Apport de la simulation en ligne dans l'aide à la décision pour le pilotage des systèmes de production – Application à un système flexible de production

Olivier Cardin

UNIVERSITE DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET DES MATÉRIAUX

Année : 2007

Thèse de Doctorat de l'Université de Nantes

Spécialité : AUTOMATIQUE

Présentée et soutenue publiquement par

Olivier CARDIN

le 26 Octobre 2007

à l'Institut Universitaire de Technologie de Nantes

APPORT DE LA SIMULATION EN LIGNE DANS L'AIDE À LA DÉCISION POUR LE PILOTAGE DES SYSTÈMES DE PRODUCTION – APPLICATION À UN SYSTÈME FLEXIBLE DE PRODUCTION –

Jury

Rapporteurs	:	H. PIERREVAL, Professeur à l'IFMA	LIMOS, Clermont-Ferrand
		A. THOMAS, Professeur à l'ENSTIB	CRAN, Nancy
Examinateurs	:	P. CASTAGNA, Professeur à l'IUT de Nantes	IRCCyN, Nantes
		J.-J. LOISEAU, Directeur de Recherche CNRS	IRCCyN, Nantes
		F. FONTANILI, Maître assistant à l'EMAC	CGI, Albi
		K. KOUISS, Maître de Conférences à l'IFMA	LIMOS, Clermont-Ferrand
		P. PUJO, Maître de Conférences Université Aix-Marseille III	LSIS, Marseille

---

Directeur de Thèse : Pierre CASTAGNA
Laboratoire : Institut de Recherche en Communications et Cybernétique de Nantes
Composante de rattachement du directeur de thèse : Institut Universitaire de Technologie de Nantes

N° ED 366-325

UNIVERSITE DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET DES MATÉRIAUX

Année : 2007

Thèse de Doctorat de l'Université de Nantes

Spécialité : AUTOMATIQUE

Présentée et soutenue publiquement par

Olivier CARDIN

le 26 Octobre 2007

à l'Institut Universitaire de Technologie de Nantes

APPORT DE LA SIMULATION EN LIGNE DANS L'AIDE À LA DÉCISION POUR LE PILOTAGE DES SYSTÈMES DE PRODUCTION – APPLICATION À UN SYSTÈME FLEXIBLE DE PRODUCTION –

Jury

Rapporteurs	:	H. PIERREVAL, Professeur à l'IFMA	LIMOS, Clermont-Ferrand
		A. THOMAS, Professeur à l'ENSTIB	CRAN, Nancy
Examinateurs	:	P. CASTAGNA, Professeur à l'IUT de Nantes	IRCCyN, Nantes
		J.-J. LOISEAU, Directeur de Recherche CNRS	IRCCyN, Nantes
		F. FONTANILI, Maître assistant à l'EMAC	CGI, Albi
		K. KOUISS, Maître de Conférences à l'IFMA	LIMOS, Clermont-Ferrand
		P. PUJO, Maître de Conférences Université Aix-Marseille III	LSIS, Marseille

---

Directeur de Thèse : Pierre CASTAGNA
Laboratoire : Institut de Recherche en Communications et Cybernétique de Nantes
Composante de rattachement du directeur de thèse : Institut Universitaire de Technologie de Nantes

N° ED 366-325

Remerciements

Mes remerciements et toute ma gratitude vont tout d'abord à Pierre Castagna, professeur de l'IUT de Nantes, pour avoir su me guider tout au long de ces trois années et me communiquer tout l'enthousiasme qui le caractérise.

Ma reconnaissance se porte également vers Messieurs Henri Pierreval, professeur à l'Institut Français de Mécanique Avancée, et André Thomas, professeur à l'École Nationale Supérieure des Technologies et Industries du Bois, d'avoir accepté d'évaluer ce travail en qualité de rapporteurs. Leur analyse du manuscrit a été d'une grande aide pour la valorisation de celui-ci.

Que Messieurs Jean-Jacques Loiseau, directeur de recherche CNRS à l'IRCCyN, Franck Fontanili, maître assistant à l'École des Mines d'Albi-Carmaux, Khalid Kouiss, maître de conférences à l'IFMA, et Patrick Pujo, maître de conférences à l'Université d'Aix-Marseille III, trouvent ici l'expression de toute ma gratitude pour avoir accepté d'examiner ces travaux et me faire profiter de leurs interrogations pertinentes.

Il ne m'est pas possible d'oublier l'ensemble du département Qualité, Logistique Industrielle et Organisation de l'IUT de Nantes qui a su prendre, par son soutien tant matériel que moral, et par l'intérêt porté tout au long de ces travaux, une part importante dans la réussite finale de ceux-ci. De même, l'AIP Primeca des Pays-de-Loire, et plus particulièrement Yannick Graton, a une grande implication, notamment dans la mise en place de la démonstration de ces travaux sur la ligne d'assemblage de l'IUT. Je remercie également les membres de l'équipe ACSED de l'IRCCyN pour m'avoir notamment éclairé sur les relations entre ces travaux et ceux des communautés scientifiques voisines.

Une pensée particulière va enfin et bien sur à ma femme et à ma fille pour leur soutien et leur amour, et encore toutes mes excuses pour les heures que mon grand travail vous a volées.

Table des matières

Table des figures

Introduction

Pour assurer leur compétitivité, les entreprises sont souvent contraintes d'investir massivement au niveau des moyens de production, dans l'objectif général de la réduction des coûts et délais à qualité constante. Cette contrainte est particulièrement à l'ordre du jour depuis l'apparition des pays dits *émergents* sur le marché mondial. Ces pays ont réussi à s'imposer grâce aux coûts de fabrication très bas de leurs produits, non seulement grâce au faible coût de la main d'œuvre, mais également grâce à une automatisation pragmatique et efficace des systèmes de production. Pour faire face, les entreprises produisant en petites ou moyennes séries se tournent de plus en plus vers des systèmes de productions dits *flexibles*, permettant de conserver une efficacité comparable à celle de systèmes de production de masse tout en gardant une flexibilité comparable à celle d'ateliers manuels. Ces systèmes ont été conçus [Vollmann *et al.*, 1988] pour raccourcir la durée globale de production dans des ateliers accueillant une grande variété de références par la réduction des temps de réglages des machines. Il en découle une diminution des en-cours et des coûts d'outillage.

L'apparition de ces systèmes il y a une vingtaine d'années a été suivie par celle des dispositifs d'auto-identification [García *et al.*, 2003]. Ces équipements permettent d'embarquer de l'information sur le produit, qui devient alors un agent communiquant capable d'interagir avec son environnement et de participer aux décisions concernant son avenir. Ces dernières années, de nombreuses études ont permis de développer le concept de pilotage par le produit, et d'étudier leur impact sur la production [Pannequin, 2007]. À la base du concept de pilotage par le produit, le système de production est vu comme un système réactif avec lequel le produit, en véritable acteur, interagit. Dans ce type d'approche, avoir une vision globale, complète et détaillée de la production est très difficile, du fait de l'absence de hiérarchie forte entre les différentes composantes de l'architecture de commande. On observe une décentralisation des décisions, puisque chaque produit en cours de production y participe tout au long de son évolution dans le système. De ce fait, le travail de gestionnaire de production est rendu très difficile : lorsqu'une décision doit être prise, le gestionnaire n'a aucune vision précise sur l'effet à court terme de cette décision sur le comportement global du système.

Partant de la constatation que très peu d'outils ont jusqu'à présent été développé pour l'aider dans cette tâche, nous avons cherché dans ces travaux à définir une solution permettant une aide à la décision centrée sur le pilotage de la production, c'est-à-dire sur des décisions affectant le système sur un terme court. Bien que ces travaux aient été motivés de prime abord par une étude sur les systèmes de production flexible, la solution envisagée se doit d'être applicable à l'ensemble des systèmes de production dits *complexes*, c'est-à-dire les systèmes dont le nombre et/ou le degré d'interconnexions des variables à

commander sont importants [Millot, 2007]. C'est principalement cette interconnexion qui, dans le cas général, rend l'appréhension du comportement du système difficile au gestionnaire.

La complexité de ces systèmes a souvent poussé l'entreprise concernée à mener des études de simulation préalables à leur installation. La simulation de flux est un outil très puissant, utilisé depuis déjà plus de quarante ans dans la phase de conception ou de reconception des systèmes manufacturiers. Le développement de tels modèles de simulation s'avère en général long et coûteux, ce qui pousse les industriels à chercher à rentabiliser cet investissement par la réutilisation du modèle. Cette réutilisation peut principalement prendre deux formes : soit le modèle peut servir à la reconception de ce système en vue d'une évolution future, à la conception d'un autre système basé sur les mêmes éléments modulaires, etc. ; soit le modèle est utilisé dans la phase de fonctionnement du système.

Cette dernière possibilité est très prometteuse, mais est rarement utilisée dans les faits car beaucoup de problèmes, tant techniques que théoriques, s'y opposent. Ainsi, il manque une définition claire du cadre dans lequel cette simulation peut intervenir, tant au niveau de l'architecture de commande qu'au niveau de la liaison entre le modèle, le système et le gestionnaire de production.

Cette thèse a été réalisée au sein de l'équipe Analyse et Commande des Systèmes à Évènements Discrets (ACSED) de l'Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN). Depuis plusieurs années, cette équipe intervient en milieu industriel. Cette expérience a permis d'observer un glissement de l'intérêt des industriels pour la simulation : après avoir recherché dans la simulation de flux un outil d'aide à la décision à long terme, ces derniers demandent une aide à la décision sur des horizons beaucoup plus courts.

Dans cette thèse, nous avons voulu explorer la possibilité d'utiliser la simulation dans un cadre plus large d'aide à la décision pour le pilotage des systèmes de production complexes. Le premier chapitre nous permettra, tout d'abord, de positionner la classe de systèmes que nous considérons dans les différentes structures de pilotage existantes. Après avoir présenté la simulation de flux en général, et les deux progiciels que nous avons utilisé dans ces travaux, ce chapitre sera l'occasion de montrer la difficulté qu'ont eue les auteurs de la littérature à définir des solutions applicables dans le cas général d'utilisation de la simulation en liaison avec le système de production. Ce chapitre nous permettra également de dégager les deux principaux problèmes rencontrés par les auteurs et qui, selon nous, les ont empêchés de mener à son terme l'implantation de la solution.

Le second chapitre de ce document s'attarde donc sur ces deux problèmes. Tout d'abord, nous nous focaliserons sur la relation qu'il faut nécessairement établir entre le gestionnaire de production et le module d'aide à la décision. Cette relation est principalement basée sur un partage des tâches, selon les capacités d'analyse et de calcul de chacun des membres de l'architecture. Nous verrons que plusieurs approches sont possibles, allant de l'automatisation complète du processus de prise de décision à la relégation du module au rôle de simple calculateur permettant à l'opérateur humain de prendre sa décision. Ensuite, nous étudierons l'impact de la liaison entre la simulation et le système de production. Cette étude a deux objectifs : évaluer et finalement démontrer l'intérêt de

l'initialisation du simulateur à l'état actuel du système, puis définir les éléments nécessaires à cette initialisation, en termes de correspondance de données entre les deux composantes.

Plusieurs solutions à cette initialisation seront présentées lors du troisième chapitre. Concernant les solutions les plus simples, nous verrons que chacune d'entre elles a un champ d'application préférentiel relativement limité. Dans le but de proposer une solution à la fois performante et applicable dans le cas général, nous proposerons alors le concept d'*observation par la simulation*. Ce concept est développé tout au long de la seconde moitié du chapitre, et de nombreux exemples viennent illustrer l'application de ce concept à un système de production.

La dernière partie de ce document présentera les travaux de développement qui ont été réalisés lors de cette thèse. Ceux-ci portent sur un système de production flexible taille réelle. Les différents concepts présentés préalablement dans ces travaux verront leur implantation détaillée, et des résultats sur l'utilisation de cette aide à la décision dans le lancement d'une production sur ce système seront présentés.

L'avancement de ces travaux a été régulièrement publié lors de congrès internationaux. Ainsi, lors de [Cardin et Castagna, 2005], nous avons présenté l'intérêt de l'utilisation de l'observateur dans une architecture de commande nécessitant l'utilisation de la simulation en ligne. Par la suite, nous avons pu, dans [Cardin et Castagna, 2006a], expliciter les principes de fonctionnement de l'observateur notamment au niveau du recalage avec le système réel. Nous y avons de plus présenté des résultats sur l'utilisation de la simulation en ligne que nous détaillons lors du Chapitre IV de ce document. L'ensemble de ces travaux a finalement donné lieu à une publication dans un journal international [Cardin et Castagna, 2008], reprenant en substance l'ensemble des éléments présentés dans cette thèse.

Chapitre I.

Le pilotage d'une unité de production et l'aide de la simulation

L'objet de ces travaux est l'aide à la décision pour le pilotage des unités de production, et notamment des Systèmes Automatisés de Production (SAP). Ce premier chapitre nous permettra, tout d'abord, de positionner la classe de systèmes que nous considérons et les différentes structures de pilotage existantes.

Nous nous intéressons plus particulièrement à la mise en œuvre d'outils de simulation de flux sur cette classe de systèmes. La seconde partie de ce chapitre sera donc consacrée à la présentation de la simulation de flux en général, et des deux progiciels que nous avons utilisé dans ces travaux en particulier.

Nous verrons également à travers un état de l'art que la mise en œuvre de la simulation en ligne a déjà été évoquée dans la littérature, et qu'il a été très difficile pour les auteurs de définir des solutions applicables dans le cas général d'utilisation de la simulation en liaison avec le système de production. Ce chapitre nous permettra enfin de dégager les deux principaux problèmes rencontrés par les auteurs et qui, selon nous, les ont empêchés de mener à son terme l'implantation de la solution.

I.1 Le pilotage de la production

I.1.1 Structures de pilotage

Dans la fin des années 1970, le concept de système automatisé de production a été associé à celui de CIM (*Computer Integrated Manufacturing*). CIM est utilisé pour décrire l'intégration de tous les éléments impliqués dans la production par des moyens informatiques. [Doumeingts *et al.*, 1995] proposent une définition élargie de CIM comme étant est une approche globale, dans un environnement industriel, qui tend à améliorer les performances industrielles. Cette approche est appliquée de manière intégrée à toutes les activités, de la conception à la livraison et au service après-vente, et utilise de multiples méthodes, moyens et techniques (informatisées et automatiques) pour simultanément améliorer la productivité, diminuer les coûts, respecter les dates prévues, augmenter la qualité des produits, assurer la flexibilité à un niveau local ou global du système de

production, et inclure tous les acteurs. Dans une telle approche, les aspects économiques, sociaux et humains sont au moins aussi importants que les aspects techniques. Le concept CIM a été développé pour aider les industriels, qui, au vu de cette définition, ont compris qu'une solution CIM universelle ne pouvait exister, et qu'il fallait donc en créer un original pour chaque application. Pour faire face à la complexité induite par la prise en compte de tous les acteurs du pilotage, le système est décomposé en plusieurs niveaux, selon l'horizon temporel avec lequel ils travaillent (Figure 1). Chaque niveau englobe plusieurs acteurs, en général conçus pour pouvoir communiquer entre eux. L'enjeu des méthodologies basées sur le concept CIM est de construire, selon les fonctionnalités nécessaires au bon pilotage du système, les différents niveaux et de s'assurer de la bonne communication entre ces niveaux.

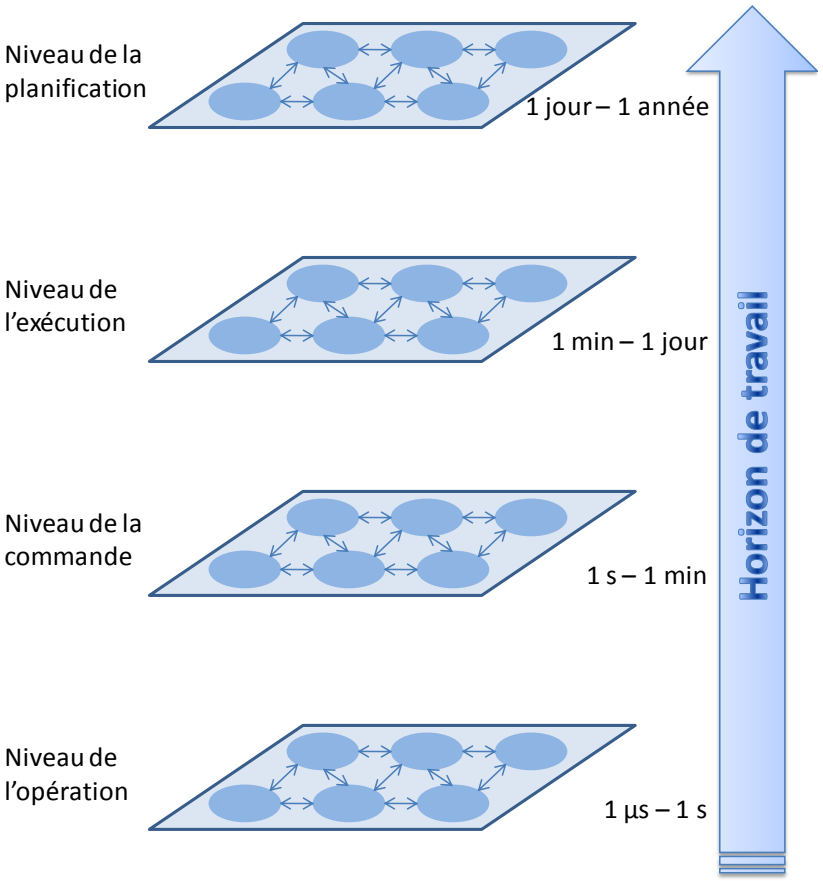

Figure 1 Décomposition d'un Système de Production

I.1.1.1 Le niveau de la planification

Le premier niveau, Niveau de la planification, contient toutes les applications permettant une vision à long terme de l'évolution du système. Diverses décompositions sont encore faites à l'intérieur du niveau, mais ce n'est pas l'objet de notre propos. Parmi les applications rencontrées, nous pourrons citer :

- Les ERP : *Enterprise Resources Planning*. Système d'information intégré d'entreprise adressant l'ensemble (transactions et exécution) des fonctions classiques d'une entreprise, comptabilité, commercial et ventes, production et gestion des matières, administration des ventes, maintenance. Les différents modules et fonctions sont intégrés autour d'un référentiel de processus unique. L'organisation des traitements et des données de tels systèmes répond à des préoccupations de découpage analytique par fonction, de suivi des performances financières et d'exécution, de suivi des principales (ex : SAP, Oracle, Adonix, etc.).
- Les SCM : *Supply Chain Management*. C'est le processus de planification, d'implémentation et de contrôle de la supply chain. Le SCM englobe les transports, les stockages et les transformations des matières premières, des encours de production, et des produits finis du point d'origine des produits à leur point de consommation. Le principe de base de la SCM est qu'on obtient un résultat moins bon en optimisant chaque élément de la chaîne, plutôt qu'en optimisant la chaîne globale (la somme des optimums de chaque élément de la chaîne est inférieure à l'optimum global).
- Les APS : *Advanced Planning and Scheduling*. L'APS est une application destinée à la planification. En fonction de la demande, elle permet d'analyser la capacité des ressources et les contraintes afin de proposer un horaire détaillé et adaptable pour une production optimale.

