pour fermer ensuite cette fenêtre, il faut cliquer sur le bouton qui se trouve en haut à gauche de la fenêtre, ce qui provoque l'affichage d'un certain nombre d'actions possibles, parmi lesquelles Fermer. L'option Delete This Sub-Window élimine la fenêtre et réintègre ses composants dans la fenêtre principale de la modélisation. L'exemple Service_des_urgences.S8 simule le fonctionnement d'un service d'urgence; il utilise un composant appelé box, intégrant de nombreuses files d'attente et activités et permettant d'avoir une vision plus globale du fonctionnement du système productif.



III-1.2 Activité fictive

Nous avons utilisé à plusieurs reprises des activités fictives qui sont des activités ayant un temps opératoire nul, que l'on utilise pour résoudre des problèmes de modélisation que l'on ne peut pas traiter directement avec les composants de base de la modélisation. Les activités, que l'on peut qualifier de «réelles», ont des temps opératoires positifs et effectuent un traitement sur l'item qu'elles accueillent. Ces activités fictives peuvent se trouver, soit en alimentation d'une activité réelle (avec, généralement, une file d'attente intermédiaire entre l'activité fictive et l'activité réelle), soit en destination d'une activité réelle. Examinons successivement ces deux cas de figure.

III-1.2.1 Activité fictive pour résoudre un problème d'alimentation d'une activité réelle



Sous Simul8, les méthodes de sélection des items par une activité réelle, ont été décrites aux pages 72 et suivantes et résumées dans le tableau 3 de la page 74. Lorsqu'une activité est connectée à plusieurs files d'attente, ces méthodes de sélection relèvent, d'une manière exclusive, soit d'une logique de type **OU**, soit d'une logique de type **ET**. En logistique, il est fréquent d'avoir des files d'attente banalisées dans lesquelles une même référence peut se trouver dans plusieurs files d'attente, le nombre de références étant sans aucune mesure avec le nombre de files d'attente. L'arrivée, dans une file d'attente (dans le schéma, «Commandes à traiter»), d'un bon d'enlèvement (ou équivalent) portant sur une référence de produit conduit à prélever une ou plusieurs unités de cette référence qui peut se trouver dans un ou plusieurs stocks (dans le schéma, «Article 1 (rouge)», «Article 2 (bleu)», «Article 1 (noir)»). On est alors en présence d'une combinaison d'une logique de type **OU** avec une logique de type **ET**, ce qu'illustre le schéma ci-dessus. La manière de traiter ce problème sera détaillée au § III-3.1.1, page 116.

Par ailleurs, on peut ajouter qu'une activité fictive est nécessaire pour lancer en production des items dont la gamme est explicite (*jobs matrix*, voir page 114 et l'exemple Routing_Job_Matrix.S8) et donc pour alimenter la file d'attente située en amont de la première activité réalisant la première opération de la gamme de l'item considéré.

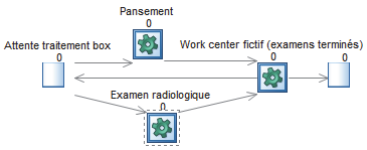
III-1.2.2 Activité fictive pour résoudre un problème d'orientation en sortie d'une activité réelle

L'activité fictive située après une activité réelle permet de définir l'orientation d'un item obéissant à une règle plus complexe que les règles de base implantées dans le *Routing Out* des activités. Sans prétendre à l'exhaustivité, on peut signaler les quatre cas suivants, dans lesquels l'item sortant peut être envoyé dans plus d'une file d'attente.

- La **destination en sortie** d'une activité (ou d'un point d'entrée) peut être **unique** et choisie parmi un ensemble de destinations possibles à partir des

informations définies par **deux labels (ou plus)** correspondant à des caractérisations intrinsèques de l'item. Une première solution consiste à définir, par un programme en *Visual Logic* la valeur pertinente de label définissant le numéro de destination. Il est plus rapide de passer par des activités fictives pour effectuer en cascade ces tests d'orientation. L'exemple Routing_out_2_Labels.S8 reprend celui de la définition de la destination sur valeur d'un *label* (Routing_out_Label_1.S8, page 87), en ajoutant un second *label* «type de client» (valant 1 ou 2); pour trier en sortie sur 6 files d'attente sur la combinaison de ces 2 labels.

- La **destination en sortie** d'une activité (ou d'un point d'entrée) peut être **multiple** et à choisir dans un **sous-ensemble** de destinations pris dans un ensemble fini de destinations possibles, ce sous-ensemble dépendant soit de la valeur d'un label possédé par l'item sortant (et correspondant donc à une caractéristique intrinsèque de l'item), soit d'une information extérieure accessible (par exemple, l'effectif disponible d'une ressource), indépendante des caractéristiques propres de l'item. Le premier cas (sous-ensemble dépendant d'une caractéristique intrinsèque d'un item) est illustré par l'exemple PoliceAeroport.S8 décrivant l'organisation (totalement déconnectée de la réalité) des formalités de police des frontières pour les passagers venant de vols internationaux dans un aéroport français; dans ce cadre, 3 types de guichets existent (guichets A pour les citoyens français, guichets B pour les citoyens de l'UE non français et guichets C pour les citoyens d'un pays n'appartenant pas à l'UE) et les citoyens français ont accès à tous les guichets, les citoyens de l'UE non français ont accès aux guichets B et C et que les citoyens d'un pays n'appartenant pas à l'UE n'ont accès qu'aux guichets C. Le premier cas (sous-ensemble dépendant d'une information extérieure) est illustré par l'exemple Work_Center_Shifts_Stock_in_individuel_1.S8 dans lequel, la valeur du label utilisé pour définir la direction en sortie ne dépend pas des caractéristiques «intrinsèques» de l'item mais du nombre d'opérateurs disponibles au moment où l'item quitte le point d'entrée; l'item est alors dirigé vers une activité fictive qui donne accès au sous-ensemble pertinent de files d'attente.
- La modélisation d'un problème d'ordonnancement de type **open shop**, se pose lorsque plusieurs traitements doivent être exécutés sur un item, sans ordre particulier entre ces traitements. La solution consistant à imposer un ordre entre ces opérations n'est pas judicieuse et conduit à une mauvaise utilisation des ressources.