I.1.1.2 Le niveau de l'exécution

Le second niveau, Niveau de l'exécution, contient les applications permettant la gestion des produits de leur entrée dans le système à leur sortie. Ce niveau est de plus en plus composé du seul *Manufacturing Execution System* (MES) [Huang, 2002]. Nous opterons pour la définition de [Barkmeyer *et al.*, 1999] :

> *Un MES est un ensemble de composants matériels et logiciels permettant la gestion et l'optimisation des activités de production du lancement de la commande aux produits finis. Tout en maintenant des données précises et à jour, le MES guide, initie, répond et rapporte les évènements de l'usine au moment où ils se produisent. Un MES fournit des informations critiques aux modules d'aide à la décision à travers l'entreprise.*

Notons que le MES ne se substitue pas au système informatique d'entreprise (SIE) qui regroupe des fonctions de gestion de l'entreprise (comptabilité, logistique, GPAO etc.) Le MES vient compléter le SIE en assurant des fonctions de contrôle/commande (SCC) qui permettent le pilotage en temps réel des ateliers de fabrication. Malgré toute l'utilité pouvant être retirée de ce complément, force est de constater que le MES est encore mal connu, ce qui freine son implantation quasi-systématique, au même titre que les ERP.

L'un des principaux travaux du MESA, association américaine à but non lucratif a été, outre la création du terme de MES, l'établissement d'une liste détaillée de fonctionnalités, connues comme les « 11 fonctions du MES ». Cette classification est d'un grand intérêt pour délimiter clairement le domaine du MES et évaluer la couverture des différentes offres. Ces 11 fonctions sont :

1. Gestion des ressources : Cette fonction suit le statut des ressources et met à jour un historique détaillé.
2. Ordonnancement : Cette fonction fournit le jalonnement des activités indépendantes basé sur des priorités, attributs, caractéristiques et/ou gammes associés aux produits concernés pour respecter les objectifs de performance définis par l'utilisateur.
3. Cheminement des produits et des lots : Cette fonction dirige le flux de produits selon le plan de production et l'ordonnancement détaillé.
4. Gestion des documents : Cette fonction contrôle, gère et restitue diverses informations relatives aux produits, dont les gammes, les plans, les procédures opératoires standards, etc.
5. Collecte et acquisition de données : Cette fonction collecte et met à jour les informations de production utilisées pour le suivi des produits, les historiques de production, etc.
6. Gestion du personnel : Cette fonction permet un suivi « à la minute près » du statut du personnel (présence, production, préparation de matériel, etc.)
7. Gestion de la Qualité : Cette fonction permet une analyse des produits et des processus de production pour assurer une qualité suffisante de la production.
8. Gestion du procédé : Cette fonction est totalement décrite dans les descriptions de la Gestion de la Qualité (7) et du Cheminement des produits et des lots (3). Cette distinction n'est réalisée que pour préciser qu'un système différent peut être utilisé pour réaliser cette fonction.
9. Gestion de la maintenance : Cette fonction historie, dirige, assure la périodicité et gère la disponibilité des outils nécessaires aux opérations de maintenance.
10. Traçabilité produit et généalogie : Cette fonction permet une visibilité sur la disposition spatiale et temporelle de chaque opération.
11. Analyse des performances : Cette fonction permet un suivi « à la minute près » des résultats courants des opérations de production (utilisation des ressources, disponibilité des ressources, temps de production unitaire, conformité à l'ordonnancement prévu, etc.) et une comparaison à l'historique et aux objectifs de performance.

I.1.1.3 Le niveau de la commande et le niveau de l'opération

Le troisième niveau, Niveau de la commande, comprend les applications et matériels nécessaires au contrôle/commande du système automatisé de production. Les travaux que nous présentons ici ont pour but une application aussi large que possible de la méthodologie proposée. Toutefois, pour ne pas surcharger inutilement notre propos, nous ne considèrerons que les architectures centrées sur des Automates Programmables Industriels (API).

Enfin, le quatrième niveau, Niveau de l'opération, contient tous les matériels nécessaires à la détection et à l'actuation : capteurs, actionneurs, etc.

La communication entre les deux derniers niveaux se fait au moyen de différents protocoles (Ethernet industriel, FipIO, ProfiNet, etc.). Au travers des fonctions décrites ci-avant, le MES est l'outil de communication permettant un lien entre les trois premiers niveaux. On obtient donc finalement une architecture décrite Figure 2.

En réalité, une telle décomposition est relativement artificielle. Les frontières entre ces différents niveaux sont souvent mal définies. Ainsi, on voit par exemple se développer aujourd'hui le concept de capteur intelligent, qui place ce dernier entre les troisième et quatrième niveaux, les ERP (1er niveau) intègrent de plus en plus de fonction décrites comme appartenant au MES (2ième niveau), etc.

On notera que cette décomposition est une variante de la décomposition systémique présentée dans [Le Moigne, 1990] et [Lenclud, 1993] qui ne considère le système de production comme n'étant composé que de trois sous-systèmes :

- Un sous-système opérant (Niveau de l'opération)
- Un sous-système d'information (Niveau de l'exécution et Niveau de la Commande)
- Un sous-système de décision (Niveau de la planification).

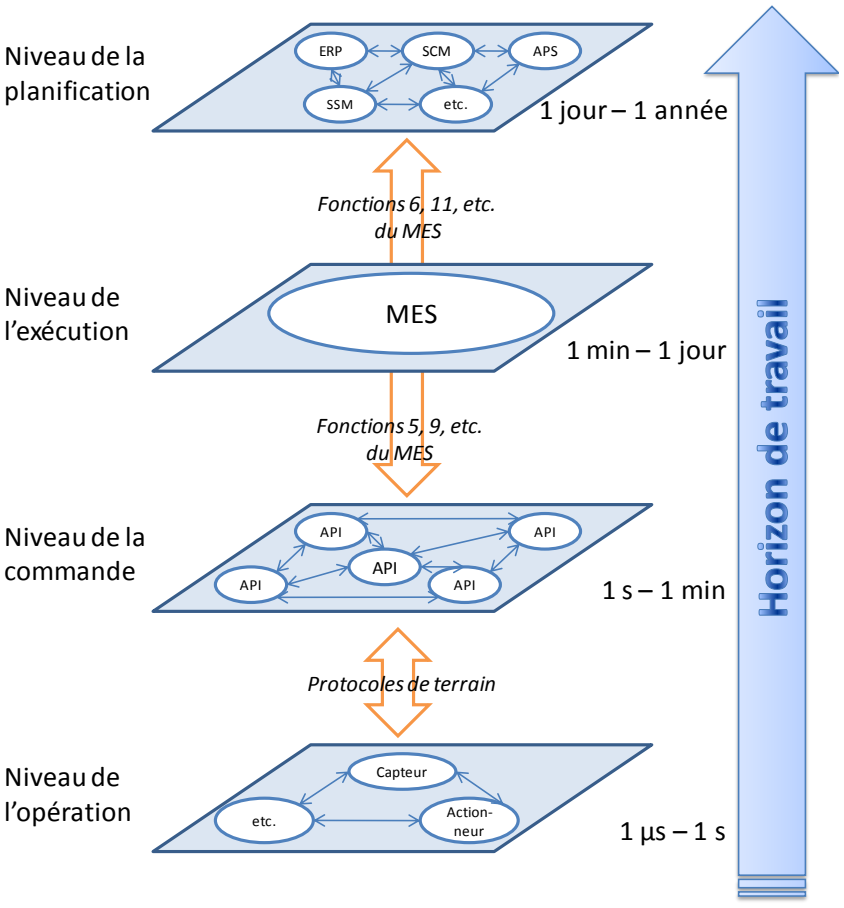

Figure 2 Décomposition détaillée d'un Système de Production

I.1.1.4 Intégration de nos travaux à l'architecture de commande

L'objet de nos travaux est l'ajout, au niveau de l'exécution, de nouvelles fonctionnalités d'aide à la décision. En effet, les fonctions 1, 2, 3, 6 et 9 du MES (voir ci-dessus) induisent des prises de décision pour leur accomplissement ou directement après analyse des données qu'elles fournissent. Nous proposons dans ce travail d'introduire un outil de simulation communiquant avec le niveau de la commande pour compléter les informations fournies par le MES avec des résultats de tests d'alternatives de décision. Cet outil vient compléter la relation qui existe entre l'opérateur et le niveau de la commande.

La vision du SAP traduite par la Figure 2 est une vision très hiérarchique du pilotage du système de production. Si cette approche du pilotage est encore largement répandue dans l'industrie, elle est depuis une vingtaine d'année largement remise en question par la communauté scientifique. Le développement des concepts de flexibilité et d'agilité de la production ont entrainé l'émergence de nouvelles structures de pilotages plus réparties comme les structures hétérarchiques ou holoniques. Cette évolution a aussi été permise par le développement des réseaux informatiques permettant d'avoir des systèmes d'informations et de décision eux même répartis.

Le prochain chapitre examine les différentes structures hiérarchiques possibles du niveau de la commande. Nous nous intéressons plus particulièrement à cette structure car c'est elle qui engendre des besoins variés au niveau de la communication entre les niveaux.

I.1.2 Hiérarchies de pilotage

I.1.2.1 Structures hiérarchiques

Utilisées depuis presque un siècle, les hiérarchies à approche descendante (top-down) ont été formalisées dans les années 1970 et l'application de la Théorie des Systèmes Hiérarchisés aux systèmes de production [Mesarovic *et al.*, 1970]. Dans cette approche très centralisée, une hiérarchie forte existe entre toutes les composantes du système qui s'appuie sur un flot descendant de commandes et un flot remontant d'informations (Figure 3). Ainsi, un niveau supérieur définit les contraintes et objectifs à atteindre par le niveau suivant. De ce fait, un problème de coordination entre les différents niveaux apparaît, puisqu'il est très fréquent que les problèmes de chaque niveau soient interdépendants. Il faut donc instaurer des mécanismes assurant la cohérence des décisions prises à chaque niveau. [Blanc, 2006] distingue deux approches permettant de résoudre ces problèmes de cohérence et de robustesse apparaissant dans les approches centralisées : une approche analytique où ce sont les niveaux supérieurs qui imposent les décisions et une approche itérative où les niveaux inférieurs peuvent demander la génération d'une nouvelle décision au niveau supérieur en cas d'impossibilité d'application de cette dernière.

Le principal avantage d'une telle structure est qu'elle donne une vision globale du système de production. Cette vision globale permet non seulement la définition claire d'objectifs globaux mais elle permet aussi de prendre des décisions cohérentes à chaque niveau pour atteindre ces objectifs globaux. Par contre, les évolutions de la demande (multiplication des produits et des variantes), et la complexification croissante des systèmes de production due à la mondialisation font qu'aujourd'hui cette vision hiérarchique pose de plus en plus de problèmes. Les avantages d'une telle approche sont nombreux (optimisation

des objectifs globaux ou fiabilité des données par exemple), mais ce qui nous intéresse plus particulièrement concerne sa structure même. En effet, les chemins de remontée d'informations sont particulièrement bien définis, ce qui permet de ne communiquer qu'avec le niveau supérieur pour obtenir l'ensemble des informations qui nous intéressent. Toutefois, la rigidité de l'architecture a tendance à diminuer fortement la réactivité du système. Ce pilotage très hiérarchisé est en effet beaucoup trop peu réactif pour répondre aux contraintes multiples des productions actuelles.

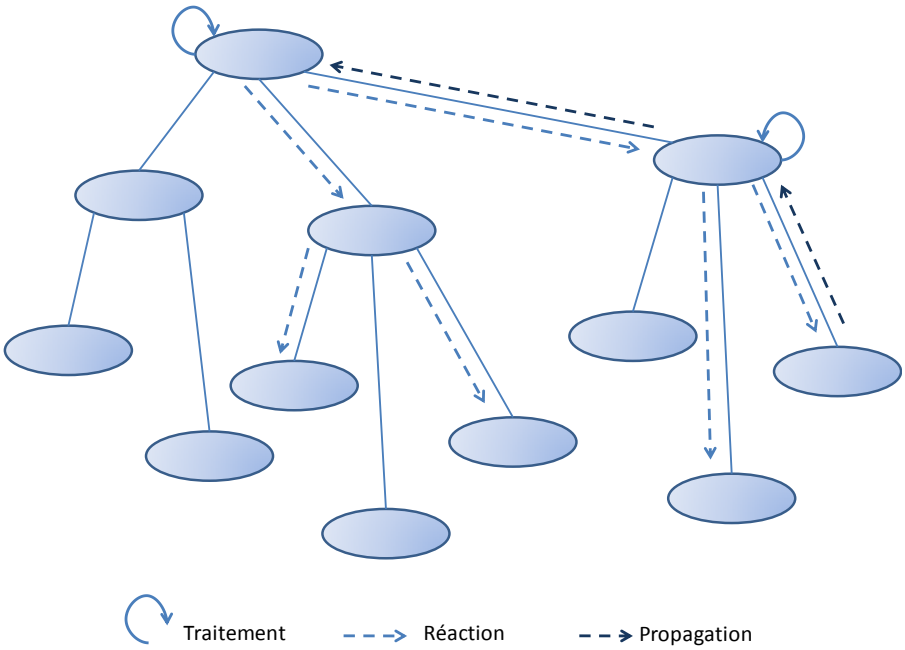

Figure 3 Structure hiérarchique de pilotage [Blanc, 2006]

I.1.2.2 Structures hétérarchiques

Comme le précise Damien Trentesaux dans [Trentesaux, 2002], le substantif hétérarchie (en anglais, Heterarchy) et son adjectif hétérarchique (en anglais, heterarchical) constituent des néologismes à la fois en français et en anglais. Ce terme a été formé à partir de deux termes grecs : *heteros* (autre) et *Arckhein* (commander), signifiant ainsi à l'origine « c*ommandement par les autres »*. L'hétérarchie renvoie à l'idée d'acteurs différents qui assument en collégialité la coordination d'une action collective donnée et s'oppose par essence au terme hiérarchie.

Depuis le milieu des années 1990, des structures hétérarchiques se sont développées, où l'orientation des relations entre les différents éléments constitutifs du système n'est pas définie à priori. Ces éléments sont capables d'agir et de communiquer, ainsi que d'accéder aux ressources ou d'interagir : on les appelle des *agents*. Beaucoup de définitions concurrentes du concept d'agent existent, dues à l'augmentation incessante de leurs fonctionnalités. Néanmoins, nous considèrerons la définition tirée de [Tweedale *et al.*, 2007] : « un agent peut être considéré comme un composant logiciel et/ou matériel du système capable d'agir en se conformant aux règles de fonctionnement du système pour accomplir une tâche précise pour son utilisateur ». L'information et la décision sont donc distribuées, ce qui impose d'instaurer des mécanismes de coordination : chaque proposition est transmise à tous les agents concernés qui peuvent l'accepter ou la remettre en cause.

L'avantage d'une telle structure est bien évidemment sa robustesse : l'ajout d'un agent ou la suppression temporaire pour cause de panne par exemple sont très faciles à mettre en œuvre dans une telle structure. Toutefois, une démarche d'optimisation de critères globaux est très difficile : aucun agent n'ayant une vue complète du système, il ne connaît pas l'impact de sa décision sur le fonctionnement global de celui-ci. De plus, c'est par l'utilisation de règles locales qu'émerge le comportement global du système. Une règle permet une prise de décision basée sur le résultat d'une condition logique [McFarlane *et al.*, 2002]. Ceci rend impossible à prévoir les performances d'un agent pris individuellement, puisqu'il dépend grandement des interactions avec les autres agents. Ces règles ont évidemment un énorme impact sur les performances globales du système, et seule la simulation permet de les ajuster [Blanc, 2006].

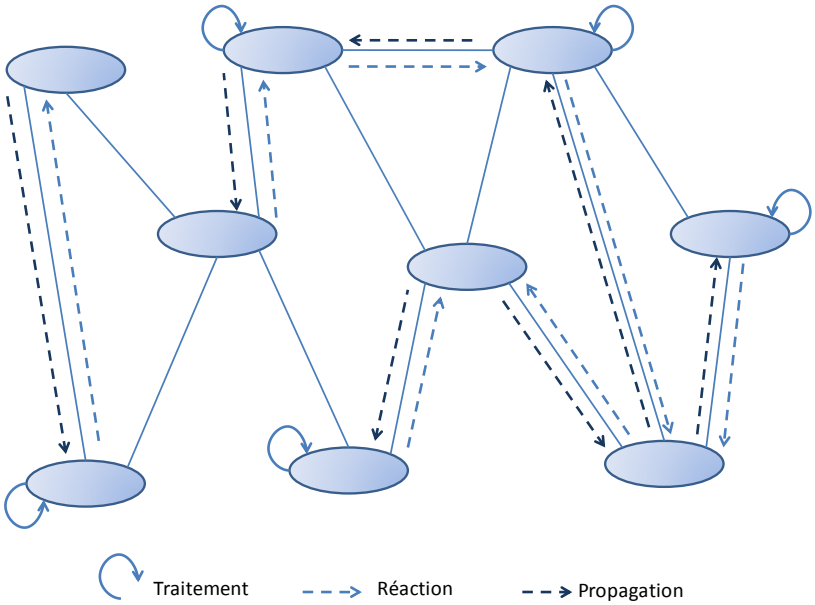

Figure 4 Structure hétérarchique de pilotage [Blanc, 2006]

I.1.2.3 Structures holoniques

Le concept holonique a été introduit par l'écrivain et philosophe hongrois Arthur Koestler qui étudiait les organismes vivants et les organisations sociales. Il a mis en évidence le fait que, dans la vie réelle, une entité, pouvant être un tout ou une partie, ne peut-être considérée comme un absolu. Il a alors introduit le concept de Holon, concaténation du terme « holos », signifiant le tout et du suffixe « on » suggérant une partie.

Dans un système de production holonique, le holon comporte alors à la fois une dimension physique (il correspond à une réalité du système physique de production) et une dimension logique (il possède des informations et est doué d'intelligence). De plus, comme dans la définition générique de Koestler, les holons doivent obéir à des règles générales fixées par l'organisation et qui permettent d'atteindre un objectif commun.

L'architecture de référence PROSA [Van Brussel *et al.*, 1998] spécifie la manière dont chaque composant du système à piloter est représenté. PROSA est un acronyme pour *Product Resource Order Staff Architecture*. En effet, cette architecture est composée de trois types d'holons, auxquels s'ajoutent les holons *staff* :

- les holons *produits* représentent le savoir-faire utile à la réalisation des produits ;
- les holons *ressources* représentent les équipements permettant de réaliser la production ;
- les holons *ordres* représentent les ordres de fabrication devant être exécutés ;
- les holons *staff*, dont la présence est optionnelle, permettent d'intégrer au système des centres de décision ayant un champ de vision plus large, par exemple un centre effectuant un processus de planification ou d'ordonnancement.

Le paradigme holonique [Valckenaers, 2001] s'est développé, visant à associer une entité informationnelle et décisionnelle à chaque élément physique du système de production [Pannequin et Thomas, 2006a].

I.1.2.4 Les systèmes pilotés par le produit

Avec le développement conjoint des technologies d'identification par ondes radio (RFID – Radio Frequency Identification), une application de plus en plus fréquente est liée au concept de produits intelligents. Un produit intelligent est muni d'un système d'auto-identification (Auto-ID) appelé étiquette (tag), qui permet de le relier aux informations du réseau de commande du système de production. Un produit intelligent se doit d'avoir tout ou partie de ces cinq caractéristiques [Wong *et al.*, 2002] :

1. Il possède une identification unique
2. Il est capable de communiquer efficacement avec son environnement
3. Il peut retenir et conserver des données le concernant directement
4. Il utilise un langage pour communiquer ses besoins, ses possibilités etc.
5. Il est capable de participer à la prise de décision concernant son devenir

L'intérêt de l'utilisation de cette technologie est discuté dans [García *et al.*, 2003] au niveau de la production, du stockage, de la distribution ou encore de la vente au détail :

- Possibilité d'adapter facilement le processus de production au produit, sans les limitations physiques du code-barres (accessibilité, distance de lecture). Les systèmes de production deviennent ainsi vraiment flexibles ;
- Stockage de toutes les informations nécessaires sur la même étiquette, sans avoir besoin de ré-étiqueter les produits en fonction des besoins spécifiques du stockage, de la livraison ou après un assemblage entre différents produits.
- Possibilité de gérer facilement des palettes mixtes, contenant plusieurs références de produits ;
- Les inventaires sont grandement facilités, voire totalement automatisés, etc.