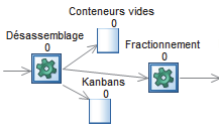


Supposons, voir ci-contre, qu'un patient se trouvant dans un box d'un service d'urgence se soit vu prescrire un pansement et une radio à réaliser avant de revoir le médecin; l'ordre dans lequel ces opérations seront effectuées dépend seulement de la disponibilité des ressources mobilisées. Pour résoudre ce problème, on crée trois labels: un pour chaque examen pour savoir s'il a été réalisé ou non et un pour savoir si les deux ont été réalisés, information qu'exploitera une activité fictive. Une solution facile d'implémentation et difficile à contourner dans des problèmes complexes d'*open shop* consiste à connecter une activité fictive en sortie des activités réelles et à y réaliser le test permettant de garder le patient dans le box ou à le faire sortir du box, ce qu'illustre l'exemple Open_shop.S8.

- L'utilisation d'une activité fictive s'impose également lorsqu'on désire émettre un **lot en sortie** d'une activité ou d'un point d'entrée, dans lequel on veut que tous les items possèdent une même valeur pour un label et que cette valeur est définie comme l'occurrence d'une variable aléatoire. En effet, lorsqu'un lot est défini en sortie, les différentes actions commandées par le

bouton Actions sont exécutées indépendamment pour chaque item de ce lot; il s'ensuit que les valeurs prises pour ce label dans le lot n'ont aucune chance d'être identique. Pour garder la même valeur, il suffit de déplacer l'opération d'éclatement sur une activité fictive liée à l'activité définissant ces labels. Un exemple de cette solution a été traité à la page 57 (voir l'exemple Arrivee_items_par_groupe.S8).

- L'utilisation d'une activité fictive s'impose enfin lorsqu'on désire émettre des lots de tailles différentes dans les différentes directions définies en *Routing out*. L'exemple Routing_out_Desassembler.S8 illustre la solution à adopter pour résoudre un problème de juste-à-temps (kanban).



III-1.2.3 Activité fictive et purge d'une file d'attente

Pour terminer, évoquons une dernière utilisation d'une activité fictive intervenant en assistance non pas d'une activité réelle mais d'une file d'attente. Il a été déjà souligné que les items ne pouvaient «bouger» dans un système productif que parce qu'ils sont poussés ou tirés par des activités (ou poussés par des points d'entrée). Les files d'attente sont des points de passage passifs: ils ne peuvent pas attirer ou expulser les items. C'est dans ce contexte que se pose le problème de l'élimination dans une file d'attente, des items qui y ont séjourné trop longtemps. La solution de ce problème, présenté à la page 64 et illustré par les exemples Peremption_fixe.S8 et Peremption_variable.S8, passe par l'utilisation d'une activité fictive pour retirer de la file d'attente, les items concernés et les envoyer vers un point de sortie ou dans une file d'attente réunissant les items périmés (cette seconde solution permettant une analyse plus fine du comportement du système productif).

III-2 Le traitement des gammes de production dans la simulation

La gamme, associée à la production d'un bien ou d'un service, se définit par une liste d'opérations à exécuter dans un certain ordre (mais pas toujours), chaque opération devant être réalisée dans un centre de production ayant des caractéristiques précises et mobilisant éventuellement certaines ressources. Deux systèmes sont mobilisables avec Simul8, le premier gère les gammes de manière implicite (§ III-2.1); le second introduit la gamme de manière explicite, comme un «objet de simulation» complémentaire (§ III-2.2, page 114).

III-2.1 Gammes implicites

Dans toutes les modélisations étudiées jusqu'ici, les gammes étaient sous-jacentes dans la circulation des items entre leurs arrivées dans le système productif, jusqu'à leurs départs.

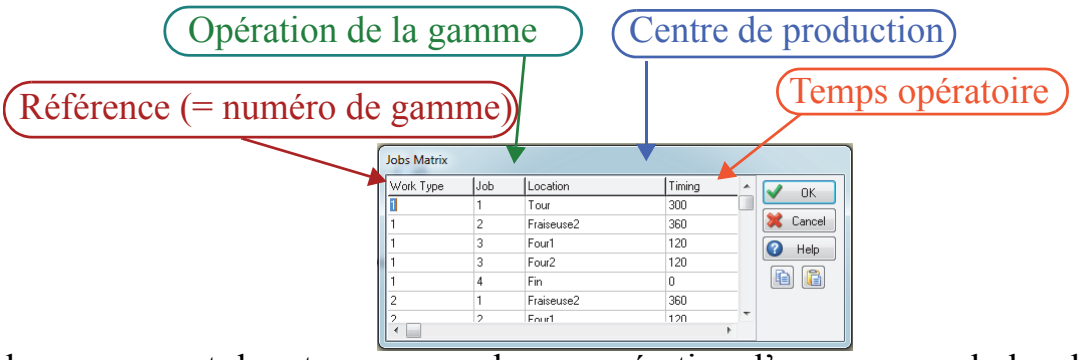
L'exploitation du **graphe** de modélisation d'un système productif, en partant du (ou des) point(s) d'arrivée d'un item, pour aller jusqu'à un (ou plusieurs) point(s) de sortie et passant par une succession de files d'attente et d'activités permet implicitement de retrouver toutes les séquences possibles d'opérations définissant une gamme de production. Cet ensemble est généralement **restreint** par l'exploitation de la **caractérisation d'un item** par ses valeurs des labels qu'une *activité* exploite avec ses règles de sélection de l'item à traiter ou de détermination de la

file d'attente dans laquelle il sera envoyé après traitement. Le temps d'exécution d'une opération dans une activité peut avoir des caractéristiques soit partagées par l'ensemble des items qu'il traite (type de loi et paramètres de cette loi), soit spécifiques à l'item (liées aux valeurs de certains de ses labels).

Dans cette première approche, la gamme est implicite et ses informations réparties dans la description du système productif par son graphe et ses composantes complétées par celle des items. La modification, au cours de la simulation, de la valeur prise par certains labels (directement ou par des instructions en *Visual Logic*) permet de prendre en compte le caractère non complètement prédéterminé de certaines gammes (reprise d'opérations pour des raisons de qualité, gammes alternatives…).