Le développement des systèmes contrôlés par le produit [Morel *et al.*, 2003][Pannequin et Thomas, 2006b] permet de créer des systèmes de production partiellement voire totalement réactifs.

Le pilotage de tels systèmes se dirige donc vers des architectures où la plupart des éléments sont autonomes. Ceci permet des stratégies de pilotage très performantes, mais empêche d'avoir une vision globale de l'activité du système, ce qui rend difficile la prise de décision des niveaux supérieurs en relation avec le niveau de la commande. Parallèlement, l'adéquation des structures hétérarchiques avec la simulation de flux pour leur mise au point nous pousse à envisager l'utilisation de la simulation dans un contexte plus large. Nous

proposons donc que l'outil d'aide à la décision évoqué en I.1.1 soit réalisé à l'aide de techniques de simulation de flux.

I.2 La simulation de flux

La simulation est un outil largement répandu dans le monde industriel, et dans le monde de la recherche académique. Depuis 30 ans, la simulation à évènements discrets s'impose, avec l'objectif de pouvoir travailler sur un système de production virtuel, dont le comportement peut être très proche du système réel, à moindre coût et sans risques.

Nous avons choisi d'utiliser des COTS (Commercial-off-the-Shelf), c'est-à-dire des logiciels commerciaux, plutôt que de développer entièrement une application de simulation. Ce choix présente bien sûr des avantages et des inconvénients. Parmi les avantages, nous pouvons citer :

- La réduction des temps et des coûts de développement ;
- Le fait que ces outils sont très implantés dans le milieu industriel, etc.

Mais l'utilisation de COTS n'est pas sans poser quelques problèmes :

- Leur coût d'achat et de maintenance est relativement élevé
- Leur fonctionnement interne peut parfois être relativement « opaque »[Boer et Verbraeck, 2003], etc.

L'utilisation des COTS est principalement dictée par notre souci d'application de ces travaux au milieu industriel. Il est intéressant de noter que l'origine des travaux de l'équipe ACSED (Analyse et Commande des Systèmes à Évènements Discrets) de l'IRCCyN, concernant l'aide de la simulation pour le pilotage des systèmes de production est venue de réflexions avec la société AIRBUS, dans le cadre d'un contrat industriel [Castagna *et al.*, 2001]

La simulation à évènements discrets n'est qu'une petite partie des outils de simulation numériques, très largement utilisés dans le monde industriel. Des spécialistes du marketing simulant des comportements de clients aux concepteurs simulant des comportements de structure mécanique en passant par les concepteurs de systèmes électroniques simulant des circuits intégrés, la simulation est aujourd'hui présente dans l'industrie. Concernant la simulation des systèmes de production, on distingue les modèles continus et les modèles discrets. Les premiers concernent principalement des processus de type continu, comme par exemple une raffinerie dans l'industrie pétrolière. Les seconds concernent des productions de type manufacturières, où les produits sont des objets formant un tout. Nos travaux concernent plutôt ce dernier type de système, et c'est pourquoi nous utilisons des simulateurs à évènements discrets.

Le travail présenté ici a été développé sous deux langages concurrents : le langage SIMAN utilisé dans l'outil Arena (développé par Rockwell Automation) et les langages SCL et BCL implantés dans l'outil Quest (de la suite Delmia développée par Dassault Systèmes). Ce travail ayant vocation à être appliqué à moyen terme sur une installation industrielle, nous avons pris le parti de développer les concepts présentés dans ce mémoire directement sur ces logiciels en particulier, car ils représentent une part importante du parc industriel mondial.

La première partie de ce chapitre présente des cas typiques d'utilisation de la simulation en recherche et en industrie. La seconde partie explicite les principes et objectifs de ce type de simulation. Enfin, nous ferons un rapide panorama des langages et logiciels de simulation présents sur le marché.

I.2.1 Définition – Fonctionnement - Limites - Objectifs

I.2.1.1 Définitions

Nos travaux portent sur la simulation des systèmes à évènements discrets. Elle concerne les systèmes dans lesquels les variables d'état ne changent qu'à des dates discrètes [Banks *et al.*, 1996]. La différenciation entre systèmes discrets et continus a été faite par [Law et Kelton, 1982] : peu de systèmes sont totalement discrets ou continus, mais puisqu'un des deux types prédomine dans la plupart des systèmes, il sera souvent possible de classifier un système comme étant discret ou continu.

Avant de commencer, il semble nécessaire de définir les concepts qui seront utilisés dans ces travaux. Ces définitions sont tirées de [Banks *et al.*, 1996] :

- **Modèle** : Représentation abstraite d'un système, contenant habituellement des relations structurelles logiques ou mathématiques qui décrivent un système en termes d'état, d'entités et attributs, d'ensembles, de processus, d'évènements, d'activités et de retards.
- **Entité** : Tout objet ou composante du système nécessitant une représentation explicite dans le modèle.
- **Attribut** : Propriété d'une entité donnée.
- **Évènement** : Occurrence instantanée qui change l'état d'un système.
- **Activité** : Durée de longueur spécifiée et connue lorsqu'elle débute (ex : temps de production sur un poste de travail).
- **Délai** : Durée d'une longueur indéfinie et non spécifiée qui ne peut pas être connue tant qu'elle n'est pas terminée (ex : temps de séjour dans un stock).
- **Horloge** : Variable système représentant le temps simulé.

I.2.1.2 Démarche d'utilisation

La simulation des systèmes à évènements discrets intègre à la fois la construction d'un modèle et l'utilisation expérimentale de ce modèle pour étudier un problème. De nombreux auteurs proposent de découper une étude de simulation en une succession d'étapes [Law et Kelton, 1982][Banks *et al.*, 1996][Draghici *et al.*, 1998][Fontanili, 1999] [Berchet, 2000] [De Vin et Jägstam, 2001]. Cette méthodologie peut être résumée en ces quelques points (Figure 5) :

1. Analyser, formuler et planifier le projet de simulation, ainsi que les objectifs de ce projet ;
2. Modéliser le problème à l'aide de langages tels que les réseaux de Pétri, chaînes de Markov, UML, etc. À l'issue de cela, choisir le progiciel le mieux adapté à la modélisation choisie et aux objectifs définis. Il faut remarquer ici l'ambigüité progiciel/langage de simulation qui fait que l'on utilise souvent directement le langage de simulation dans cette phase de simulation. Évidemment, on choisit dans

ce cas le progiciel avant de modéliser. Dans le même temps, collecter sur le terrain les données nécessaires à la construction effective du modèle ;

3. Réalisation du modèle, avec notamment la phase d'agrégation des données collectées ;
4. Vérification : le modèle de simulation fonctionne-t-il correctement (non-blocage, conservation des entités, etc.)
5. Validation : les règles implantées dans le logiciel décrivent-elles les flux comme indiqué dans la phase de modélisation ? Cette phase sera principalement réalisée par des tests de bon fonctionnement. Les résultats fournis par le modèle sont-ils justes ? S'ils existent, les résultats de la simulation pourront être par exemple comparés à ceux du système réel ;
6. Définition, exécution et analyse des tests de simulation ;
7. Dès que les objectifs définis au début du projet sont atteints, une documentation précise, puis la prise de décision et l'implantation de la solution retenue doivent être réalisés préalablement à l'exploitation de la solution.

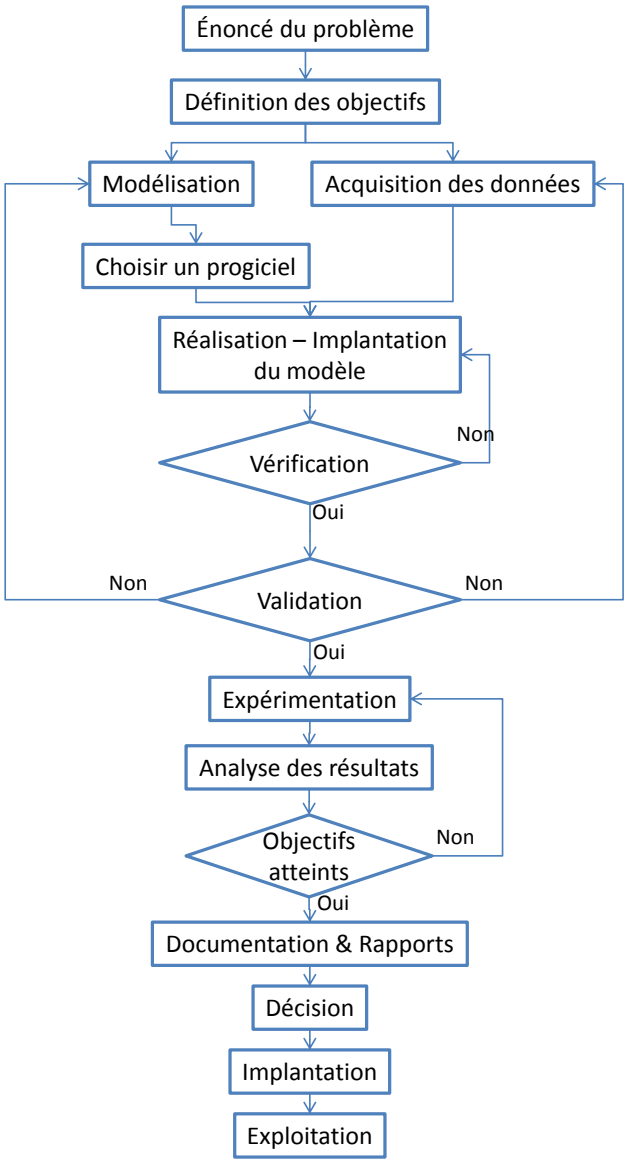

Figure 5 Étapes dans une étude de simulation

Toute la partie qui va de la modélisation à la validation est la plus longue et la plus technique. En effet, la complexité des progiciels de simulation impose en général à un expert de simulation de réaliser le modèle. Partant de ce constat, [Kim et Gibson Jr, 2003] estiment que pour développer l'utilisation de la simulation dans le domaine du bâtiment, une aide interactive à la modélisation devrait être apportée aux entrepreneurs. Cet article traite du *Knowledge-embedded, MOdularized Simulation system* (KMOS), appliqué sur un exemple industriel à travers une interface type « boîtes de dialogue ». [Hollingworth et Spackman, 2007] expliquent de leur côté que, dans le domaine biomédical, le côté « boite noire » des modèles de simulation, que seul le concepteur-expert en simulation peut comprendre, effraie pour le moment encore les praticiens et empêche son déploiement à grande échelle.

On n'utilise pas en général la simulation pour trouver des solutions optimales, mais pour évaluer et comparer des scénarios possibles (voir [Swisher *et al.*, 2003] pour plus d'informations sur les techniques de choix entre les résultats des différents scénarios). Un modèle de simulation est caractérisé par un vecteur de paramétrage. Le problème se résume à comparer différents vecteurs de paramétrage. Un scénario correspond à *un vecteur de paramétrage possible couplé à un ensemble d'évènements de nature et de date prédéterminées venant influer sur le déroulement de la simulation*. Ainsi, un scénario peut par exemple contenir le paramétrage des différents éléments du système modélisé (vecteur de paramétrage) et l'évènement consistant à mettre en panne une des machines du système à une date donnée (nature et date prédéterminées).

Si les aléas jouent un rôle significatif dans le comportement du système (c'est-à-dire avec une fréquence d'occurrence importante relativement à l'horizon simulé), on utilise un modèle stochastique qui sera lancé plusieurs fois successivement pour chaque scénario afin de construire un intervalle de confiance [Howard *et al.*, 1992]. Il reste ensuite à évaluer les solutions, évaluation qui s'avère souvent être multicritère [Mebarki *et al.*, 1998]. Chaque simulation lancée dans ce cadre sera appelée *réplication*. S'il est évident que l'augmentation du nombre de réplications influe sur le temps de calcul global de la simulation, elles n'en restent pas moins indispensable pour confirmer ou infirmer les tendances vues sur les premiers résultats [Kelton *et al.*, 2004].

Il est toutefois à noter que la simulation à évènements discrets est, depuis quelques années déjà [Tautou-Guillaume, 1997] et de plus en plus souvent couplée à des techniques de recherche d'un optimum. Ainsi, [Van Volsem *et al.*, 2007] couplent la simulation à un algorithme évolutionniste pour déterminer la répartition la moins coûteuse des contrôles en cours de production sur un système de production à plusieurs phases en série. Dans cet exemple, la stratégie de contrôle optimale doit décider :

1. Le nombre et la position des stations de contrôle;
2. La taille du lot sujet au contrôle (sample size);
3. La rigueur du contrôle (limites d'acceptation) à chaque station de contrôle qui minimise le coût total du contrôle.

La simulation à évènements discrets est utilisée pour modéliser le système sujet aux contrôles et pour calculer les coûts résultant de la stratégie de contrôle, l'algorithme évolutionniste est utilisé pour optimiser les stratégies de contrôle. Il convient donc de noter que la simulation travaille finalement dans son cadre habituel, celui de test de scénarios (ici les solutions de l'algorithme), même si elle est incluse dans une démarche d'optimisation.

I.2.1.3 Fonctionnement

L'un des avantages principaux d'une simulation est de pouvoir simuler plusieurs heures de fonctionnement du système modélisé en quelques minutes (voire quelques secondes). Pour ce faire, différents modes de fonctionnement des moteurs de simulation ont été envisagés, et plus particulièrement plusieurs mécanismes d'avancement du temps. En effet, le temps étant un paramètre crucial et commun à tout modèle de simulation, la manière dont il avance aura une grande influence sur le fonctionnement du simulateur.

Ainsi, il a par exemple été question d'utiliser un fonctionnement synchrone cadencé par une horloge. Dans ce cas, le temps est discrétisé en intervalles égaux (1 minute par exemple). À chaque top d'horloge, le moteur détermine l'évolution du simulateur compte tenu des évènements qui se sont déroulés sur la dernière période. Ce mode de fonctionnement a deux inconvénients majeurs :

- Lors des phases d'inactivité du système, le simulateur est obligé de prendre en compte des périodes où rien ne s'est passé, ce qui ralentit inutilement son déroulement ;
- Lors des phases d'activité plus intenses, plusieurs évènements peuvent avoir lieu durant la même période, ce qui implique une indétermination sur l'ordre d'exécution des évènements dans le simulateur.

Par contre, ce fonctionnement cadencé par une horloge permet une grande maitrise de la vitesse d'évolution du temps dans le simulateur. Cette maitrise est précieuse lorsque l'on cherche à faire communiquer entre eux plusieurs simulateurs ou lorsqu'on cherche à piloter un simulateur avec un système de commande externe (concept d'émulation).

Dès lors, un nouveau mode de fonctionnement est apparu, qui est désormais largement répandu dans la communauté des progiciels de simulation de flux. Ces moteurs fonctionnent sur le principe du pilotage par les évènements. L'horloge avance de manière discrète (de pas souvent inégaux) pendant la simulation. Après que toutes les actions possibles aient été réalisées à une date donnée, l'horloge change pour la prochaine date d'occurrence d'un évènement. [Schriber et Brunner, 1994] expliquent donc que l'exécution d'une simulation prend la forme d'une boucle à deux phases : « exécuter toutes les actions possibles à un temps donné », suivi de « avancer l'horloge de simulation », avec ces deux phases répétées encore et encore jusqu'à ce qu'une condition de fin de simulation soit rencontrée. Ces deux phases sont respectivement appelées la Phase de Mouvement des Entités (PME) et la Phase de Mise à jour de l'Horloge (PMH).

À un instant donné, un évènement est datable (on connait sa prochaine date d'occurrence) ou non-datable (cette date n'est pas connue). La transformation d'un évènement de l'état non-datable vers l'état datable est la conséquence de l'exécution d'un évènement. A la fin de chaque PME, le simulateur reconstruit le calendrier, qui est la liste ordonnée par date d'occurrence croissante des évènements actuellement datables, puis consulte le calendrier pour déterminer les actions à entreprendre lors de la PMH. Lors de la PMH, l'horloge avance jusqu'à la date du premier évènement de la liste. L'entité concernée est alors traitée en PME, jusqu'à ce que toutes les actions ne nécessitant pas de faire avancer l'horloge soient exécutées, et ainsi de suite.

Le terme de calendrier n'est pas universellement adopté par tous les éditeurs de progiciels. Par exemple, Automod parle de *Delay list*, *User-managed List*, *Current Events List* et *Future Events List*, ce qui revient à séparer le calendrier en plusieurs parties distinctes mais regroupant au final les mêmes informations. Toutefois, le mode de fonctionnement reste le même. La suite de ce document ne parlera donc que de calendrier par souci de clarté.

La Figure 6 illustre l'évolution du calendrier sur une simulation simple. Le modèle est présenté dans le cadre a). Il est écrit suivant la sémantique Arena. Trois entités sont créées avec un intervalle de 5 unités de temps (ut) par le module *Create* (date de création de la première entité : t=5ut). Chaque entité passe ensuite dans le module *Delay*, et y passe une activité de 7 ut. Elle est ensuite détruite et sort du modèle par le module *Dispose*.