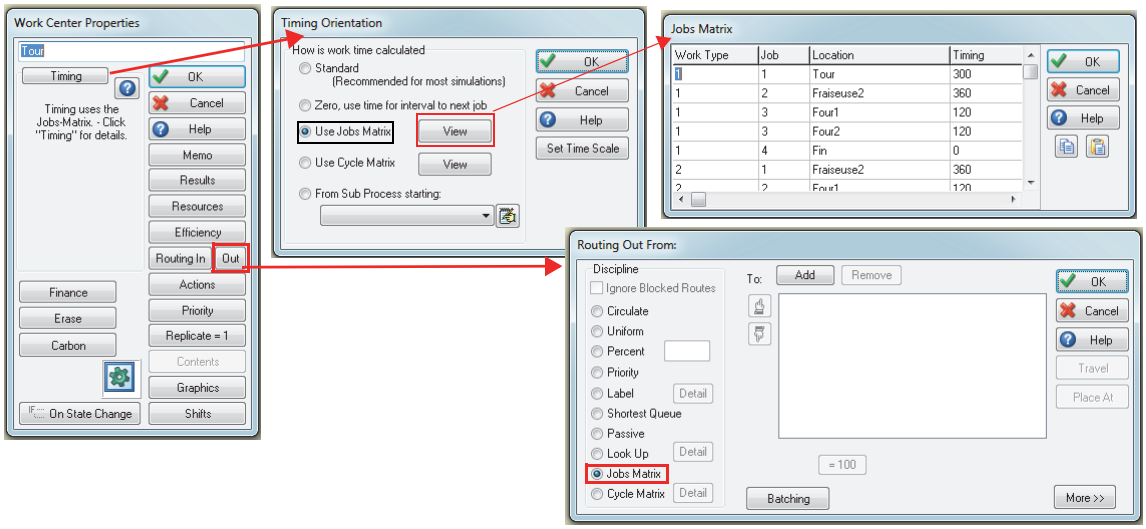
III-2.2 Gammes explicites

Les simulateurs peuvent aussi utiliser des gammes explicites qui prennent la forme de tableaux associant à une référence une liste ordonnée d'opérations avec, sur chaque ligne: le poste de travail (activité) unique ou appartenant à un ensemble de postes de travail substituables, sur laquelle l'opération doit être exécutée et sa durée d'exécution (ou la distribution de probabilités de cette durée). Sous Simul8, cette explicitation de gamme se fait dans un tableau géré par le bouton **jobs matrix** dans l'onglet Data_Rules menu général. Ce tableau à quatre



colonnes permet de retracer pour chaque opération d'une gamme, la localisation du centre de production effectuant cette opération et le temps opératoire (éventuellement défini par une distribution définie à partir d'un double-clic dans la cellule). Si une opération de la gamme peut être effectuée dans plusieurs postes de travail (activités), on aura autant de fois dans le tableau le même couple «gamme - opération» (attributs *work item* et *job*, créés automatiquement lors de l'usage de l'option *jobs matrix*). La gamme peut être identifiée par un simple numéro d'ordre ou par un libellé; les opérations d'une gamme sont nécessairement repérées à partir d'un numéro d'ordre, la première opération d'une gamme portant le numéro 1. Toutes les opérations d'une gamme sont nécessairement décrites dans la gamme et tous les items arrivant dans le système productif font l'objet d'une gamme. La modélisation du processus productif est spécifique: chaque poste de travail est rattaché à une seule file d'attente en entrée (cette file d'attente pouvant servir plusieurs postes de travail qui doivent alors être substituables) et il n'est pas nécessaire de déclarer explicitement les chemins partant des centres de production non virtuels. L'option *jobs matrix* du routage de sortie (*Routing out*) incrémente automatiquement la valeur de l'attribut *job* et envoie l'item dans la

file d'attente rattachée au centre de production défini pour l'opération suivante dans le tableau des gammes. La durée de l'opération suit la loi déclarée dans le tableau des gammes (voir ci-dessous).



Illustrons l'usage de cette option : 8 commandes comportant de 3 à 7 opérations doivent être exécutées ; un job supplémentaire est ajouté à chaque commande pour permettre l'envoi du produit terminé vers l'extérieur du système. L'introduction des 8 commandes dans le système productif peut se faire avec une arrivée unique (First at start time, page 56) d'un lot de 8 (Batching, page 57) avec un grand intervalle de temps séparant cette arrivée de la suivante; on peut également utiliser une file d'attente de contenu initialisé à 8 items (voir page 56). Dans les deux cas, on numérote ces items (*label* «*work type*» avec l'option *Unique*) de 1 à 8. Voir l'exemple Routing_Job_Matrix.S8.

Une **alternative** à la solution du tableau de gamme (*jobs matrix*) est possible (et moins commode) : elle consiste à utiliser l'*Information Store* pour décrire les gammes, ce qu'illustre l'exemple JobShop.S8 suivant (qui vise à analyser le fonctionnement en régime de croisière d'un système productif de type *job shop*).

Il existe une dernière option, **Cycle matrix**, non développée ici, qui est utilisée lorsqu'il est nécessaire de décomposer en plusieurs opérations élémentaires, chaque opération réalisée dans un poste de travail (activité).

III-3 Modélisation de systèmes complexes

On commencera par une réflexion sur l'articulation qui existe entre certains processus. Cette compréhension des mécanismes d'articulation est indispensable pour réaliser une bonne modélisation de processus productifs complexes et bien gérer la décomposition des processus. On examinera ensuite rapidement les problèmes que pose ce type de simulateur conçu au départ (comme les autres simulateurs) pour modéliser et simuler des systèmes de production de biens, lorsque l'on veut l'utiliser pour modéliser et simuler des systèmes de production de services.

III-3.1 Problèmes de synchronisation posés dans une modélisation de simulation

De nombreux problèmes de **synchronisation** se posent dans la modélisation de systèmes productifs complexes. On examinera les problèmes de synchronisation de processus convergents (§ III-3.1.1), de synchronisation de processus séquentiels ne traitant pas les mêmes types d'items (§ III-3.1.2, page 120), puis de synchronisation de processus parallèles (§ III-3.1.1, page 116). La bonne compréhension de ces classes de problèmes et des approches de leur résolution est incontournable pour atteindre une certaine expertise dans la modélisation correcte des processus complexes.

III-3.1.1 Synchronisation de processus convergents

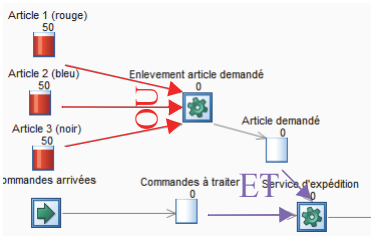
Ils se caractérisent par le fait qu'une activité combine (option *Collect*) dans une même opération des items spécifiques à chacun des processus. Ce cas de figure se retrouve en particulier dans le pilotage des flux de produits (processus de production ou d'approvisionnement) par un flux d'informations, l'exemple classique étant la préparation de livraisons à partir de commandes portant sur des références et des files d'attente de ces références. On a vu à la page 79 comment utiliser les options  et  pour apparier sur la base d'une valeur de label, deux items provenant de deux files d'attente différentes, sur la base de la valeur de *label* prise par le premier item de la première file d'attente dans la liste des files d'attente d'entrée (l'exemple Routing_in_Collect_and_Match_1.S8 et l'exemple Routing_in_Collect_and_Match_2.S8 montrent comment prélever dans une file d'attente, la référence demandée par une commande (item possédant une valeur précise pour un label donné), dans une quantité variable selon la commande, cette commande étant définie par un item d'une seconde file d'attente.

Cette solution n'est plus possible si, au lieu d'avoir une file d'attente unique comportant toutes les références possibles de produits, on a une file d'attente par référence ou plusieurs files d'attente banalisées, susceptibles de posséder la même référence, ce qui revient à combiner la relation OU (sélection du produit) et la relation ET (produit et commande fusionnés). La solution est alors un peu plus compliquée et implique l'usage d'une activité fictive pour des raisons évoquées à la page 113. Appuyons-nous sur un exemple pour illustrer, pas à pas la démarche à suivre. On supposera que l'on s'intéresse à 3 articles référencés (*label* numérique «L couleur») respectivement 1 (rouge), 2 (bleu) et 3 (noir) et disponibles dans 3 stocks (files d'attente) différents, chacun étant initialisé à 50[1]. Dans l'exemple, chacune de ces files d'attente est spécialisée dans une couleur (pour faciliter le suivi de la simulation) ***mais rien n'empêche de trouver une même référence dans plusieurs files d'attente***[2]. Un flux de commandes (point d'entrée) est à honorer dans l'ordre de présentation; chaque commande porte sur 1 article d'une couleur donnée demandé en 1 seul exemplaire (on verra ensuite, comment généraliser la démarche pour prélever un nombre variable d'unité d'une référence susceptible de

1. L'initialisation des files d'attente avant le début d'une simulation a été présentée à la page 56.

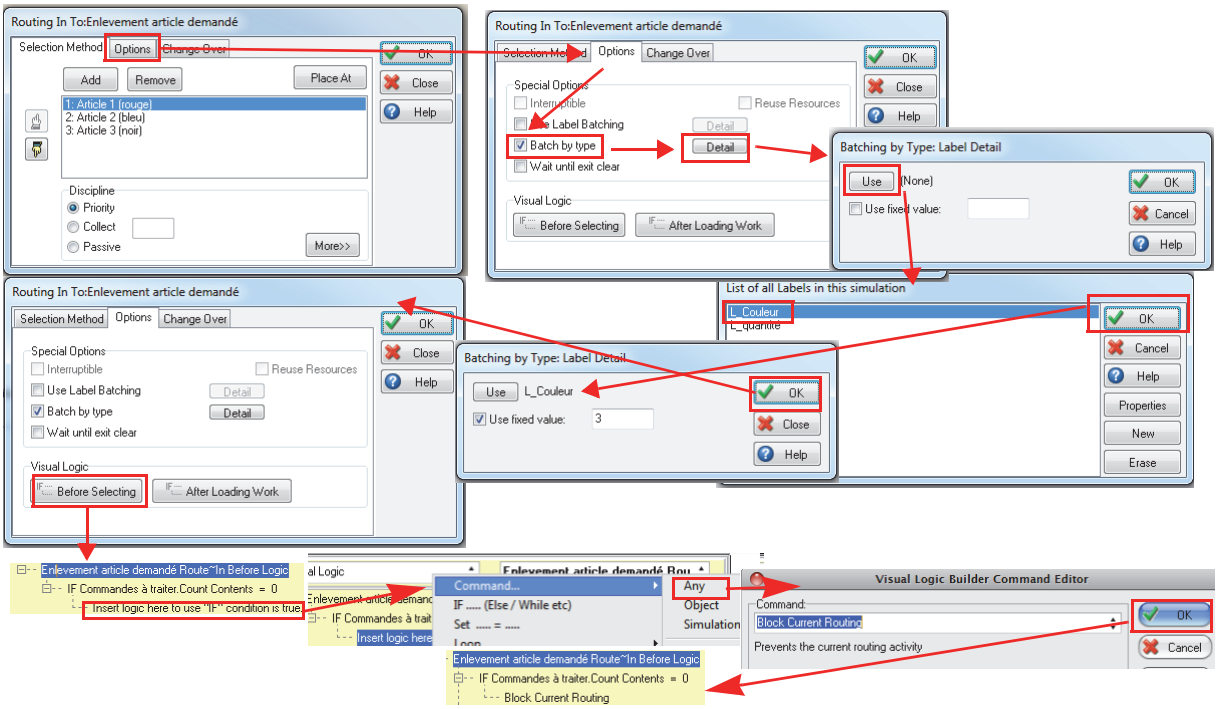
2. Si la variété des références est faible et que chaque référence fait l'objet d'une file d'attente spécialisée, l'approche décrite à la page 80 est possible; voir l'exemple Routing_in_Collect_and_Assemble_2.S8.

se trouver dans plusieurs files d'attente). La modélisation se fait en plusieurs étapes.



On commence par établir la carte du processus étudié et on crée une activité fictive «Enlèvement de l'article demandé» qui doit prendre l'article demandé dans la bonne file d'attente (les files d'attente pointés dans le *Routing in* ne sont pas liés par la relation ET, induite par le choix de l'option *Collect* → il s'agit donc d'une relation OU impliquant que l'item demandé peut être tiré de l'une de ces trois files d'attente) et le mettre dans la file d'attente «Article demandé» qu'utilisera le service d'expédition (là, l'option *Collect* est retenue, ce qui implique que les items provenant de chacune de ces files d'attente doivent être prélevés dans les quantités voulues → il s'agit donc d'une relation ET). Le temps opératoire de cette activité fictive est nul mais il peut ne pas l'être si l'on souhaite décomposer le temps de travail en temps de recherche et temps de traitement ultérieur (paquet…), ce qui est judicieux dans le cas de quantités différentes de 1.