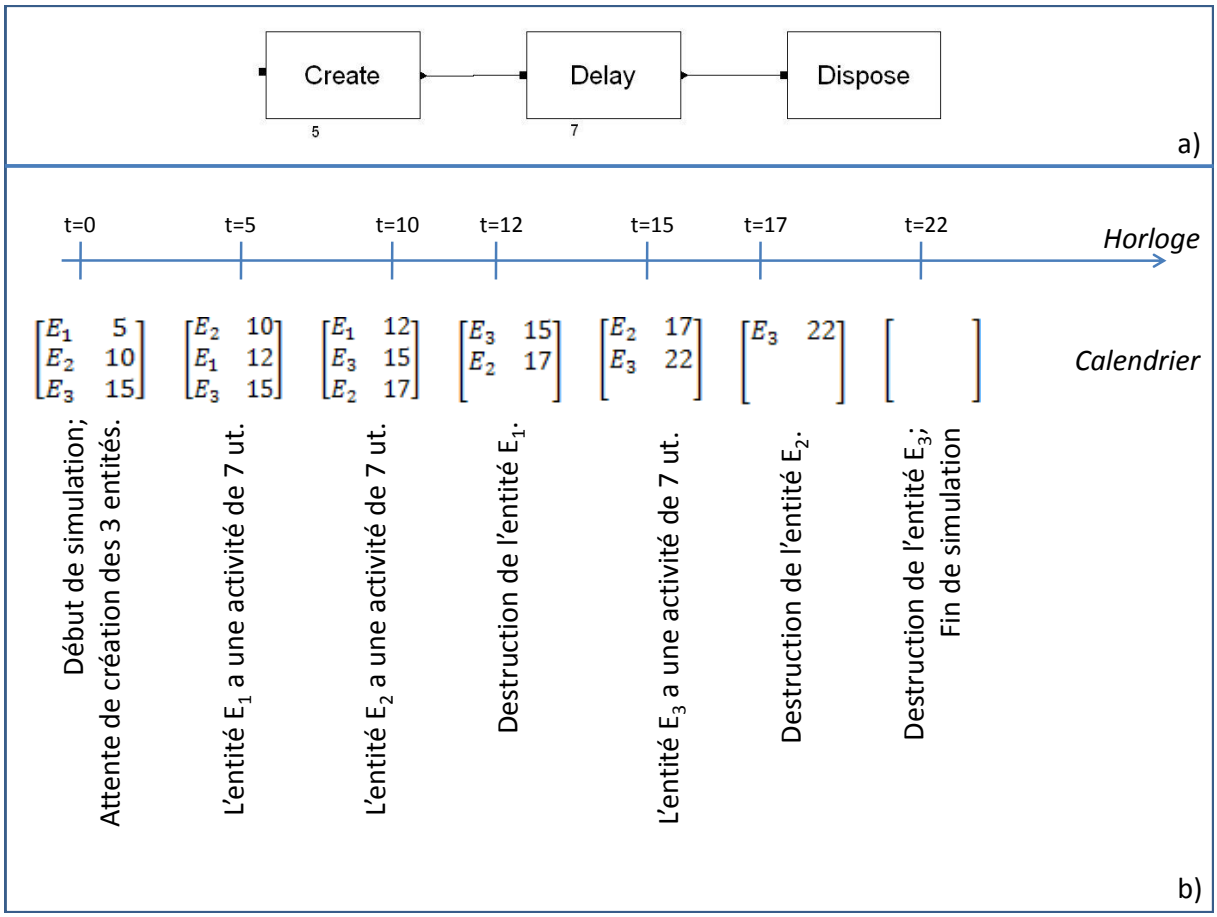

Figure 6 L'évolution du calendrier des évènements

Le cadre b) montre l'évolution du calendrier des évènements au cours du temps. La première colonne du calendrier correspond à l'identification de l'entité concernée, et la deuxième colonne la date à laquelle celle-ci va évoluer la prochaine fois. À t=0, c'est-à-dire au début de la simulation, il n'y a pas encore d'entité dans le modèle. Toutefois, le module *Create* inscrit toutes les entités qu'il doit créer dans le calendrier (ici trois). La PMH fait alors passer l'horloge à t=5. Lors de la PME suivante, l'entité E_1 passe du module *Create* au module *Delay*, et est donc inscrite dans le calendrier avec une date d'évolution de (5+7=12 ut).

À t=22, la dernière entité du calendrier est détruite. Celui-ci est donc vide, et la PMH ne peut donc pas s'effectuer. La simulation s'arrête alors.

Ce mode de fonctionnement permet aux progiciels de simuler de longues périodes en un temps réduit : contrairement au fonctionnement synchrone, la vitesse de simulation n'est pas liée à la période choisie, mais au nombre d'évènements à traiter par unité de temps. Toutefois, dans les deux cas, cela impose au concepteur du modèle une bonne connaissance du progiciel pour lever toute ambiguïté, notamment au niveau des évènements simultanés (même date d'occurrence) – évènements apparaissant lors de la même période en fonctionnement synchrone.

I.2.2 Langages et progiciels de simulation

La liste des langages de simulation et des progiciels associés qui ont pu être utilisés au cours des trente dernières années est relativement longue. Quelques travaux ont de ce fait été réalisés pour comparer ces différentes alternatives sur des critères objectifs. Parmi ceux-ci, il est possible de citer [Banks, 1996] ou encore [Klingstam et Gullander, 1999] dont le tableau de synthèse est proposé en Annexe 1. Les travaux de [Law et Kelton, 1991] ont permis d'extraire les différents critères de choix d'un langage selon l'entité en charge de ce choix. Ainsi, si c'est une entreprise qui a ce choix à faire, ses critères de choix seront :

- La compatibilité entre le langage de simulation et le système informatique de l'entreprise ;
- Le coût d'installation et de maintenance du langage ;
- Le nombre d'études de simulations à effectuer ;
- Le type des systèmes à simuler ;
- La qualité de la documentation ;
- La facilité d'apprentissage du langage ;
- Les performances temporelles du langage ;
- La flexibilité et la puissance du langage.

À l'inverse, si le choix revient à un analyste pour une application particulière, ses critères de choix seront plus spécifiquement :

- La disponibilité du langage dans l'entreprise ;
- La nature du problème à résoudre ;
- La connaissance du langage par l'analyste ;
- Le temps disponible pour l'étude et la programmation ;
- La facilité qu'a un non-spécialiste (chef de projet, client, utilisateur) pour comprendre le modèle.

Il est à noter une différence sensible avec les critères proposés par [Klingstam et Gullander, 1999], notamment due à la prise en compte de la subjectivité des décisionnaires. De plus, ne connaissant pas la pondération que chacun pourra assigner à ces critères, nos travaux ont été réalisés dans l'optique d'être applicables à un maximum de langage ou de progiciel, moyennant le fait que ceux-ci aient certaines caractéristiques qui seront développées au cours de ce document. Parallèlement, nous verrons dans les chapitres suivants que ces travaux ont nécessairement abouti à une validation sur deux progiciels

concurrents : *SIMAN Arena* et *Delmia Quest*. Nous proposons donc ici quelques éclaircissements sur les possibilités offertes par ces deux progiciels uniquement.

I.2.2.1 Quest

Quest, de par sa structure, permet de modéliser le système étudié en se calquant sur sa géométrie [Hugan, 1994]. Pour se faire, l'utilisateur positionne sur une surface plane représentant le sol de l'atelier les représentations géométriques 3D des éléments constitutifs de celui-ci à leur position exacte. Ce progiciel est particulièrement orienté vers la modélisation de systèmes manufacturiers, comme ont pu l'être Witness ou Automod II avant lui[Fegan *et al.*, 1991]. Les éléments dont nous parlons sont donc des machines, des stocks ou des convoyeurs par exemple. Pour les éléments mobiles (opérateurs, chariots autoguidés AGV, etc.), un circuit peut, de plus, être ajouté pour permettre un déplacement précis et réaliste (Figure 7).

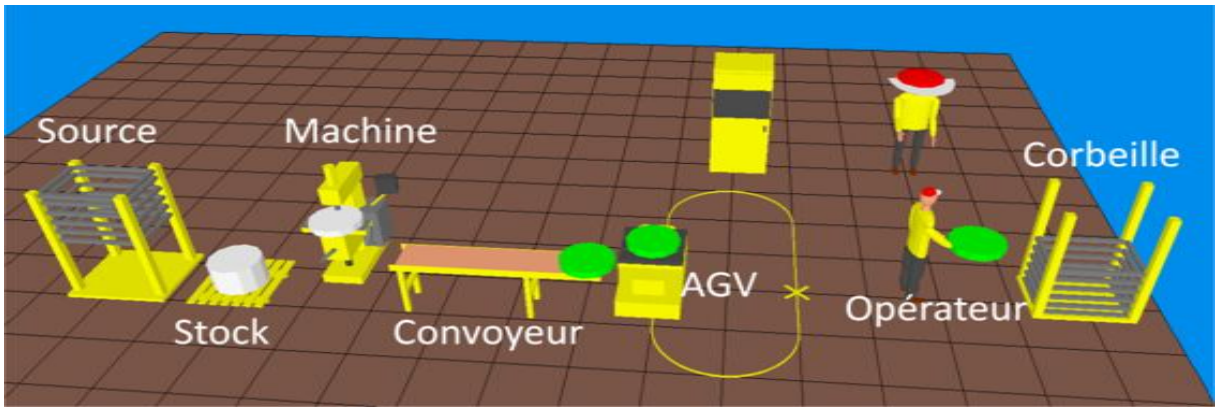

Figure 7 Représentation d'une ligne de production sous Quest

Quest utilise une méthode orientée objet pour la définition des éléments. L'utilisateur commence par définir une classe d'éléments ayant une logique et une géométrie commune. Cette classe a différents paramètres et est instanciée un nombre de fois déterminé à l'avance. Chacune de ces instances, qui est alors un élément, sera placée dans le modèle géométrique.

Ces éléments sont ensuite reliés aux autres éléments par des connexions logiques (Figure 8). Les connexions relient deux éléments entre eux. La liaison entre deux éléments autorise la circulation des pièces entre ces deux éléments. Les pièces ne pourront pas aller d'un élément à un autre si une des sorties du premier élément n'est pas reliée à une des entrées du deuxième élément.

C'est dans cet environnement, constituant le modèle physique de l'atelier, que se déplacent les produits, appelés *parts*. À chaque élément est associé un ensemble de *logics*. Les logics, qu'il est possible de traduire en français par logiques, sont des programmes qui régissent le fonctionnement des éléments. On peut distinguer deux parties dans un élément Quest :

- L'entrée dans laquelle est exécutée la *process logic* ;
- La sortie dans laquelle est exécutée la *route logic*.

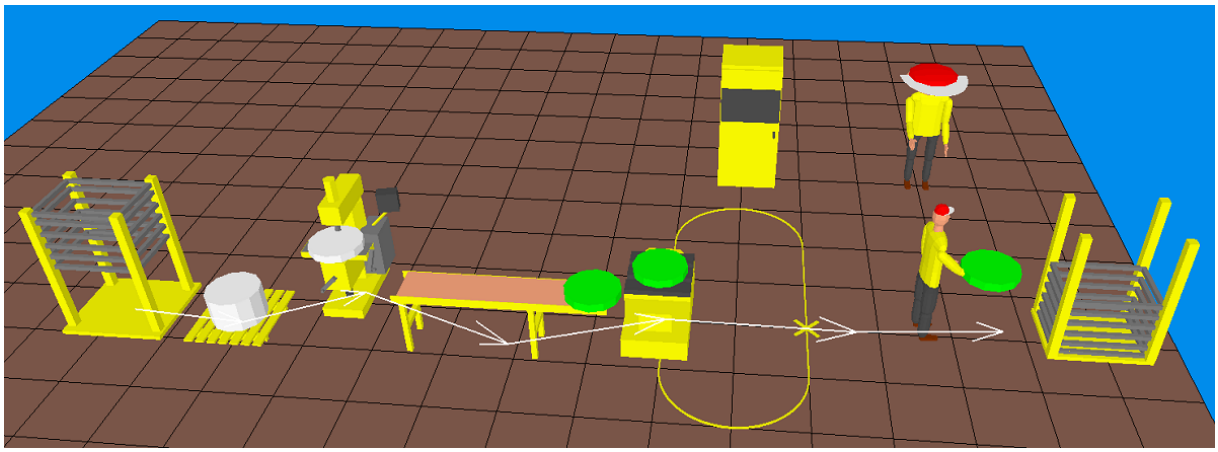

Figure 8 Connexions sous Quest

La process logic contrôle l'entrée des pièces et applique les transformations définies dans les processus sur les pièces. La route logic est exécutée sur chaque pièce qui sort de la process logic. Cette logique sert à choisir la destination de sortie de la pièce. On peut schématiser un élément standard suivant le principe de la Figure 9, où les *inputs* et les *outputs* représentent les différents points d'entrée et de sortie de parts dans l'élément.

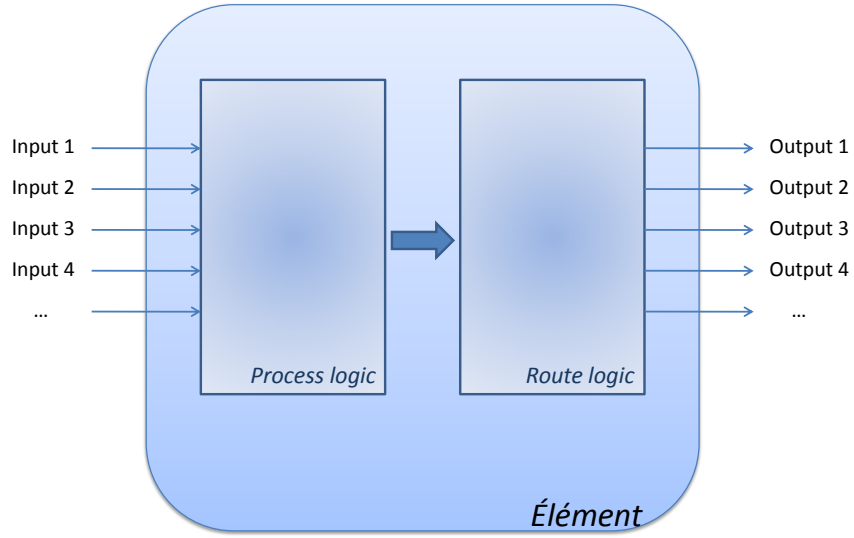

Figure 9 Schématisation d'un élément standard Quest

Dans le cas général ces deux logics sont activées séquentiellement, mais dans le cas où il n'est pas souhaitable d'attendre que l'élément suivant soit libre avant d'accepter une nouvelle pièce, il est possible de les exécuter simultanément.

A ces deux logiques principales, il est possible d'ajouter d'autres logiques spécifiques :

- La *request logic*, utilisée pour appeler des pièces dans un système en flux tiré ;
- La *queueing logic*, servant à régir la loi de stockage d'un stock (FIFO, LIFO, etc.). Cette logique n'est disponible que sur les classes de stocks ;

- L'*init logic*, exécutée une fois au début de la simulation ;
- etc.

Ces logiques sont toutes définies au niveau des éléments. Dans certains cas, il est utile de définir des logiques au niveau du modèle global :

- L'*init logic* est l'équivalent de celle que l'on trouve sur les éléments, mais est exécutée par le modèle, lors du lancement de son exécution ;
- La *sim logic* est exécutée à intervalles réguliers par le modèle ;
- La *term logic* est exécutée lorsque l'on arrive à la fin du temps de la simulation (elle n'est donc pas exécutée si l'on interrompt la simulation avant le temps programmé) ;
- L'*action logic* est exécutée juste avant ou juste après chaque évènement du simulateur.

Pour permettre une modélisation rapide, toutes ces logics ont un comportement établi par défaut. Il est toutefois possible de les personnaliser pour augmenter la fidélité du comportement du simulateur par rapport à celui du système étudié en réécrivant le programme associé en langage SCL (Simulation Control Language). Ce langage de simulation propriétaire permet d'une part le contrôle des éléments de la simulation, et d'autre part l'interfaçage avec l'utilisateur et le monde extérieur. Un second langage est disponible sous Quest : le BCL (Batch Control Language). Ce langage est utilisé pour contrôler Quest. On peut, à l'aide de celui-ci :

- Lire un modèle ;
- Modifier un modèle (ajout, suppression, modification d'éléments) ;
- Exécuter une simulation ;
- Interroger les résultats de simulation ;
- Enregistrer un modèle ;
- Contrôler les caméras ;

De nombreuses méthodes existent pour exécuter du code BCL dans Quest :

- Depuis le SCL ;
- Par une boite de dialogue ;
- Par un bouton personnalisé ;
- En chargeant un modèle Quest avec un script BCL : il est possible, au chargement d'un modèle Quest, de lui imposer un script BCL à exécuter ;
- En chargeant Quest avec une Socket : il est possible de lancer le logiciel avec un port sur lequel écouter. Toute donnée reçue sur ce port sera une commande BCL exécutée par le logiciel.

On peut alors trouver de nombreuses applications à ces fonctionnalités :

- Génération automatisée de modèles ;
- Création d'un modèle d'optimisation contrôlé par un programme externe ;
- Modifications du modèle en cours de simulation en SCL ;
- Envoi de données depuis un programme externe.

Comme on peut le constater, les points forts de Quest sont, d'une part, sa puissante animation graphique, et d'autre part sa forte compatibilité avec les systèmes de production.

Ce dernier point assure une prise en main plus conviviale pour les non-experts. Les principaux points faibles de Quest se situent au niveau des faibles possibilités de communication avec l'extérieur et au niveau de la programmation avancée en SCL et BCL, qui requièrent un haut niveau d'expertise pour affiner le comportement des éléments et plus généralement des modèles.

I.2.2.2 SIMAN Arena

Nous avons vu dans le paragraphe précédent que Quest était très orienté vers les systèmes de production. Arena de son côté, se basant sur le langage SIMAN qui ressemble à un langage de programmation classique, n'a pas cette spécificité. SIMAN est un langage général de simulation de flux. De ce fait, la construction de modèles de systèmes de production est plus complexe. Par contre, Arena permet de modéliser plus facilement d'autres types de systèmes, tels que les réseaux informatiques par exemple, et est beaucoup plus efficace pour modéliser le système d'informations inhérent à beaucoup de systèmes de production.

Au contraire de Quest, la modélisation sous Arena est entièrement centrée sur le modèle logique. La programmation est très orientée processus. Le modèle est construit en décrivant un circuit à travers lequel vont circuler des entités. Les entités sont les objets de base, circulant dans le modèle. Ce modèle/circuit est construit par la liaison graphiques de modules, dont les entrées sont à gauche et les sorties à droite. Chacun de ces modules représentent une instruction élémentaire du langage SIMAN. Lorsque tous les blocs nécessaires sont reliés entre eux, le programme ressemble à un gigantesque schéma-bloc. Par abus de langage, il n'est donc pas rare de parler de blocs en lieu et place de modules. Des modules sont également nécessaires pour la déclaration des objets du modèle (ressources, variables, attributs, etc.). À la compilation du modèle, avant exécution, le programme SIMAN est généré, ce qui permet de visualiser, si nécessaire, le programme sous forme entièrement textuelle.

Ces modules font partie des bibliothèques *Blocks* et *Elements*, qui sont les seules que nous avons utilisées dans ces travaux, exception faite de celles que nous avons construites. En effet, un des principes de base d'Arena est de permettre la création très puissante de bibliothèques, où chaque nouveau module est une agrégation de plusieurs autres modules. Ainsi, plusieurs bibliothèques sont prévues par défaut dans Arena, permettant de construire plus rapidement et plus facilement des modèles. Nous avons choisi de ne pas les utiliser car, en s'éloignant des primitives de base de SIMAN, le comportement de ces nouveaux modules devient moins bien connu. L'utilisateur a le choix de construire ses propres bibliothèques, adaptées à son besoin et à son secteur d'activité. Nous verrons dans le Chapitre IV de ce document qu'une bibliothèque a notamment été construite lors de ces travaux.

Tout programme doit commencer, d'une manière ou d'un autre, par la création d'une ou plusieurs entités. Une entité est un objet qui évolue dans les différents blocs fonctionnels constituant le modèle du système. Elle correspond en général à un objet concret, par exemple, une personne ou une pièce dans un atelier, mais peut également être utilisée pour transmettre une information. Le déplacement des entités au sein des différents blocs provoque un changement d'état du modèle de simulation, ce qui est analogue aux déplacements des jetons dans un modèle en Réseau de Petri. Le principe de fonctionnement du logiciel ARENA est de suivre chacune des entités évoluant d'un bloc fonctionnel vers un

autre dans le modèle, de sa création à sa destruction. L'ordonnancement dans le temps des différents évènements rattachés à l'évolution des entités dans les blocs constituant le modèle se fait au travers d'un échéancier. Quand une entité est introduite dans un bloc fonctionnel, elle déclenche/active le « service » qui lui est associé, ce qui provoque une modification de l'état du modèle.

Étant très lié et fortement compatible avec les logiciels Microsoft, Arena intègre la possibilité d'inclure des programmes VBA dans les modèles. Ces programmes peuvent être déclenchés :

- Soit par le passage d'une entité dans un bloc spécifique VBA.
- Soit lors d'évènements spécifiques prédéfinis au niveau du modèle, tels que le début d'une réplication ou l'occurrence d'une erreur par exemple.