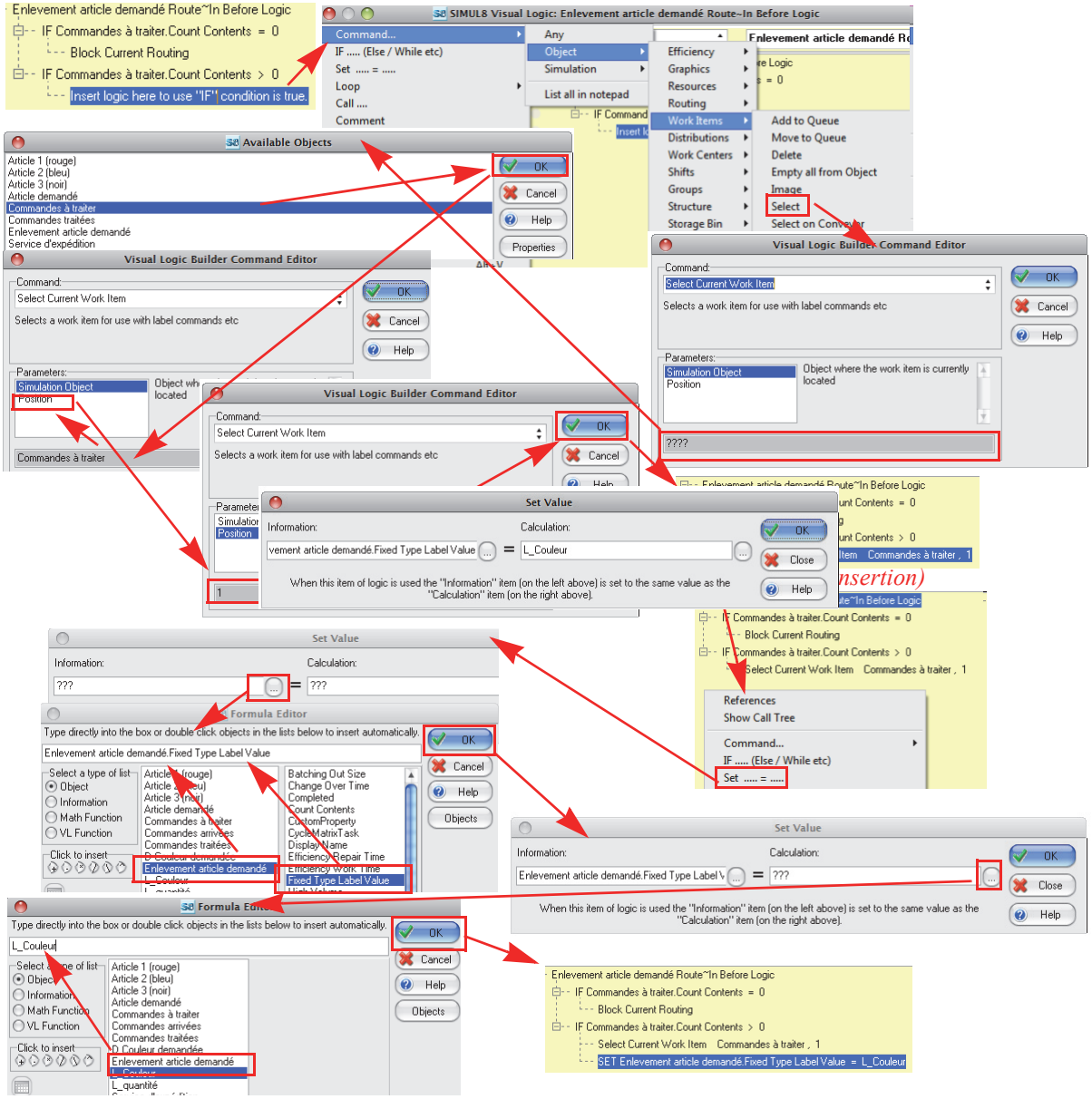
Pour enlever le «bon» article, le centre Enlèvement doit indiquer qu'il recherche, dans les files d'attente qui l'alimentent, un article dont la valeur du *label* «couleur» est 1, valeur qu'il convient de changer avec *Visual Logic* en



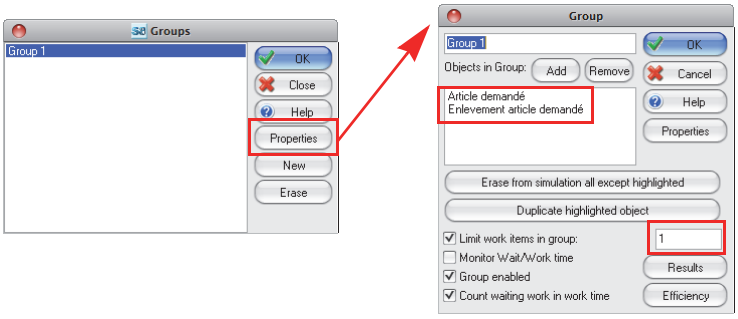
cliquant sur le bouton 

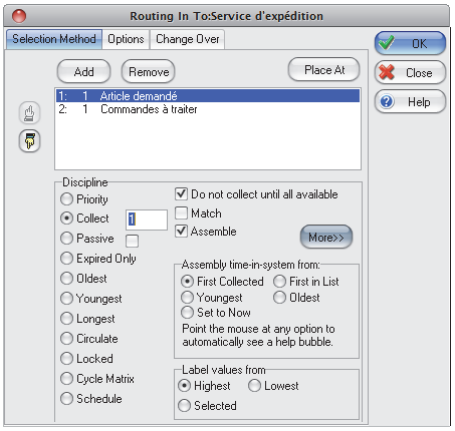
, on ouvre la fenêtre de saisie de *Visual Logic*. Il ne faut rien faire tant qu'il n'y a pas de commande à traiter, ce qui implique l'usage d'une condition portant sur le nombre d'items en attente dans la file d'attente «Commandes à traiter» et de la commande de blocage.

S'il y a au moins une commande à traiter (IF Commandes à traiter.Count Contents > 0) il faut sélectionner le premier *item* de la file d'attente «Commandes à traiter» On



assigne alors la valeur du *label* «couleur» de cet *item* de la file d'attente «Commandes à traiter» comme valeur de *label* du *Batch By Type* dont on avait donné la valeur 1 par défaut. Il ne reste plus qu'à empêcher qu'il y ait plus d'un article dans le **groupe** (voir page 63) constitué de l'activité «Enlevement article demandé» et de la file d'attente qu'il alimente «Article demandé» afin d'éviter tout conflit.





Il faut enfin spécifier le travail du service d'expédition qui «compare» l'information et le produit sur le *label* «couleur» (on se retrouve à ce point exactement dans le cas de figure traité à la page 79, puisqu'un article sera à prélever dans chaque file d'attente).

Le modèle de simulation est prêt (voir Synchronisation_stock_produits_Multiples.S8). Cette simulation propose une variante de modélisation dans laquelle le nombre d'unités demandées est supérieur à 1 et peut changer d'une commande à l'autre. Insistons sur le fait que cette modélisation n'implique pas d'avoir des files d'attente spécialisées; autrement dit une même référence peut se trouver dans plusieurs files d'attente.

Implicitement, ici, on est sur le traitement d'une ligne d'un bon de commande; l'exemple Generation_et_Traitement_BC.S8 traite de manière complète le traitement d'un bon de commande.

Une première alternative à l'usage du concept de *group* (introduit à la page 63) consiste à utiliser une variable créée dans *Information Store*, bloquant l'accès à l'activité «Enlèvement article demandé» si elle vaut 1 (voir l'exemple Synchronisation_stock_produits_Multiples_2.S8) cette solution est préférable dans la modélisation de certains processus de service complexes (caisse de supermarché, par exemple). La seconde alternative, plus simple, consiste à ajouter une seconde condition de blocage des arrivées dans l'activité «Enlèvement article demandé» si la file d'attente «article demandé» n'est pas vide (voir l'exemple Synchronisation_stock_produits_Multiples_3.S8).

Le tableau 8 de la page 120 récapitule des solutions à utiliser pour gérer la synchronisation des processus convergents (flux de demandes avec un flux de produits - référence unique demandée)

Tableau 8 : Récapitulatif des solutions à utiliser pour gérer la synchronisation des processus convergents

Quantité demandée	La référence demandée	
	se trouve une file d'attente unique†	peut se trouver dans plusieurs files d'attente‡
1	*Collect* + *Match on*. Voir page 116 et Routing_in_Collect_and_Match_1.S8	Introduction de 1 activité fictive et du *Batch by Type*. Voir page 116 et le premier modèle de Synchronisation_stock_produits_Multiples.S8
>1	Collect + *Match on* + modification du *Collect Number*. Voir page 79 et Routing_in_Collect_and_Match_2.S8	Introduction de 1 activité fictive et du *Batch by Type*. Voir page 116 et le second modèle de Synchronisation_stock_produits_Multiples.S8 Utilisation directe du *Collect,* voir page 105 Synchronisation_stock_produits_Multiples_Collect.S8

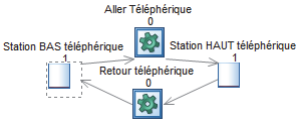
†. Cette file d'attente peut contenir des références différentes (valeurs différentes pour le *label* utilisé pour prélever un item dans une file d'attente).
‡. Chaque file d'attente peut contenir des références différentes. L'item demandé est ensuite fusionné avec l'item demandant par l'option *Collect* d'une activité (autre que ceux mentionnés ici).

III-3.1.2 Synchronisation de deux processus séquentiels ne traitant pas les mêmes items.

Par exemple dans un service d'urgence, coexistent un processus de traitement des patients et un processus de nettoyage des box qui n'est mis en œuvre que si le patient a sali le box (on n'est donc pas dans une alternance systématique de deux processus). Un box est déclaré «libre» si le patient a quitté le box et si le box a été nettoyé, lorsque cette opération est jugée nécessaire. Cette synchronisation ne peut se faire que par l'usage d'un programme en *Visual Logic*. L'usage du concept de groupe déjà utilisé dans un exemple de gestion d'un box (voir page 96 et la première modélisation de l'exemple Box.S8) n'est pas ici la meilleure solution; il est préférable d'utiliser deux variables stockées dans l'*Information Store*, la première relative à l'occupation du box (solution adoptée dans la troisième modélisation de l'exemple Box.S8), la seconde relative au besoin de nettoyage du box. L'exemple Synchronisation_processus_sequentiels.S8 commenté permet de comprendre les bases de ce type de synchronisation; l'exemple Service_des_urgences.S8 de simulation d'un service d'urgence comportant une dizaine de box illustre une généralisation de ce mécanisme et l'usage des ressources partagées (médecins, infirmières, brancardiers et agents de service).

III-3.1.3 Synchronisation de deux processus parallèles

Ce type de problème se rencontre, par exemple, dans la mise en place de tournées cadencées (tournées régulières de livraison ou d'enlèvement de marchandise ou de personnes), avec un temps de transport inférieur à l'intervalle de temps séparant deux tournées successives utilisant le même véhicule (au sens large).



L'exemple Synchro_processus_paralleles.S8 s'intéresse à la modélisation du fonctionnement d'un téléphérique et du transport des skieurs. À la fin d'un transport, les deux cabines sont localisées dans les gares (haut et bas). Intéressons-nous à la gare du bas. Un téléphérique la quitte toutes les 10', avec un temps de transport de 4'; on crée un processeur «Transport» avec un temps opératoire de 4'et une capacité maximale de personnes embarquées (voir page 76), ce qui fait que 4'après le départ, les skieurs embarqués arrivent à la gare supérieure. Ce processus de transport des skieurs doit être synchronisé avec le processus de changement de localisation des cabines; ce dernier processus utilise deux stocks (files d'attente), l'un représentant la plate-forme du bas et l'autre la plateforme du haut; ces stocks sont initialisés à 1, ce qui signifie qu'en début de journée, chacun

d'entre eux contient une cabine; l'activité «Aller téléphérique» prend le téléphérique du bas et l'amène en haut, avec un temps opératoire de 10'représentant l'intervalle entre deux départs d'une plateforme. Simultanément, l'activité «retour téléphérique» réalise l'opération inverse. Le déclenchement du transport de voyageurs est alors conditionné à la présence (fugitive dans cette modélisation) d'une cabine dans le stock du bas. La file d'attente qui approvisionne l'activité «transport» correspond en fait à la file d'attente des skieurs après contrôle, une partie étant sur le quai. On considère ici que la limitation du nombre de skieurs admis dans la cabine se fait au départ de celle-ci, ce qui est inexact mais sans incidence sur la qualité de la simulation (une modélisation plus fine est possible mais sans réel intérêt).