Cette fonctionnalité ouvre la possibilité d'inclure simplement des algorithmes assez complexes à modéliser en SIMAN, de faire de la programmation orientée objet, de la programmation évènementielle, ou encore ouvre des fonctionnalités de communication avec d'autres applications grâce à l'intégration de bibliothèques de liens dynamiques (dll pour Dynamic Link Library). Il est à noter que le langage SIMAN est déjà capable de communiquer avec des bases de données ou d'échanger avec une autre application sous la forme de fichiers textuels.

Arena offre également la possibilité d'animer graphiquement l'évolution du programme SIMAN. Cette animation permet de faire évoluer des indicateurs selon l'évolution des variables du programme. Ces variables peuvent être décomposées en trois groupes :

- Les variables liées au modèle et aux objets le composant, telles que la date courante de la simulation ou le taux d'utilisation d'une ressource ;
- Les variables globales, déclarées par le concepteur du modèle, et concernant l'ensemble du modèle. Leur évolution est liée à celle de l'ensemble des entités ;
- Les attributs, liés aux entités, et qui sont modifiés par l'évolution de l'entité concernée.

Contrairement à Quest, l'animation est plane. Elle permet de représenter le parcours des entités dans le programme, le déplacement de transporteurs sur un circuit ou encore la valeur numérique de variables par exemple. Cette animation est présente pour la mise au point, puis pour la communication du modèle, mais n'a aucune influence sur le programme. Ainsi, lorsqu'un modèle a été mis au point avec une animation graphique gourmande en ressources, il est possible de simplement supprimer les éléments d'animation graphique pour accélérer la simulation sans modifier le comportement du simulateur. Une fonctionnalité de simulation sans animation graphique est toutefois disponible pour accélérer la simulation sans supprimer les éléments d'animation.

I.3 La simulation en ligne

La société UPS, spécialisée dans la livraison de colis à travers le monde, a utilisé la simulation sous Arena pour réduire les temps de chargement-déchargement de ses avions à

l'aéroport de Louisville Air Pack[1]. Le modèle construit simule le déchargement des avions, le transport des colis à travers l'aéroport, le tri de ces colis et le chargement sur les nouveaux avions. De plus, les opérations de maintenance et de ravitaillement sont intégrées pour déterminer les besoins en équipement et personnel. Des rapports de simulation personnalisés, incluant les horaires prévus des avions, les besoins en personnel et en équipement, permettent de planifier avec suffisamment de justesse l'activité de l'aéroport sur un horizon inférieur à 24 heures. Les conclusions tirées de l'analyse de ces résultats ont éliminé le besoin ressenti d'augmentation du niveau d'équipement par une meilleure répartition des moyens. L'économie annoncée par UPS s'élève à plusieurs millions de dollars et correspond à une des applications classiques de la simulation : l'implantation d'une nouvelle unité.

Toutefois, le modèle continue ici d'être utilisé pour tester des stratégies opérationnelles avant de les implanter (ordonnancement des vols, répartition des tâches, etc.). Les opérateurs peuvent désormais simuler une nuit de travail complète (6 heures) en quelques minutes, ce qui laisse la flexibilité suffisante pour les changements de dernière minute sans causer de retard dans le trafic aérien. La simulation devient ainsi un outil d'aide à la décision pour le pilotage d'une unité complète.

Comme le montre cet exemple, certains utilisateurs industriels ont senti le besoin de passer de simulations à long terme à des simulations à court terme. Cette évolution s'explique par plusieurs facteurs : la volonté de réutiliser les modèles de simulation réalisés à la conception de grands systèmes, la nécessité d'avoir un outil de prévision à disposition des opérateurs, etc. De ce fait, plusieurs travaux ont été réalisés dans ce sens, souvent liés à des problématiques industrielles. Elles se regroupent principalement dans le cadre de la « simulation en ligne ». Ces travaux ont principalement été réalisés depuis le milieu des années 1990.

I.3.1 Définition

En général, les prévisions que l'on réalise par simulation peuvent être classées en trois catégories : long, moyen et court terme [Taylor III, 1996]. Les prévisions à long terme englobent généralement des périodes d'une à plusieurs années. Elles sont principalement dédiées aux constructions de nouvelles usines, aux changements de marché, ou à la consolidation des plans financiers. C'est dans ce cadre que la simulation utilisée en phase de conception est classiquement implantée. Le moyen terme s'étale généralement sur une période allant de un mois à une année. On parle plutôt ici de production annuelle prenant en compte les variations saisonnières de la demande. La séparation entre moyen et long terme est toutefois très souvent bien plus floue que celle mentionnée ici.

Les prévisions à court terme sont plutôt reliées au futur immédiat et traitent les opérations au jour le jour, telles que les besoins quotidiens en personnel ou les variations de demande à la semaine. Les prévisions à court terme excèdent rarement les deux mois.

À la fin des années 1990, plusieurs réflexions ont été menées sur l'avenir de la simulation. L'idée motrice de ces réflexions a été de passer d'une simulation destinée à la conception d'un système de production à une simulation pour l'aide au pilotage. Ce passage

[1] Voir http://www.arenasimulation.com/pdf/UPSERV-AP001A-EN-P.pdf

a bien évidemment d'abord été envisagé pour les grandes installations, où des modèles de simulation avaient déjà été construits. Dans [Castagna *et al.*, 2001], les auteurs répondent à une demande de l'industrie aéronautique qui voulait un simulateur permettant d'avoir une vision claire du comportement de leur unité de production sur un horizon court (une journée). Le concept associé a notamment été exposé dans [Davis, 1998] sous le nom de *simulation en ligne*.

D'un autre côté, en 1999, [Davis, 1999] prédit qu'une des directions futures d'évolution de la simulation sera la simulation basée sur les technologies web. Ces simulations permettront à tous les acteurs allant du fournisseur en matière première au vendeur de magasin, d'interagir sur un même ensemble de modèles à travers une simple interface, programmée par exemple en java. Partant de cette idée, il énonce un certain nombre de perspectives de développements, ainsi que les verrous scientifiques et technologiques à éliminer pour y parvenir. Certains de ces verrous sont identiques dans la perspective de la simulation en ligne. Ainsi, l'auteur énonce notamment qu'il n'existe pas de procédures établies qui permettent aux utilisateurs d'interfacer efficacement avec les données temps-réel qui seront générées par les systèmes de management en ligne. De plus, le rôle le plus efficace de l'humain dans cette configuration n'a pas encore été défini.

Pour qu'un modèle de simulation soit dit en ligne, il doit tout d'abord être connecté avec le monde réel. Cela signifie que chaque fois qu'une simulation débute, le modèle doit être initialisé avec l'état actuel du système réel. La seconde exigence est que les résultats de simulation doivent être obtenus avant une date limite. Le moteur de simulation doit donc être suffisamment rapide pour délivrer les résultats en un délai court pour qu'ils puissent être utilisés dans le processus de décision. Pour [Drake et Smith, 1996], les systèmes de simulation en ligne incorporeront à terme deux puissantes caractéristiques :

1. la capacité à prévoir correctement le comportement futur de l'atelier compte-tenu de son état actuel
2. la capacité d'émuler et/ou supplanter la logique de commande d'un système manufacturier.

La simulation en ligne a clairement pour vocation le « court terme ». Il n'est pas envisageable de quantifier précisément cet horizon, puisque cela dépend en grande partie de la dynamique du système. En effet, un horizon court dans le bâtiment peut être des centaines de fois plus long que celui de l'industrie des semi-conducteurs par exemple.

D'après [Hanisch, Tolujew et Richter *et al.*, 2003], l'incertitude sur les résultats de simulation est corrélée à la longueur de la simulation : plus l'horizon de simulation est grand, plus les résultats sont faussés. Il est possible d'identifier deux principales raisons de cette différence dans les résultats :

1. Les sources de contingence, d'incertitude et d'arbitraire [Roy, 1989] : la modélisation du système prend forcément en compte des approximations qui prennent d'autant plus d'importance que l'horizon de simulation est long (plus de détails seront apportés à ce point au Chapitre II).
2. Lors d'une simulation sur un terme court, on ne tient pas compte de la nature stochastique des évènements. En effet, il faut que cet évènement ait une occurrence suffisamment fréquente relativement à l'horizon de simulation pour que les résultats

soient significatifs. Or, plus l'horizon de simulation est long, plus le système réel (ou tout du moins une partie) a une forte probabilité de tomber en panne par exemple, ce qui amène une différence entre le comportement réel du système de production, et son comportement simulé.

Les auteurs définissent les types d'utilisations concernées par les bénéfices de la simulation en ligne. Premièrement, la simulation peut aider les planificateurs de ressources à déterminer les tailles de lots d'ordres de fabrication (OF), les dates de lancement et les calendriers de disponibilités de ressources en incorporant des contraintes d'ordonnancement simulées dans leur prise de décision. Deuxièmement, une fois que les ordres sont lancés, la simulation offre aux ordonnanceurs la possibilité d'évaluer en temps-réel les stratégies pour séquencer les opérations au travers des ateliers. Dans des environnements manufacturiers dynamiques, il pourrait être avantageux de changer la manière dont l'atelier est commandé à certaines dates. Troisièmement, le fait que la simulation courre en parallèle du système réel en temps-réel permet au système de simulation de suivre l'état actuel et d'envoyer des feedbacks aux fonctions d'ordonnancement sur les performances de l'ordonnancement actuel. Quatrièmement, en émulant les conditions actuelles de l'atelier, la simulation peut aider les fonctions vente et marketing à prédire avec précisions les délais de production et indiquer les dates de livraison (due dates) aux clients. Utiliser la simulation de cette manière peut diminuer la quantité d'en-cours et participer à l'amélioration de la réputation de l'organisation en améliorant sa capacité à respecter les dates de livraison promises.

Au vu de ces définitions et réflexions, on peut conclure que :

- Le modèle de simulation en ligne doit être connecté avec le système réel ;
- Le modèle doit fournir des résultats de simulation dans une fenêtre de temps donnée ;
- La valeur de l'horizon de simulation, relativement court, doit être particulièrement étudiée pour ne pas souffrir de la non-prise en compte des évènements stochastiques ;
- La simulation en ligne peut toucher l'ensemble des acteurs d'une industrie, des couches marketing à l'opérateur de production.

I.3.2 La simulation en ligne et le système réel

L'objectif à atteindre est de réussir à initialiser les simulations en ligne sur l'état actuel du système de production réel. D'après [Fowler et Rose, 2004], les problèmes suivants doivent d'abord être résolus :

- une définition claire du terme « état du système » est manquante : avant de commencer à collecter des données, nous devons savoir exactement de quel type de données nous avons besoin pour obtenir une image nette du système.
- Il y a un manque de données : les données nécessaires pour la simulation ne sont pas disponibles et ne peuvent pas être déduites automatiquement à partir d'autres informations sur le système
- La qualité des données est trop basse : par exemple, le modèle de simulation nécessite un histogramme quand l'outil ne peut fournir que des valeurs moyennes d'un des paramètres.

- La fréquence de mise à jour est trop faible : l'outil peut générer des rapports après un intervalle donné et long, disons une fois par jour.

Toujours d'après [Fowler et Rose, 2004], les informations doivent être très précises et tous les problèmes listés ci-dessus doivent être résolus ou bien une estimation de l'erreur sur la réponse induite par une information fausse, manquante ou en retard doit au moins être fournie. Pour pouvoir reproduire le comportement réel du système, le modèle de simulation doit avoir les fonctionnalités MES appropriées. Cela peut être réalisé par l'intégration d'une copie du MES dans le modèle ou par une reconstruction d'une partie du MES dans le modèle. Dans ce cas, le MES reconstruit n'aura qu'un comportement approximé du vrai MES.

[Gonzalez et Davis, 1998] décrivent comme plus grande difficulté la relation existant entre la commande du système, le système réel et la simulation. En effet, le but n'est ici que d'avoir une relation entre le système réel et la simulation. Or, la commande a une grande influence sur le réel, et connaît forcément une partie de celui-ci pour son exécution. Le problème est que, souvent, dans les applications de commandes commerciales, les structures de données internes sont cachées à l'utilisateur. Et même si elles sont accessibles, l'utilisateur doit encore déterminer la relation qu'il existe entre la donnée et sa signification dans l'état du système réel. De ce fait, ils préconisent l'utilisation d'un dictionnaire complet de données commun entre la commande et la simulation. Mais étant donné que les solutions commerciales ne leur donnent pas satisfaction de ce côté, ils concluent à la quasi-impossibilité d'initialiser les simulations à l'état du système réel en passant par des solutions commerciales. Pour parer à cela, ils préconisent l'utilisation du même modèle pour la commande et la simulation, en particulier pour les fonctions de transition des états du système. Les progiciels de simulation n'étant pas toujours capables de modéliser efficacement de la même manière que les progiciels de commande, il convient donc de développer conjointement les deux outils. Ceci impose de n'appliquer la simulation en ligne qu'à des systèmes qui ne sont pas encore en production, sous peine de devoir recommencer complètement la commande sous un modèle éventuellement totalement différent. De plus, cela empêche pratiquement l'utilisation de logiciels de simulation commerciaux pour la simulation en ligne. Bien qu'une parfaite connaissance de la commande est probablement indispensable à la bonne application de la simulation en ligne sur un système, nous partons de l'hypothèse que le développement des outils de simulation en ligne ne doivent pas influer sur la conception de la commande, ou tout du moins le moins possible. De plus, au vu des besoins industriels de réutilisation de modèles, il semble également préférable l'utilisation de logiciels commerciaux pour démontrer la faisabilité de l'approche avec les outils qu'ils utilisent éventuellement déjà dans les phases de conception. Nos travaux s'inscrivent donc dans cette logique de développement autonome de la simulation en ligne sur des logiciels commerciaux.

I.3.3 Quelques exemples de développements de la simulation en ligne

Les auteurs précédents ont monté une plate-forme expérimentale pour tester la validité de leur approche. Cette plate-forme (voir [Gonzalez et Davis, 1997]) permet la commande et la simulation du système en modélisant explicitement les interactions entre contrôleurs pour décrire les mécanismes de transition d'états qui gouvernent l'évolution d'un système distribué. De ce fait, ils se basent en grande partie sur les échanges de

messages sur le réseau pour déterminer l'état du système. De nouveau, cette approche élimine beaucoup de systèmes de l'utilisation de la simulation en ligne.

Dans [Rogers et Flanagan, 1991], les auteurs utilisent une simulation en ligne pour aider les contrôleurs aériens à diminuer les problèmes de trafic et augmenter la sécurité. Les données temps-réel provenant des centres de contrôle aérien sont utilisées pour construire une base de données indiquant l'état actuel du réseau de transport. Ces seules données sont utilisées pour initialiser les simulations (appelées *faster-than-real-time*) qui sont faites pour explorer l'impact des décisions prises par les contrôleurs aériens dans le but de déterminer les trajectoires. L'initialisation des simulations dans ce cas est un cas très particulier. En effet, le triptyque position-trajet-vitesse de chaque élément contrôlé est très bien connu et défini à tout instant. De plus, très peu de perturbations relativement à la dynamique du système ne viennent perturber les prédictions. De ce fait, la relation entre objets du système réel et objets simulés est presque immédiate, et donc l'initialisation également.

Devant la difficulté d'initialiser les simulations en ligne, certains auteurs ont diminué le niveau d'expertise de leur solution. Ainsi, [Kouiss et Pierreval, 1999] proposent une application basée sur une simulation en ligne. Cette application a pour objectifs d'aider à surveiller le système par l'analyse et la comparaison entre les données réelles et simulées et d'aider l'opérateur à prendre des décisions spécialement concernant l'ordonnancement. Pour montrer la validité de l'approche, ils ont développé cette application sur un FMS « taille réelle » installé à l'Institut Français de Mécanique Avancée (IFMA). La collecte d'informations sur le terrain se résume ici à une mise à jour des paramètres intrinsèques de la simulation (durées de production etc.) pour affiner cette dernière. Toutefois, les simulations partent toutes d'un état initial identique généralement considéré vide et inactif (*idle* et *empty*). Ici, la simulation est « en ligne » dans le sens où l'on initialise automatiquement les données du simulateur à partir du système réel, mais on ne traite pas du problème de l'initialisation de l'état du simulateur.

De la même manière, [Peters et Smith, 1998] présentent un système de commande appelé TSCS développé par TAMCAM, utilisé pour étudier les avantages (et désavantages) de la simulation en ligne dans l'architecture de commande. La simulation est utilisée pour faire des prévisions destinées à évaluer des politiques de commande alternatives : ils appellent cela « travailler en *Look Ahead »*. L'implémentation est faite sur Arena équipé du modèle temps-réel (Arena RT), Access, Visual Basic et Visual C++. La modélisation de leur système réel considère que les durées opératoires sont toutes basées sur une loi de distribution. La simulation qu'ils font tourner en parallèle du système réel sert à collecter les différentes durées de production. L'historique de ces durées sert alors à évaluer les coefficients des distributions aléatoires, coefficients qui sont réutilisés pour la simulation en mode *Look Ahead*. De ce fait, ils n'ont qu'une vue sur « Comment le système fonctionne en ce moment ? », car ils ne prennent pas en compte toutes les données nécessaires pour savoir « Dans quel état est le système en ce moment ? », données qui sont les plus compliquées à obtenir. Les conclusions qu'ils peuvent tirer sur l'utilité de la simulation en ligne n'ont donc pas une grande portée puisque leur simulation est finalement assez loin de ce que l'on pourrait attendre d'une simulation en ligne complète.

Même si la finalité du système est différente de la notre, l'implémentation de la solution envisagée est intéressante. La Figure 10 présente l'implémentation qu'ils envisagent. Le concept et surtout les finalités de la simulation temps-réel sont relativement

différents dans nos travaux, mais le découpage entre une simulation temps-réel qui suit la production pour fournir des données à une simulation en mode *Look Ahead* sera développé dans la suite de ce document.

[Ramakrishnan *et al.*, 2002] utilisent également un modèle type *look-ahead* et un modèle appelé *RT* lancé en temps réel et connecté au système de production. Ce modèle RT sert à comparer les fins de tâche programmées des opérations d'une SCM avec celles définies dans la base de données. S'il y a une déviation, une simulation *look-ahead* est lancée pour déterminer si la tendance va à l'aggravation et s'il faut alors déclencher des actions correctives. Les modèles sont développés sur Arena 4.0 avec une base de données SQL. À part cela, peu d'informations sont données sur l'implantation pratique de leur solution.

[Wu et Wysk, 1989] ont choisi de discrétiser le temps en intervalles égaux. À chaque début de période *dt*, une simulation est lancée pour tester différentes règles locales d'ordonnancement et décider quelles sont la ou les meilleures selon plusieurs critères (mean flow time etc.). À ce moment, ils choisissent d'appliquer ces règles pendant le prochain *dt*, et recommence *dt* plus tard. L'initialisation ne leur pose pas trop de problèmes, puisque les pièces ne peuvent être au début d'une simulation que dans les machines ou dans des stocks bien définis. Si elles ne le sont pas en réalité, ils en font l'approximation. De ce fait, ils ne semblent pas prendre en compte les temps de transferts, et ne parlent pas de la remontée d'informations depuis l'atelier (notamment sur la taille des stocks et l'acquittement des opérations), puisqu'ils ne font cette simulation que pour servir de benchmark de leur solution. Au final, ils ne décrivent pas l'application de leur solution sur un cas de taille industrielle.

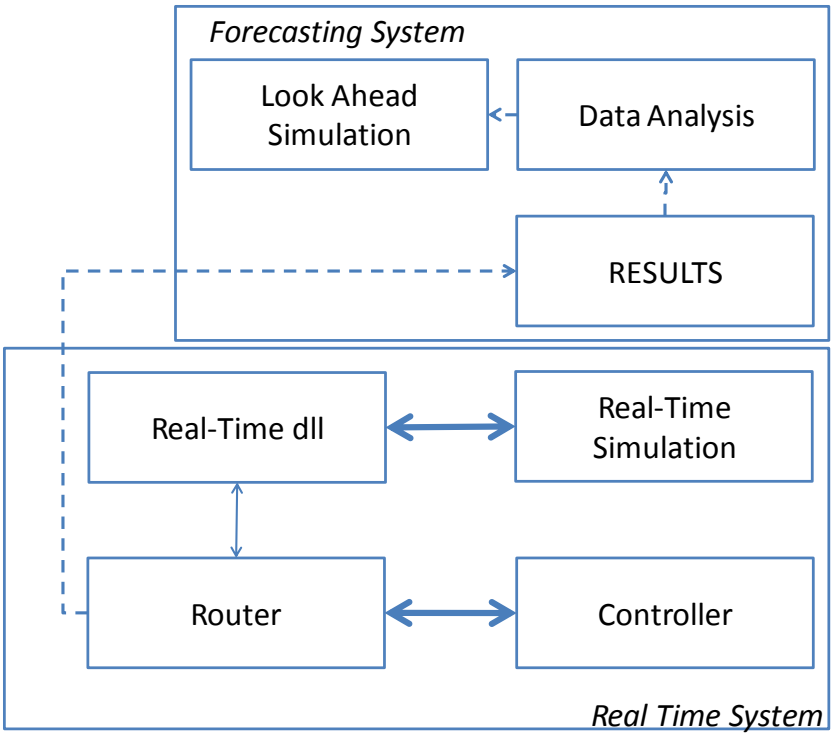

Figure 10 L'implémentation du TSCS [Peters et Smith, 1998]

L'exemple développé dans [Pujo *et al.*, 2004] est un cas particulier d'une autre nature. Cet article traite un exemple de pilotage proactif via simulation. Les auteurs construisent le modèle DEVS d'un atelier comportant plusieurs lignes de type KANBAN enchevêtrées, traitant plusieurs références de produits. L'avantage d'utiliser DEVS est de pouvoir « créer un système de pilotage intégrant en son sein des fonctions de simulation », ce qui rejoint tout-à-fait les propos de [Gonzalez et Davis, 1998] et [Fowler et Rose, 2004] (voir chapitre I.3.2). Le problème de l'initialisation des simulations est évoqué, et les auteurs semblent se diriger vers une initialisation directe du vecteur d'état à partir des informations remontées depuis l'atelier. Cela semble effectivement tout-à-fait envisageable étant donné que l'état global du système est ici presque directement obtenu à partir des niveaux de KANBAN de chaque poste. Cette technique n'est pas toujours utilisable, car l'état du système de production est rarement aussi bien suivi qu'ici avec les KANBAN.

[Iassinovski *et al.*, 2007] développent un système de production permettant, entre autres, l'utilisation de la simulation en ligne dans une scierie. Les auteurs développent en parallèle un outil appelé SDBuilder (System Dynamics Builder) qui permet d'intégrer dans un outil d'aide à la décision des techniques de parcours de graphes, de simulation en ligne, etc. Le but est de découper au mieux les troncs en fonction du carnet de commande actuel pour diminuer les chutes. La validation de leur approche se fait sur le modèle émulé par simulation d'une scierie. Les auteurs ont choisi de développer cette simulation également sur SDBuilder. De ce fait, les auteurs ne démontrent pour l'instant pas la faisabilité de leur approche au niveau du couplage avec le système réel, mais uniquement le couplage entre deux simulations. De plus, leur implantation impose l'utilisation globale de leur solution logicielle, ce qui représente une contrainte forte pour une utilisation industrielle.

Le cas traité dans [Hanisch, Tolujew et Richter *et al.*, 2003] est encore différent. Le papier traite de la simulation de l'évolution du flux de personnes dans un lieu public (gares, aéroports, centres commerciaux, etc.). Le problème est qu'il n'existe pratiquement aucun outil de simulation dans la phase opératoire pour organiser le déplacement de grandes quantités de piétons dans de grands lieux publics. Le but des auteurs est donc de créer une simulation en ligne transparente (l'opérateur n'a pas à paramétrer et démarrer la simulation, ni à analyser des résultats directs de simulation) pour prendre des décisions visant à réguler proactivement le trafic aux heures de pointe. Le prototype doit être capable d'anticiper et de prévenir les problèmes de surpopulation dans les *storages* (magasins, quais de gare etc.) à court terme à partir des flux actuels. Un jeu de simulations est lancé toutes les 5 minutes. Le système d'information mesure le flux de personnes en plusieurs points du bâtiment. A partir de ces mesures, il détermine l'évolution du public dans les 60 prochaines minutes. Des lois probabilistes sur les temps de déplacement entre deux points sont utilisées pour déterminer la dynamique du système. Le prototype est programmé en VBA sur Excel. Le programme estime les entrées et sorties dans chaque *storage* pour déterminer leur nouvelle population. Si une population dépasse un seuil critique, des procédures d'ouverture de portes sont déclenchées pour fluidifier le trafic. Ce prototype est pour le moment totalement offline.

[Kouiss et Najid, 2004] et [Hotz *et al.*, 2006] présentent une utilisation de la simulation en ligne comme détecteur de déviation du système. Une simulation est lancée pour donner une estimation au MES du comportement du système. À chaque instant, celui-ci compare les résultats obtenus à ces estimations, et si l'erreur devient trop importante, une alarme est

déclenchée et le pilote est invité à prendre une décision pour corriger cela. La question du lien avec le réel est traitée différemment. [Kouiss et Najid, 2004] ne font qu'observer le comportement réel et ajuster en conséquence les distributions aléatoires utilisées dans le modèle de simulation. Ce modèle part ensuite d'un état du système vide. [Hotz *et al.*, 2006] de leur côté n'ont pas encore implanté de solution, mais penchent pour l'utilisation massive de RFID longue portée pour connaître l'emplacement exact de tous les véhicules dans l'usine.

Enfin, [Gupta et Sivakumar, 2005] présentent une application de la simulation en ligne en ordonnancement. La simulation est liée à un ordonnanceur dans un CSS (*Conjunctive Simulated Scheduling*), ce qui permet à l'algorithme de s'affranchir de la NP-complexité du cas traité (l'ordonnancement de la fabrication de semi-conducteurs). La simulation est utilisée pour tester l'influence de l'ordonnancement choisi, et cet ordonnancement se sert du calendrier de la simulation comme base de temps de l'algorithme, ce qui permet de diminuer les temps de calcul. Ce papier, relativement récent, expose les avantages et bénéfices théoriques d'une telle méthode, mais pointe également la difficulté d'implantation sur un cas réel, du fait de l'utilisation du concept de simulation en ligne et de sa nécessaire liaison avec le système réel.

I.4 Conclusion

Ce chapitre avait pour but de définir le contexte dans lequel la simulation en ligne se situe. Ce type de simulation a une vocation de prévision sur le comportement à court terme du système de production. Elle a donc une application potentielle intéressante dans le pilotage du système. À l'opposé de la simulation classique, un lien constant avec le système réel est établi pour pouvoir prendre en compte des données récentes et complètes pour initialiser et paramétrer le modèle. Ce lien rend toutefois la conception du modèle et de l'architecture de commande du système plus complexe. Au travers des différents travaux de la littérature, on s'aperçoit donc que ce concept appliqué à la simulation à évènements discrets a été beaucoup évoqué, mais que les applications concrètes sont très rares.

Les travaux présentés dans ce document permettent de répondre aux difficultés évoquées par les différents auteurs de la littérature. Ces réponses concerneront tout d'abord la place relative de l'homme et de la simulation en ligne dans l'architecture de commande du système de production (Chapitre II), puis le lien entre la simulation et le système, de la collecte des informations sur celui-ci à l'intégration de ces informations au simulateur (Chapitre III).

Chapitre II.

La simulation en ligne et l'aide à la décision

Au vu du chapitre précédent, de nombreux auteurs partent du principe établi que l'utilisation de la simulation en ligne aura de grands bénéfices sur le pilotage de la production. Toutefois, les problèmes apparaissant lors de l'application de cette technologie sur des systèmes réels n'ont jamais été clairement listés, et des solutions possibles ont encore moins été proposées.

Nos travaux ont tout d'abord porté sur la relation nouée entre le pilote du système, en charge des décisions affectant le pilotage de la production, et le système d'aide à la décision dans lequel le module de simulation en ligne est sensé être inclus. En effet, de nombreux travaux de la littérature ont porté sur les architectures de commande nécessaire à cette application. Ces architectures sont principalement centrées sur l'aspect technique et incluent très peu l'opérateur humain, ne définissant pas à quel niveau l'humain et la simulation coopèrent. Dans la première partie de ce chapitre, nous nous attacherons à positionner les rôles de l'humain et de la simulation en ligne dans le pilotage d'un système de production.

Dans la deuxième partie de ce chapitre, nous avons travaillé à l'application technique de la simulation en ligne. Le point crucial, comme il a été montré au chapitre précédent, se situe au niveau de l'initialisation des simulations, celles-ci devant le plus possible prendre en compte l'état du système de production. Ce point se révèle en outre être l'un des plus compliqués à réaliser. La seconde partie de ce chapitre se décompose donc en deux parties. Dans la première nous essaierons d'évaluer l'impact de cette initialisation à l'état du système réel par rapport à une initialisation classique, puis nous présenterons les enjeux de cette initialisation du point de vue de la disponibilité des informations autant que de l'intégration à l'intérieur d'un simulateur.

II.1 L'aide à la décision

II.1.1 Résolution de problèmes dans le pilotage des systèmes de production

Depuis le début des années 1980, les rôles de l'humain se déplacent des tâches opérationnelles vers celles de surveillance et de décision : surveillance, détection, anticipation, diagnostic, prévention et traitement des problèmes. Ces problèmes concernent désormais l'aide à la décision pour le pilotage ou le management de la production ou encore la surveillance, le diagnostic ou la reconfiguration du système de production en cas de perturbation. Dans [Cauvin, 2005], l'auteur définit une perturbation comme étant un évènement imprévu dont l'occurrence gêne la réalisation des objectifs de production du système. Ces perturbations apparaissent principalement sur cinq éléments d'un système de production :

- Disponibilité des ressources internes (machines indisponibles, etc.);
- Approvisionnement (retards de livraison, etc.);
- Demande (succès surprenant d'un produit, etc.);
- Information (erreurs dans la retranscription de données, etc.);
- Décision (données mal prises en compte, etc.).

Bien évidemment, dans cette notion de perturbation, nous englobons tout ce qui concerne les pannes et autres indisponibilité de ressources, mais également l'arrivée d'un nouvel ordre de fabrication ou l'ouverture d'un nouveau poste de travail par exemple. L'auteur décrit le cycle de vie d'une telle perturbation (Figure 11).

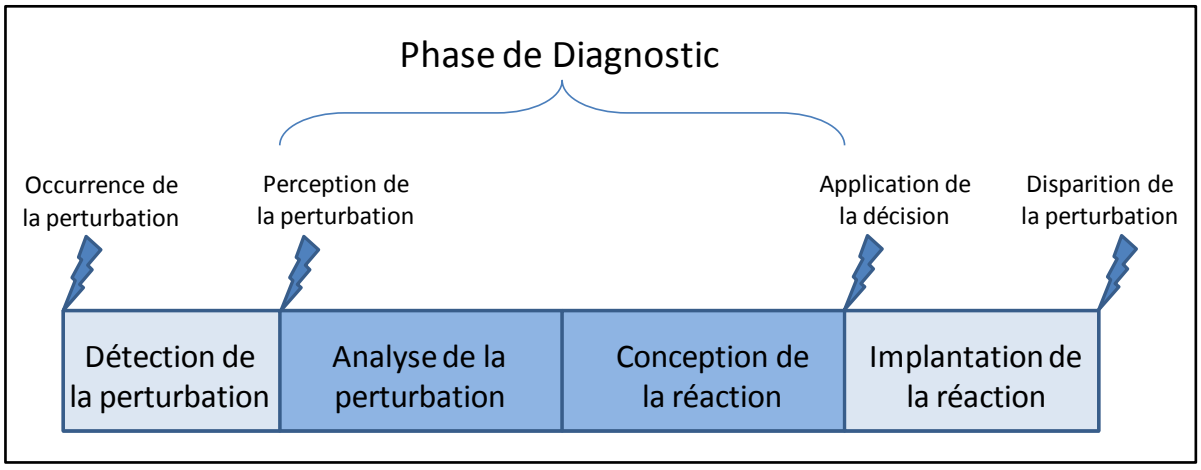

Figure 11 Le cycle de vie d'une perturbation

Lors de l'occurrence d'une perturbation, une première étape consiste à détecter cette perturbation. Il faut détecter le plus rapidement possible cette perturbation et en informer le pilote de la ligne. À partir de la perception d'une perturbation, l'humain doit élaborer un diagnostic. La définition de diagnostic que nous considérons dans ces travaux est celle présentée dans [Hoc, 1990] et [Cegarra et Hoc, 2006] : « une activité de compréhension relative à une décision d'action ». Les auteurs précisent que le diagnostic est une activité de compréhension pour organiser des éléments à l'intérieur d'une structure ayant un sens. Cette organisation est orientée par l'opérateur vers des décisions puis vers des actions. Tout en diagnostiquant, l'opérateur maintient un équilibre entre coûts et bénéfices, tout en essayant d'atteindre une performance acceptable par rapport aux objectifs. Durant la phase

de diagnostic décrite dans [Cauvin, 2005], l'opérateur doit, dans un premier temps, analyser, c'est-à-dire identifier et caractériser, la perturbation. Dans un second temps, la perturbation étant clairement identifiée, il doit déterminer comment réagir. Il s'agit là de concevoir la réaction appropriée qui permettra la meilleure réaction possible du système à la perturbation. Lorsqu'une décision a été prise concernant la réaction appropriée, celle-ci est appliquée. Un certain laps de temps est ensuite généralement nécessaire à la disparition de la perturbation. Le terme choisi par [Cauvin, 2005], *Implantation de la réaction*, est légèrement ambigu du fait de la confusion possible avec l'*Application de la décision*. Le reste de nos travaux ne traitant pas de cette phase, mais plutôt des deux sous-phases précédentes constituant le diagnostic, nous ne différencierons pas les deux termes d'application et d'implantation dans la suite de ce document.

Cette phase de diagnostic étant prédominante dans nos travaux, nous avons cherché à approfondir les mécanismes permettant de la mener à bien. [Hoc, 1996] présente une version révisée du modèle de Rasmussen [Rasmussen, 1983] [Rasmussen, 1986] concernant l'approche couplée homme/machine de la résolution de problèmes dans la phase de diagnostic. La Figure 12 présente les mécanismes possibles de résolution de perturbations par une entité décisionnaire. Les travaux de Hoc et Rasmussen sont principalement orientés vers les comportements de l'humain, mais celui-ci est généralement aidé par la machine dans sa prise de décision.

Chaque cadre de la Figure 12 représente l'action que réalise l'entité dans sa démarche de résolution, à l'exception du cadre *Exécution* qui représente le mode de marche normal du système étudié :

- *Détection de conditions anormales* : l'entité détecte, par un ensemble de capteurs et/ou d'indicateurs de performance, une déviation entre le comportement attendu du système et le comportement réel ;
- *Recherche de données explicites* : l'entité liste l'ensemble des données spécifiant explicitement la perturbation ;
- *Identification de l'état du système* : agrégation des données issues du système étudié permettant l'analyse de son état actuel par l'entité. Ces trois étapes constituent la phase de diagnostic du système ;
- *Prévision de l'évolution du système* : pronostic sur l'impact des stratégies correctives envisagées sur le comportement du système ;
- *Interprétation des conséquences sur les objectifs du système* et *Évaluation en relation avec les contraintes du système* : remise en cause des objectifs globaux du système dans la définition d'une solution ;
- *Définition de la tâche* et *Formulation de la procédure* : procédure d'application de la solution retenue sur le système.

Les flèches de la figure représentent les évolutions possibles entre les étapes de résolution, éventuellement paramétrées par des informations extérieures.

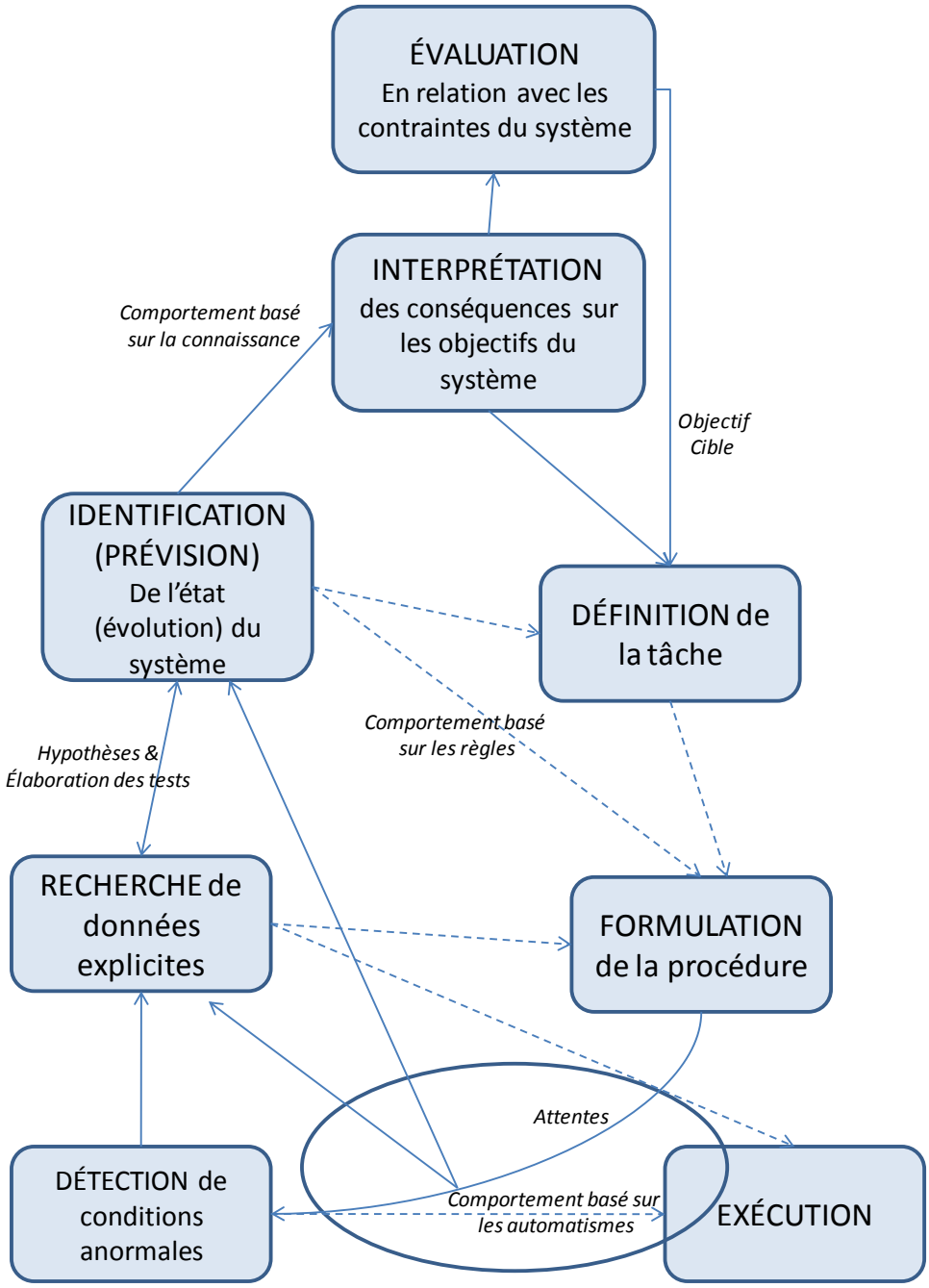

Figure 12 Approche couplée homme/machine de la résolution de problèmes ([Rasmussen, 1983][Rasmussen, 1986] révisé par [Hoc, 1996])

Dans ce modèle, dès que l'entité a détecté une perturbation (passage de l'état *Exécution* à *Détection de conditions anormales*), deux possibilités s'offrent à elle :

- Si elle a exactement rencontré cette situation auparavant, elle peut réagir dans une procédure type réflexe, appelée ici *Comportement basé sur les automatismes*. À la fin de cette procédure, le système est de nouveau dans son état de fonctionnement normal et peut entrer en phase d'*Exécution* ;
- Dans le cas inverse, elle évalue la situation en observant les données disponibles (Analyse de la perturbation en *Recherche de données explicites*).