Une solution alternative consiste à utiliser un programme en *Visual Logic* exécuté à cadence régulière (*Visual Logic Time Check*, voir page 108) en utilisant une variable binaire permettant d'autoriser le départ du transport. Cette solution est moins intéressante que la précédente car elle ne permet pas de modélisation du processus physique de transport qui peut être soumis à des aléas.

III-3.1.4 Synchronisation d'un processus divergeant puis convergeant

Ce cas de figure se rencontre quand un item se décompose en plusieurs items que l'on doit refusionner ultérieurement. Par exemple, le prélèvement sanguin d'un patient est une «émanation» du patient, suivant un processus spécifique (analyse) pendant que le patient suit un autre processus; ultérieurement, les résultats des analyses sont communiqués au patient et deviennent de nouvelles caractéristiques du patient (implicitement ou explicitement sous la forme de *labels*). L'exemple Synchro_proces_diverg_converg.S8 illustre la prise en compte de ce cas de figure.

III-3.2 Modélisation/simulation de systèmes de production de services

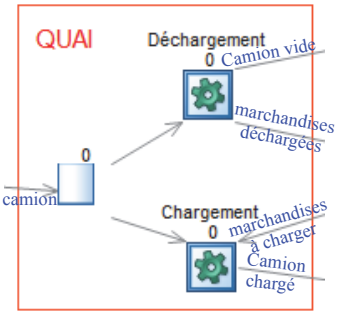
Les logiciels de modélisation et de simulation de processus ont été initialement conçus pour décrire et simuler des processus de production de biens. Ce contexte de production jouit de trois caractéristiques: le traitement d'un item ne peut s'effectuer que dans un processeur occupant un endroit dédié à ce seul processeur; les processeurs ou les ressources partageant les mêmes caractéristiques sont rigoureusement interchangeables; enfin, ni les ressources, ni les items n'interagissent. Dans les processus de production de services lorsque l'item traité est un être humain, ces caractéristiques peuvent être plus difficilement acceptables[1].

- Le **lieu d'exécution de certains traitements** ne nécessitant pas d'environnement matériel spécifique peut ne **pas être circonscrit**, ce qui rend la définition spatiale du processeur moins précise ou peut conduire à ne pas lier obligatoirement un traitement à un processeur. Par exemple, les informations échangées entre un patient et l'infirmière qui l'accompagne à son box dans un service d'urgences sont une partie d'un «traitement». Cette limitation n'est pas trop pénalisante, par rapport à la suivante.
- **Un lieu donné peut jouer successivement plusieurs rôles** et devoir être considéré conventionnellement comme la répétition d'une séquence de «file d'attente - activité», chaque activité effectuant des traitements différents. Ce cas de figure est rare dans la production de biens, sauf dans celle d'items très volumineux (hélicoptères…) où les problèmes de manutention et/ou d'espace disponible jouent en faveur d'une solution de déplacement des

1. Adapté de S. Balin et V. Giard, «Problèmes méthodologiques posés par la simulation de processus de production de services», Journal Européen des Systèmes Automatisés (APII-JESA), volume 41, n° 9-10/2007, p. 1085-1114

hommes et des équipements (et donc des processeurs) pour effectuer en un même lieu une séquence d'opérations de nature très différente. Dans la production de services, ce cas est fréquent lorsque l'item est un client.

- **Un même lieu peut jouer plusieurs fois le rôle de file d'attente et d'activité pour un même item**. Par exemple, le box d'un service d'urgences est une file d'attente à une personne lorsque le patient est seul, c'est-à-dire sans accompagnement médical et une activité, lorsque celui-ci se trouve en présence d'un médecin et/ou d'une infirmière pour faire l'objet d'un traitement (diagnostic, soins, prélèvements sanguins…). La solution consiste alors à relier une file d'attente unique à autant d'activités qu'il y a de traitements différents, cette différence portant sur les ressources mobilisées. Un exemple de cette configuration est donné avec la simulation Open_shop.S8. On peut également, pour des raisons de commodité si la séquence est prédéterminée, préférer utiliser plusieurs files d'attente qui représentent en réalité un même lieu, ce qu'illustre la simulation Box.S8.
- **Un même lieu peut être utilisé pour des traitements alternatifs radicalement différents**, à chaque fois sur un item (ou un groupe d'items) différent(s). Par exemple, un quai d'un entrepôt peut être utilisé pour décharger un camion pour rangement des marchandises en stock ou être utilisé pour charger un camion après rassemblement des marchandises à livrer. La solution consiste à décrire l'utilisation du quai par une file d'attente unitaire correspondant à l'emplacement du camion en bord de quai aux deux activités «chargement» et «déchargement» qui sont de nature très différente du point de vue de la gestion des flux (1 entrée et *n* sorties pour l'activité déchargement contre *m* entrées et 1 sortie pour l'activité chargement). L'item entrant (camion) en file d'attente doit être caractérisé par un label déterminant le type de traitement (chargement ou déchargement) à effectuer pour l'adresser à l'activité correspondante. Reste enfin à réunir la file d'attente et les deux activités en un seul groupe de capacité égale à un.



- La **banalisation des ressources** peut être remise en cause. Par exemple, si la première affectation d'un médecin à un patient arrivant aux urgences est aléatoire, on s'efforcera de garder ce médecin pour les autres actes médicaux effectués sur ce patient aux urgences impliquant un médecin de même spécialité, ces prestations pouvant ne pas s'enchaîner directement. L'analyse d'un box de l'exemple Service_des_urgences.S8 de simulation d'un service d'urgence illustre la manière de traiter ce problème.
- L'hypothèse d'une **passivité** des clients du système est parfois contestable. Une boîte de conserve accepte d'attendre, pas le client d'un service. Ceci le conduit à ne pas rentrer dans un système saturé, à ne pas respecter la discipline imposée dans une file d'attente… L'hypothèse d'une absence d'interaction entre un client et certaines ressources (opérateur) dans certaines prestations de services est contestable mais peut ne pas avoir d'incidence dans une évaluation globale de la performance d'un processus à partir d'indicateurs objectifs (par exemple, temps moyen de séjour dans le système productif). Par contre, cette interaction peut avoir une influence sur les évaluations individuelles (qualité perçue).