Ce dernier cas peut être suivi de deux procédures distinctes :

- Si l'opérateur est capable d'adapter et réutiliser une solution déjà appliquée lors d'un problème précédent, alors il n'a qu'à identifier l'état du système, prévoir son évolution future en fonction de la décision paramétrée apportée, puis formuler la procédure de correction de la perturbation ;
- Dans le cas contraire, l'opérateur doit reconstruire une solution originale en prenant en compte les objectifs globaux du système pour pouvoir construire la procédure de correction : il passe alors par les phases d'*Interprétation des conséquences sur les objectifs du système* et d'*Évaluation en relation avec les contraintes du système* où la nouvelle solution sera élaborée.

Les révisions de Hoc modifient le modèle initial de Rasmussen orienté *cognitive engineering* en ajoutant les mécanismes cognitifs d'évaluation de la situation : diagnostiquer et/ou prévoir par une méthode basée sur l'élaboration d'hypothèses suivie de tests puis vérifier l'adéquation du comportement du système aux attentes que le système de décision a par rapport à la décision qu'il a prise. En d'autres termes, ces modifications rajoutent des mécanismes de contrôle de l'application de la procédure définie en examinant si le comportement obtenu répond aux attentes que l'opérateur s'est construit lors de l'élaboration de la procédure.

L'une des richesses de ce modèle est la présence de 3 niveaux de comportement qui peuvent être utilisés indépendamment selon le type de problème rencontré :

- Un comportement basé sur les automatismes qui est adopté lorsqu'un opérateur formé effectue une action de manière automatique à partir de la perception d'un signal spécifique ;
- Un comportement basé sur les règles, où un opérateur confronté à un problème connu adapte et réutilise une solution déjà appliquée dans un problème précédent ;
- Un comportement basé sur la connaissance, où l'opérateur rencontre un problème nouveau et doit créer une nouvelle solution.

Ces deux derniers niveaux sont cognitifs : l'opérateur est considéré comme un solveur de problèmes. Toutefois, devant la complexité de ces tâches, il est souvent nécessaire d'adjoindre à l'opérateur concerné soit d'autres opérateurs soit un système d'aide à la décision pour l'aider dans sa compréhension de la situation et dans l'évaluation des décisions à prendre. Que ce soit l'humain ou la machine qui effectue tout ou partie de la démarche présentée précédemment, aucun des problèmes traités par la simulation en ligne ne peut être classé dans les comportements basés sur les automatismes et sur la connaissance. En effet, pour pouvoir utiliser un automatisme, il faut que la solution au problème soit complètement et parfaitement définie, ce qui n'inclue pas de tests par simulation. D'autre part, le comportement basé sur la connaissance imposerait à l'opérateur de modéliser le problème concerné à partir du moment où la perturbation est détectée puisqu'il se situe en face d'un problème inconnu a priori. Ceci est globalement irréaliste au vu du temps de développement nécessaire à un modèle de simulation et de la réactivité demandée en règle générale à l'opérateur en charge du pilotage de la production.

De ce fait, dans la suite de ce document, toute démarche de simulation, incluant l'opérateur ou non, se situera obligatoirement dans un comportement basé sur les règles.

Devant un problème connu ou très semblable à un problème précédemment rencontré, l'opérateur applique une démarche de simulation établie lui permettant de résoudre ce problème. Si la démarche implique par exemple le choix dans une bibliothèque d'un ou plusieurs sous-modèles à assembler pour construire le modèle approprié, ce comportement est à la limite d'un comportement basé sur la connaissance, au vu des connaissances que l'opérateur doit mobiliser. Toutefois, même si la solution construite est globalement à chaque fois différente de la fois précédente, ce qui pourrait être considérée comme une solution originale, nous considérons que cela s'appuie sur une démarche préalablement définie permettant de ne se raccrocher qu'à un comportement basé sur les règles.

II.1.2 Chronologie de la prise de décision

En se basant sur les réflexions précédentes, et en reprenant la terminologie de la Figure 12, nous pouvons résumer le comportement considéré dans ces travaux, qui part de la détection de perturbation, ou l'apparition d'une décision à prendre, dans un contexte de production non perturbée, jusqu'au retour à l'exécution normale ou à l'application effective de la prise de décision:

1. Apparition d'une décision à prendre, soit dans le cadre normal de production, soit suite à la détection d'une perturbation ;
2. Recherche de données explicites ;
3. Identification de l'état du système de production;
4. Hypothèses et élaboration des tests : choix dans l'espace des solutions possibles d'un ensemble de solutions envisageables. Si cet ensemble ne comporte qu'une solution, la recherche est terminée, la simulation n'intervient pas et l'on passe directement au point 6. Si le nombre de stratégies envisageables est supérieur à 1, alors la simulation peut intervenir. Cette phase est alors suivie du choix du modèle de simulation approprié au test de chacune des stratégies envisagées ;
5. Prévision de l'évolution du système : pour chaque stratégie envisagée, la simulation permet d'estimer l'évolution future du système. En fonction des résultats de la simulation, la stratégie la plus adaptée est choisie ;
6. Formulation de la procédure : définition du nouveau paramétrage du système de production pour l'application exacte de la stratégie choisie ;
7. Après un certain temps, le système revient à son mode d'exécution normal. Pendant ce temps, et encore quelques temps après (l'estimation de ces durées est très liée à la dynamique du système et à la nature de la perturbation), l'entité décisionnaire suit l'évolution réelle du système pour s'assurer de la cohérence de cette évolution avec l'évolution simulée.

Ce comportement nous permet de détailler le modèle de Hoc et Rasmussen dans notre cas particulier. La Figure 13a présente la modélisation du comportement précédent. Toutefois, il est possible de modifier légèrement les points 4 et 5 pour envisager le cas d'un couplage entre un module d'optimisation (type recuit simulé, recherche avec tabous, etc.) et une simulation permettant l'évaluation des hypothèses formulées par ce dernier. Dans de tels couplages, les critères d'arrêt de la simulation sont généralement la minimisation (ou maximisation) d'une fonction-objectif ou un nombre d'itérations maximal pour borner le temps de calcul. Les trois points sont alors modifiés en :

4. Élaboration d'une nouvelle hypothèse par le module d'optimisation ;

5. Prévision de l'évolution du système avec cette hypothèse. Si le critère d'arrêt de l'optimisation n'est pas atteint, on revient au point 4. Dans le cas contraire, on passe au point 6 ;

Ce comportement est celui détaillé sur la Figure 13b.

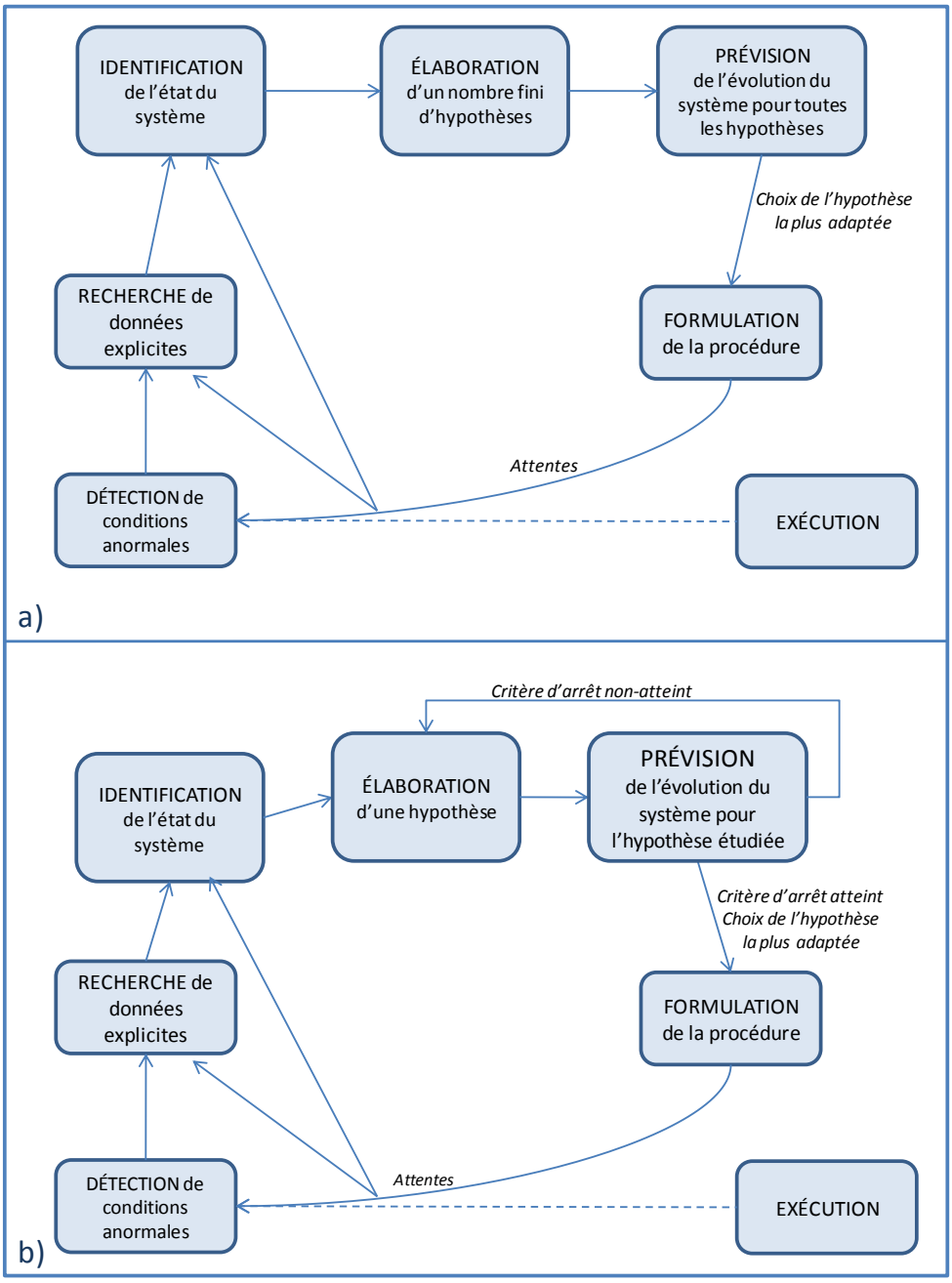

Figure 13 Approche couplée homme/simulation en ligne de la résolution de problèmes
a) Test exhaustif de toutes les hypothèses
b) Couplage optimisation/simulation avec fonction objectif ou nombre maximal d'itérations

Tout comportement non-conforme à celui-ci sort du champ d'application de la simulation en ligne. La question qui se pose désormais est d'estimer la place relative de

l'homme et de la machine dans un tel comportement, où une coopération est généralement nécessaire.

La Figure 14 présente une chronologie détaillée des évènements se déroulant au cours d'une prise de décision basée sur la simulation en ligne avec test exhaustif de toutes les hypothèses (les conclusions suivantes sont également valables pour l'approche avec optimisation). Cette figure intègre le paramètre Dd, correspondant à la durée que se laisse l'entité décisionnaire pour prendre sa décision. À partir de la détection de la perturbation, la première étape est l'analyse de cette perturbation (cf. Figure 11). Lors de cette analyse, la durée estimée de prise de décision doit être estimée. Cette durée correspond à la durée exacte qui sépare la détection de la perturbation et l'application de la réaction. Le paramètre doit être supérieur au temps effectif de prise de décision pour être capable d'appliquer la décision à la date prévue.

Cette durée doit donc être estimée au plus juste, puisqu'elle implique un retard entre la date de la prise effective de la décision et son application (Figure 14). Lorsque l'on focalise sur le déroulement de l'une des simulations (en l'occurrence la simulation testant la stratégie n°1), on s'aperçoit que le paramètre Dd est prépondérant dans la date d'application virtuelle de la décision. Si ce décalage n'entrait pas en compte, les résultats de la simulation seraient faussés puisque le comportement de la simulation et du système réel ne seraient pas équivalents.

Les deux principaux problèmes rencontrés lors de l'utilisation de ce paramètre sont donc :

- Le retard d'application de la décision ;
- La difficulté de détermination de cette durée.

Puisque la durée doit être déterminée au plus juste, il est en effet primordial d'avoir à disposition un outil permettant d'aider à sa détermination. Devant le manque d'outil disponible actuellement, une possibilité à envisager est de systématiquement considérer une durée de prise de décision nulle. Cette solution est principalement à réserver aux décisions où la durée estimée est faible relativement à la dynamique du système de production considéré.

Dans les autres cas, il est nécessaire d'utiliser ce paramètre. Sa détermination est décomposable en la détermination de la durée de plusieurs phases successives (Figure 14):

- Analyse du contexte de la prise de décision ;
- Élaboration des stratégies ;
- Simulations successives ;
- Évaluation des résultats.

De manière générale, la durée de la phase de simulation est assez simple à déterminer, par une somme des durées des simulations. Les autres phases peuvent être assignées alternativement à l'homme ou à la machine selon la décision concernée. Lorsque la machine seule est concernée lors d'une phase, il est généralement possible d'estimer finement sa durée. Lorsque l'humain entre dans la phase, il est nécessaire de réaliser une estimation empirique.

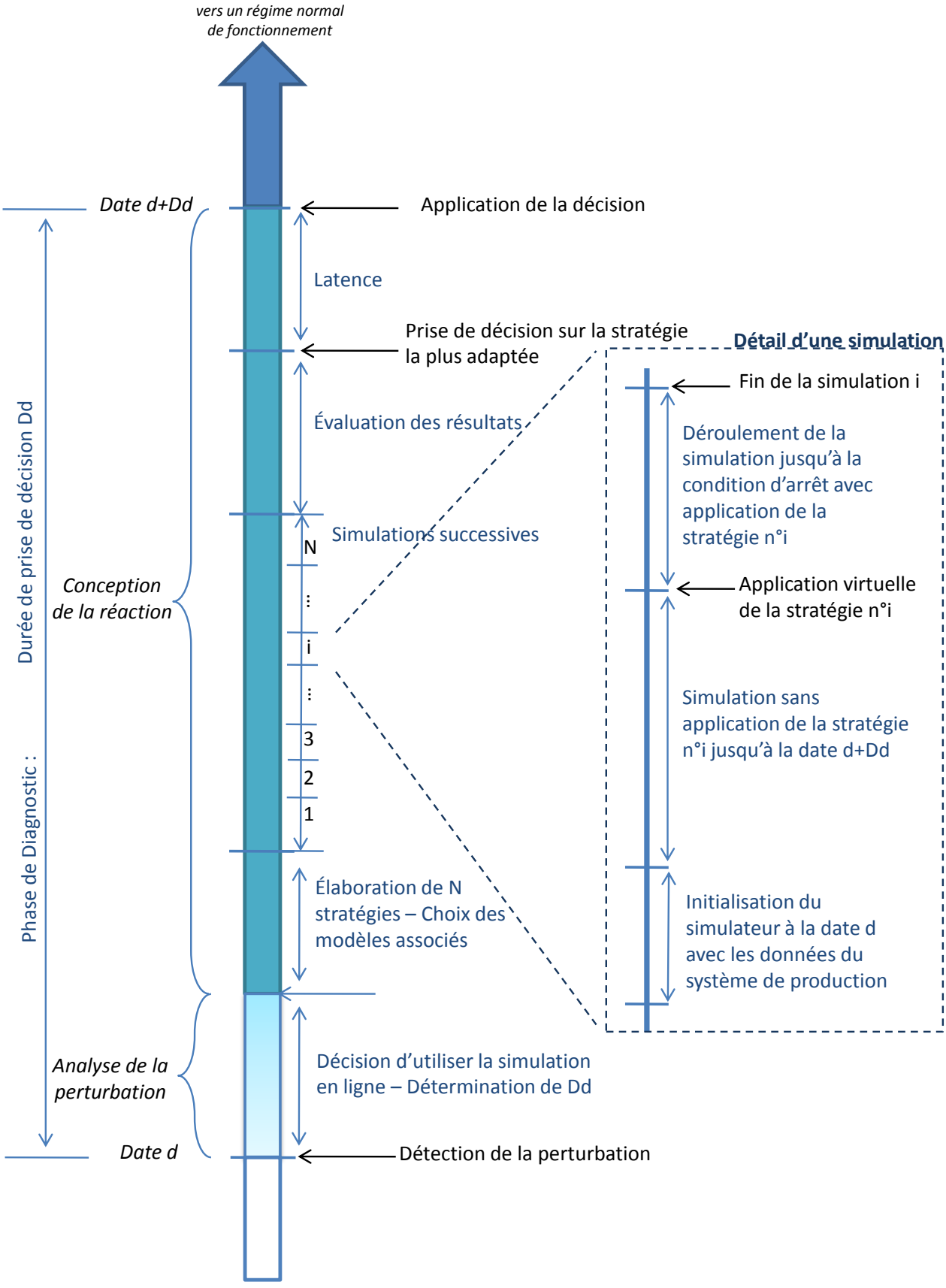

Figure 14 Chronologie de la prise de décision avec test exhaustif des stratégies

II.1.3 Place relative de l'humain et de la simulation dans un comportement basé sur les règles

Nous proposons donc d'étudier la place relative que peuvent occuper l'homme et la machine dans la prise de décision.