Chapitre VIII

LISTE DES SIMULATIONS DONNÉES EN EXEMPLE

Fichier d'exemple Simul8	*Page(s) où ce fichier est appelé*	***Fichier d'exemple Simul8***	*Page(s) où ce fichier est appelé*
Arrivee_items_par_groupe.S8	pages 57, 113	Routing_in_Circulate_Label.S8	page 78, 83
Arrivees_stochastiques_clients_connus.S8	pages 66, 67, 106	Routing_in_Collect_and_Assemble_1.S8	pages 79, 80, 85, 88
Attente_mini_dans_stock.S8	page 64	Routing_in_Collect_and_Assemble_2.S8	pages 80, 85, 116
bottling.S8	page 16	Routing_in_Collect_and_Match_1.S8	pages 79, 86, 116, 120
Box.S8	pages 63, 96, 97, 120, 122	Routing_in_Collect_and_Match_2.S8	pages 105, 116, 120
Choix_Production_stock_mini.S8	pages 67, 88	Routing_in_label_Mini.S8	page 78
Convoyeur.S8	page 95	Routing_in_longest.S8	pages 78, 83
Decision_periodique.S8	pages 108	Routing_in_longest_Label.S8	pages 78, 83
Distribution_Discrete_et_Information_Store.S8	pages 88, 96, 100, 105	Routing_in_oldest.S8	pages 65, 78, 81, 83
Distribution_Label_Based.S8	page 41	Routing_in_oldest_Label.S8	pages 78, 81, 83
Distribution_Time_Dependant.S8	page 43	Routing_in_Passive.S8	page 77
Distribution_Time_Dependant_0.S8	page 44	Routing_in_priority.S8	page 77, 81, 83
Distribution_Time_Dependant_24heures.S8	pages 44	Routing_in_Priority_Label.S8	page 77, 81, 83
Distribution_Time_Dependant_Jour_semaine.S8	pages 44, 103, 106	Routing_in_Use_Label_Batching_0.S8	page 76
Distribution_Time_Dependant_Jour_semaine_IS.S8	pages 38, 44, 103	Routing_in_Use_Label_Batching_1.S8	page 76, 81
Distribution_Time_Dependant_partition.S8	pages 44, 51, 61, 103	Routing_in_Use_Label_Batching_2.S8	page 76, 84
DOCTEUR.S8	page 46	Routing_in_Use_Label_Batching_OR_Collect.S8	page 79, 82
DOCTEUR_Information_Store_0.S8	pages 38, 41, 100	Routing_in_youngest.S8	page 78, 81, 83
DOCTEUR_Information_Store_1.S8	pages 96, 100, 107	Routing_in_youngest_Label.S8	page 78, 81, 83
DOCTEUR_Information_Store_2.S8	pages 99, 100, 103, 107	Routing_Job_Matrix.S8	page 111, 115
DOCTEUR_ressource.S8	pages 53	Routing_out_2_Labels.S8	page 88, 112

Fichier d'exemple Simul8	*Page(s) où ce fichier est appelé*	***Fichier d'exemple Simul8***	*Page(s) où ce fichier est appelé*
DOCTEUR_Shift.S8	page 54	Routing_out_2_Labels.S8	page 88, 112
Duplication_composant.S8	page 93, 110	Routing_out_Circulate.S8	page 87
fastfood.S8	page 16	Routing_out_Desassembler.S8	pages 90, 113
Generation_Bon_de_Commande.S8	pages 103	Routing_Out_Fractionnement_1.S8	page 89
Generation_et_Traitement_BC.S8	page 103, 119	Routing_Out_Fractionnement_2.S8	page 89
Icone_Item_en_sortie.S8	pages 57, 106	Routing_out_Label_1.S8	pages 88, 112
Impredictibilite_processus_simple.S8	page 14	Routing_out_Label_2.S8	page 88
Initialisation_Stock_deterministe.S8	pages 66, 102, 109	Routing_out_Label_3.S8	pages 88, 107
Initialisation_Stock_stochastique.S8	pages 65	Routing_out_Percent.S8	page 87
Introduction_Simulation.xls	pages 11	Routing_out_Priority.S8	page 88
JobShop.S8	page 115	Routing_out_Shortest.S8	page 87
LIFO.S8	page 64	Routing_out_Uniform.S8	page 87
Open_shop.S8	pages 63, 88, 109, 112, 122	Service_des_urgences.S8	pages 110, 122
Parametrage_TO_fonction_label_quanti.S8	page 37	Sortie_du_systeme_fonction_du_temps.S8	page 61
Parametrisation_distribution_probability_profile.S8	page 88	Stock_&_Information_Store.S8	pages 67, 109
Peremption_fixe.S8	pages 64, 113	Synchro_proces_diverg_converg.S8	page 121
Peremption_variable.S8	pages 64, 103, 108, 113	Synchro_processus_paralleles.S8	page 120
PoliceAeroport.S8	pages 57, 112	Synchronisation_processus_sequentiels.S8	page 120
Pool.S8	pages 69, 93	Synchronisation_stock_produits_Multiples.S8	pages 63, 67, 79, 96, 97, 105, 107, 119
Priority_Items.S8	pages 65	Synchronisation_stock_produits_Multiples_2.S8	page 119
Priority_Resources_niveau_fixe.S8	page 92	Synchronisation_stock_produits_Multiples_3.S8	page 119
Replication_work-center.S8	page 93	Synchronisation_stock_produits_Multiples_Collect.S8	pages 105, 120
Resources_Shift_1.S8	pages 69, 92	Temps_de_transport.S8	page 46, 90
Resources_Shift_2.S8	page 92	Unique.S8	page 58
Resources_Shift_3.S8	page 93	Work_Center_Shifts_Stock_in_individuel_1.S8	page 70, 93
Ressource_Polylavence_pool.S8	page 93	Work_Center_Shifts_Stock_in_individuel_2.S8	pages 93, 107, 108
Routing_in_Circulate.S8	pages 78, 83		

Index

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

V

W

Y

Z