Endsley et Kaber ont proposé une extension de la hiérarchie des niveaux d'automatisation applicable aux tâches de commande dynamique-cognitives et psychomotrices, ce qui constitue l'une des extensions de la classification initialement proposée par [Sheridan, 1984]. Ils proposent dix niveaux d'automatisation qui sont : [Kaber et Endsley, 2004]

1. Commande manuelle (Manual control) : l'humain exécute toutes les tâches dont la surveillance du système, la génération des options de performance, la sélection des options à appliquer et l'implantation physique.
2. Aide à l'action (Action support) : à ce niveau, le système assiste l'opérateur dans l'exécution d'actions sélectionnées, bien que des commandes humaines soient toujours requises. Un système de télé-opération avec asservissement de manipulateur basé sur une commande humaine est un exemple commun d'application.
3. Traitement par lots (Batch Processing) : bien que l'humain génère et sélectionne les options à appliquer, elles sont transférées à la machine pour être appliquées automatiquement. L'automatisation est, de fait, principalement au niveau de l'implantation physique des tâches. Beaucoup de systèmes, qui opèrent à ce plutôt bas niveau d'automatisation, existent tels que les systèmes de gestions des lots en industrie ou l'aide à la conduite en voiture.
4. Commande partagée (Shared Control) : à la fois l'humain et la machine génèrent les options de décision possibles. L'humain détient encore un contrôle total sur la sélection des options à implanter, même si cette application est partagée.
5. Aide à la décision (Decision Support) : l'ordinateur génère les options de décision possibles à partir de laquelle l'humain peut choisir, ou celui-ci peut générer de nouvelles options. Une fois que l'humain a sélectionné l'option, la machine l'implante. Ce niveau est représentatif de beaucoup de systèmes d'experts ou de systèmes d'aide à la décision qui proposent des options d'assistance, que l'humain peut choisir de suivre ou d'ignorer.
6. Prise de décision mixte (Blended decision making) : à ce niveau, la machine génère une liste d'options, qu'il sélectionne et implante si l'humain y consent. L'humain peut approuver la décision de la machine, ou bien choisir une autre option de la liste confectionnée par la machine ou par l'humain. La machine appliquera alors cette option.
7. Système rigide (Rigid system) : ce niveau est représentatif de systèmes qui ne présentent qu'un ensemble limité d'options à l'humain. Son rôle est de sélectionner dans cet ensemble. Il ne peut pas ajouter d'options. Ce système est donc plutôt rigide, vu la flexibilité qu'il laisse à l'humain.
8. Prise de décision automatisée (Automated Decision Making) : à ce niveau, la machine sélectionne de manière autonome la meilleure option et la met en œuvre en se basant sur une liste qu'il génère et à laquelle l'opérateur peut rajouter des alternatives.

9. Commande supervisée (Supervisory Control) : à ce niveau, la machine génère les options, sélectionne l'option à mettre en œuvre et l'implémente. L'homme surveille le système et intervient si nécessaire. L'humain surveille principalement le système et intervient si nécessaire. Cette intervention impose à l'humain de faire une sélection d'option différente, ce qui a pour effet de changer de niveau d'automatisation en passant à l'Aide à la décision.
10. Automatisation complète (Full automation) : à ce niveau, la machine effectue toutes les actions seule. L'opérateur est en dehors de la boucle et ne peut intervenir.

Le Tableau 1 résume ces dix points et précisent la répartition exacte des rôles entre l'humain et la machine sur les quatre fonctions génériques intrinsèques des prises de décision : la surveillance, la formulation des options, la sélection des options et leur mise en œuvre, qui sont les bases de cette taxonomie. Cette terminologie est légèrement différente de celle de Hoc et Rasmussen, mais les concepts se rejoignent souvent. Toutefois, les frontières ne sont pas aussi bien définies : la surveillance rejoint le concept de *détection de conditions anormales*, la formulation des options revient à *l'élaboration des différentes stratégies*, la sélection rejoint *l'identification*, et enfin la mise en œuvre est liée à la *formulation de la procédure*. De plus, la phase de *prévision* n'est pas envisagée. Implicitement, celle-ci est plutôt dédiée à l'humain dans les trois premiers niveaux, alors qu'elle est plutôt allouée à la machine dans les niveaux suivants.

Niveau d'automatisation	Rôles ($H \equiv Humain\ ;\ M \equiv Machine$)			
	Surveillance	Formulation des options	Sélection des options	Mise en œuvre
1 - Commande manuelle (Manual control)	H	H	H	H
2 - Aide à l'action (Action support)	H/M	H	H	H/M
3 - Traitement par lots (Batch Processing)	H/M	H	H	M
4 - Commande partagée (Shared Control)	H/M	H/M	H	H/M
5 - Aide à la décision (Decision Support)	H/M	H/M	H	M
6 - Prise de décision mixte (Blended decision making)	H/M	H/M	H/M	M
7 - Système rigide (Rigid system)	H/M	M	H	M
8 - Prise de décision automatisée (Automated Decision Making)	H/M	H/M	M	M
9 - Commande supervisée (Supervisory Control)	H/M	M	M	M
10 - Automatisation complète (Full automation)	M	M	M	M

Tableau 1 Taxonomie des niveaux d'automatisation de [Endsley et Kaber, 1999] pour la performance humain-machine dans des scénarios dynamiques et multitâches

Beaucoup de systèmes d'aide à la décision ont été développés dans le passé (basés par exemple sur des heuristiques d'ordonnancement, etc.) : ces systèmes sont dédiés à la phase de *Prévision* définie au chapitre précédent. En effet, l'opérateur a besoin de connaître l'impact de chacune de ses décisions sur le comportement futur du système pour pouvoir

choisir correctement entre les différentes alternatives qui s'offrent à lui. Sur la plupart des systèmes réels, l'humain seul ne peut évaluer simplement ces conséquences du fait de la complexité du problème. La simulation peut alors être un outil intéressant de test des options en vue de leur sélection. Cette phase de prévision sera donc l'apanage de la simulation dans notre travail.

De la même manière, il est généralement utopique de complètement allouer la phase d'*Identification* à l'opérateur humain. Nous considérons donc que la simulation est l'acteur principal de l'identification et que l'opérateur n'a qu'au maximum un rôle de contrôle sur cette tâche.

Toutefois, en fonction de l'allocation des autres tâches, la difficulté d'implantation de la solution varie. Comme nous l'avons vu Figure 11, la phase de diagnostic traite des tâches *Formulation des actions* et *Sélection des options* de la taxonomie de [Endsley et Kaber, 1999]. Une première approche consiste à considérer une *Automatisation complète*. Ceci est très bien adapté pour les perturbations parfaitement connues, où toutes les procédures de réaction sont très bien établies, et impose donc de pouvoir tout prévoir et modéliser.

Dans la plupart des cas, la détection et l'analyse de la perturbation, comme définis à la Figure 11, ne peuvent pas être totalement automatisées. La surveillance est donc souvent laissée à l'humain assisté de la machine ; d'un autre côté, l'implémentation est pratiquement toujours dévolue à la machine. Une seconde approche serait donc de considérer un *Système rigide*, où la machine détermine toutes les hypothèses et lance les tests automatiquement. Ensuite, elle présente ses conclusions à l'humain pour approbation. Dans [Hoc, 2001], l'auteur pointe quatre types principaux de difficultés existant entre l'humain et la machine. L'une d'entre elles, qu'il appelle la *Complaisance*, révèle que, même si l'opérateur connait, de par son expertise, une meilleure solution que celle proposée par la machine, il adoptera probablement cette dernière. Il est donc recommandé que la machine propose plusieurs solutions plutôt qu'une seule pour encourager le contrôle mutuel.

Pour encourager encore le pilote à être un acteur même de la prise de décision, il faut considérer une architecture type *Aide à la décision* ou type *Prise de décision mixte*. Dans la première, l'humain et/ou la machine établissent les stratégies possibles tandis que la machine lance les tests et affiche les résultats chiffrés, laissant à l'humain seul le soin de prendre la décision. À la différence de cette dernière, la seconde laisse la machine présélectionner les stratégies. Ces deux dernières architectures nous semblent les plus adaptées à l'échelle d'un système de production complet.

II.2 La phase d'identification de l'état et de prévision de l'évolution du système

Le rôle principal que la simulation en ligne a dans une telle architecture de commande se situe au niveau de la phase d'*Identification-Prévision*. Toutes les autres fonctions du système de décision qui relèveraient d'autres phases de la modélisation de [Hoc, 1996] sont assurées par des logiciels autres que de simulation. Nous traiterons donc principalement dans ces travaux des mécanismes intervenant dans cette phase du diagnostic.

Il est clair que la simulation est, par nature, très bien adaptée à la Prévision. La phase d'Identification est, de plus, primordiale dans le cadre de la simulation en ligne du fait de

l'importance de l'état initial sur le résultat des prévisions (chapitre I.3). Ce chapitre indique également qu'il n'existe pas de travaux traitant de cette initialisation sur un système de production complet, ou alors sous un angle très restrictif (I.3.3).

Après avoir quantifié, sur un exemple d'*Aide à la décision,* l'impact de cette initialisation et ainsi pointé l'importance de cette prise en compte, nous proposerons une définition détaillée de la notion d'état d'un simulateur nécessaire à la définition ultérieure d'une méthode fiable et précise. L'étude des implications de cette définition nous permettra ensuite de proposer une solution permettant de résoudre ce problème.

II.2.1 Évaluation de l'importance de la prise en compte de l'état initial sur un exemple

Notre démarche repose sur l'utilisation d'un système de production émulé, inspiré d'un système de production réel. L'émulation consiste classiquement à remplacer le système de production réel en phase de conception par un système simulé pour mettre au point le système de commande [Pannequin, 2007]. Nous utilisons ici ce principe dans le but de tester le système d'aide à la décision (phase d'*Identification-Prévision*) de notre architecture avec des temps d'expérimentation compatibles avec une étude en laboratoire.

Le système étudié servira à la validation de l'approche globale présentée au Chapitre IV, et sera alors présenté plus en détail. Nous commençons tout de même ce chapitre par une présentation rapide du fonctionnement de celui-ci.

II.2.1.1 Énoncé

Le principe de cette expérimentation est de chiffrer l'apport de la prise en compte de l'état de l'atelier dans l'état initial des simulations en ligne. Pour ce faire, nous avons choisi de travailler sur un job-shop à six machines. Chacune de ces machines a un stock amont de capacité finie. Les transferts entre machines sont automatisés et réalisés par des transporteurs unidirectionnels parcourant une unité de distance par unité de temps, en nombre fini (42), stockés dans un magasin lors des phases d'inactivité. Chaque transporteur déplace un même produit tout au long de sa gamme de fabrication, et peut réaliser successivement plusieurs produits d'un même ordre de fabrication sans retour au magasin. Une schématisation de cette installation est présentée Figure 15. La structure du système de transport est construite autour d'une boucle centrale. Les transporteurs se déplacent sur la boucle centrale d'entrée de poste en entrée de poste cherchant à rentrer sur l'un d'eux.

II.2.1.2 Problématique

Quand le pilote du système lance un nouvel ordre de fabrication, plusieurs paramètres de l'ordre doivent être renseignés : la gamme de fabrication associée, le nombre de produits à fabriquer et le nombre de transporteurs alloués à cet ordre. Si on a 20 produits à fabriquer, on peut allouer 5 transporteurs qui réaliseront chacun successivement 4 produits, mais on peut aussi allouer 10 transporteurs qui réaliseront chacun successivement 2 produits, etc. Pour le réglage de ce dernier paramètre, aucune règle ne permet de donner pleine satisfaction. En effet, si le pilote choisit un nombre trop faible, la parallélisation des opérations ne peut pas s'effectuer et le système peut être en sous-charge, alors que s'il choisit un grand nombre dans le but de paralléliser au maximum les opérations, cela causera

des ralentissements dus à l'engorgement des zones de transferts. De plus, cela diminue le nombre de transporteurs disponibles pour les futurs ordres de fabrication.

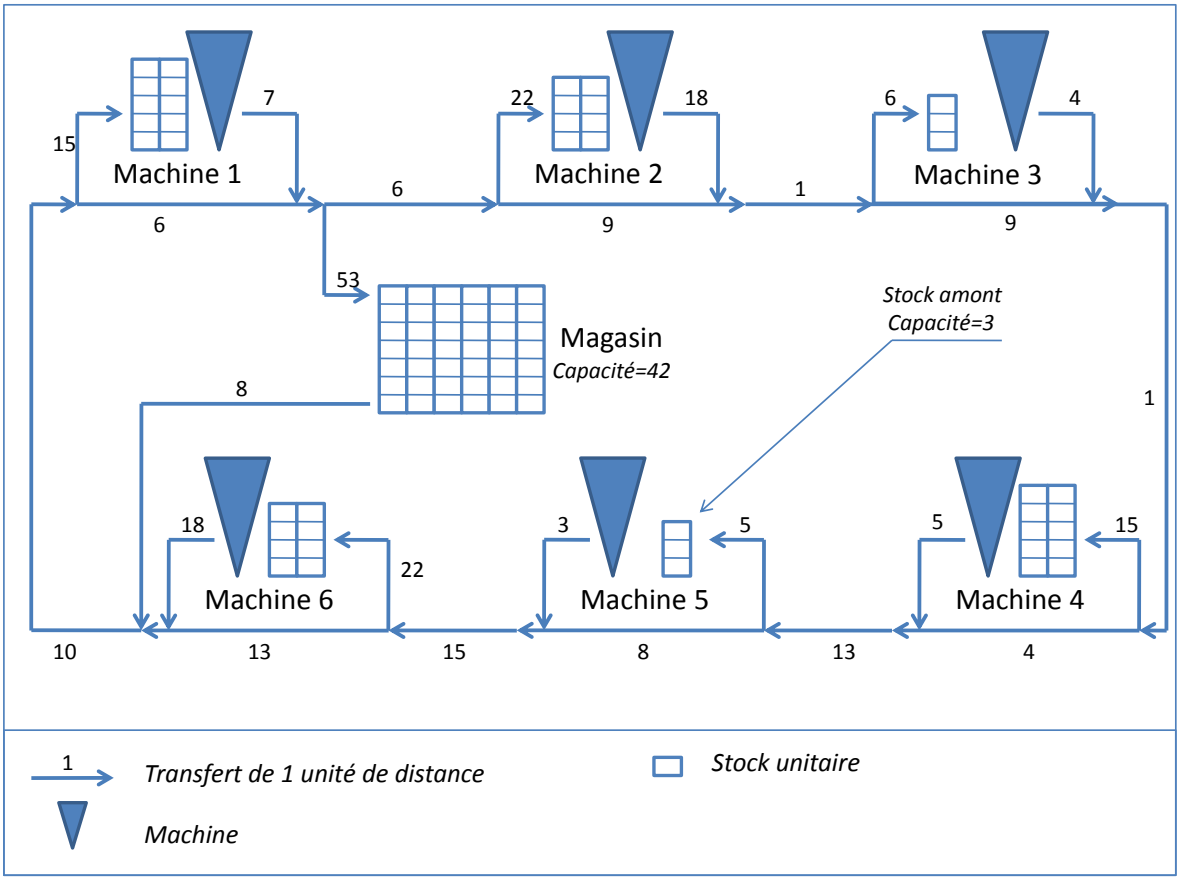

Figure 15 Job-shop à 6 machines et à transferts automatisés

Cette décision a une grande influence sur la performance du système, et s'effectue sur un terme pouvant être très court relativement à la dynamique du système. De plus, elle s'appuie très fortement sur l'état actuel du système. Par exemple, si les ordres de fabrication précédents engorgent une des machines au moment du passage des produits du nouvel ordre sur cette machine, il est préférable de diminuer le nombre de transporteurs pour ne pas ajouter à l'embouteillage. Le problème est que l'état du système au moment concerné peut être totalement différent de l'état actuel, ce qui rend presque impossible l'analyse du pilote.

II.2.1.3 Protocole d'essai

On considère un ordre de fabrication noté OF1, utilisant 20 palettes tournant en permanence sur la ligne et réalisant une gamme visitant successivement les postes 1, 2, 3, 4, 5 et 6. Cet OF permet de placer la ligne dans un état de production non vide.

On considère ensuite un deuxième ordre, noté OF2, devant réaliser 40 produits avec N_t transporteurs. Cet OF2 utilise la gamme visitant successivement les postes 3, 6, 4, 2, 1 et 5.

Le modèle ne présente aucun paramètre aléatoire lors de son lancement, ce qui fait que lancer deux fois la même expérience donnera exactement le même résultat. De ce fait, nous allons pouvoir comparer plusieurs scénarios de lancement (correspondant à plusieurs paramétrages de l'OF2) sans multiplier les simulations pour établir un intervalle de confiance.

La production débute à t=0 avec l'OF1. Puis, à une date t=D, on lance l'OF2. C'est donc à la date D que doit se décider le nombre N_t de transporteurs à affecter à l'OF2. On considère ce nombre compris entre 1 et 40. Le nombre de stratégies envisagées est donc de quarante. D'un côté, il est bien évidemment inutile d'allouer à l'ordre de fabrication plus de transporteurs que de produits à réaliser. D'un autre côté, nous ne nous limitons pas aux seuls sous-multiples de 40, qui permettraient d'assigner à tous les transporteurs un nombre de produits à effectuer identique. En effet, il est possible que l'optimum se situe avec une répartition différente. Pour ne pas multiplier les essais, nous considérons qu'une seule stratégie de répartition existe, ayant pour objectif de la lisser au maximum.

La règle de répartition des produits sur les transporteurs est la suivante. Soit N_t le nombre de transporteurs alloués à l'ordre de fabrication, et N_p le nombre de produits à réaliser (dans nos exemples, $N_p = 40$). On note E la fonction partie entière. On considère alors N_a transporteurs qui réalisent $a = E(\frac{N_p}{N_t}) + 1$ produits et N_b transporteurs qui réalisent $b = E(\frac{N_p}{N_t})$ produits. On a alors :

$$N_a = 40 - b \times N_t$$

$$N_b = a \times N_t - 40$$

Ainsi, si l'on considère $N_p = 40$ et N_t =22, alors $a = 2,\ b = 1,\ N_a = 18\ et\ N_b = 4$. Si N_t =9 alors $a = 5, b = 4,\ N_a = 1\ et\ N_b = 1$.

II.2.1.4 Calcul du gain de performance

Pour déterminer N_t, nous envisageons deux situations :

- Situation A : on considère que le simulateur débute à un état vide. On ne simule donc que l'OF2. On fait varier le nombre de transporteurs. On note N_0 le nombre de transporteurs qui aurait été retenu si on ne pouvait réaliser que des simulations démarrant à un état initial vide, sans tenir compte de l'OF1 en cours. On réalise ensuite une simulation prenant en compte les deux ordres, le deuxième utilisant N_0 transporteurs et on note M_0 le MAKESPAN obtenu pour l'OF2.
- Situation B : On prend en compte l'état initial au début de la simulation. On réalise donc des simulations avec les deux ordres. De la même façon, on fait varier N_t pour déterminer sa valeur optimale. On note N_1 la valeur optimale du nombre de transporteurs trouvée et M_1 le MAKESPAN correspondant.

On définit le gain en % de la situation B par rapport à la situation A par :

$$G = 100\ \frac{M_0 - M_1}{M_0}$$

C'est le gain obtenu en ayant une meilleure prise de décision du fait de la simulation réalisée.

II.2.1.5 Résultats

Nous avons réalisé cet essai à 30 dates de lancement différentes pour éviter de tomber sur un cas particulier lors des simulations (dates uniformément réparties de 0 à 3000 ut). La moyenne des gains obtenus s'élève à environ 21% avec un écart-type de 3%. Ce résultat nous semble probant de l'efficacité de la prise en compte de la dynamique actuelle de l'atelier dans la simulation, et notamment au niveau de son état initial. En effet, avoir un gain si important, en comparaison d'une technique déjà efficace qu'est la simulation de flux, n'est pas négligeable.

Pour aller plus loin, nous avons cherché à évaluer l'évolution de ce gain en fonction de la charge de l'atelier au moment du lancement de l'OF2. Tel que l'expérimentation a été construite, nous avons défini la charge comme étant *le pourcentage de transporteurs indisponibles pour le prochain ordre de fabrication*. Ainsi, la charge de l'exemple cité précédemment était aux alentours de 48% (20 transporteurs sur les 42). Nous avons donc mené le même raisonnement pour des charges successives d'environ 12, 24, 36, 48, 61 et 83% à une même date de lancement (t=1000 ut). S'il est évident que le gain sera nul à 0% (la simulation en ligne est alors une simulation classique), nous ne sommes pas non plus montés dans de trop grandes charges car les résultats n'auraient plus été significatifs en raison du trop faible nombre de transporteurs disponibles. La Figure 16 représente l'évolution du gain obtenu en fonction de la charge initiale de l'atelier.

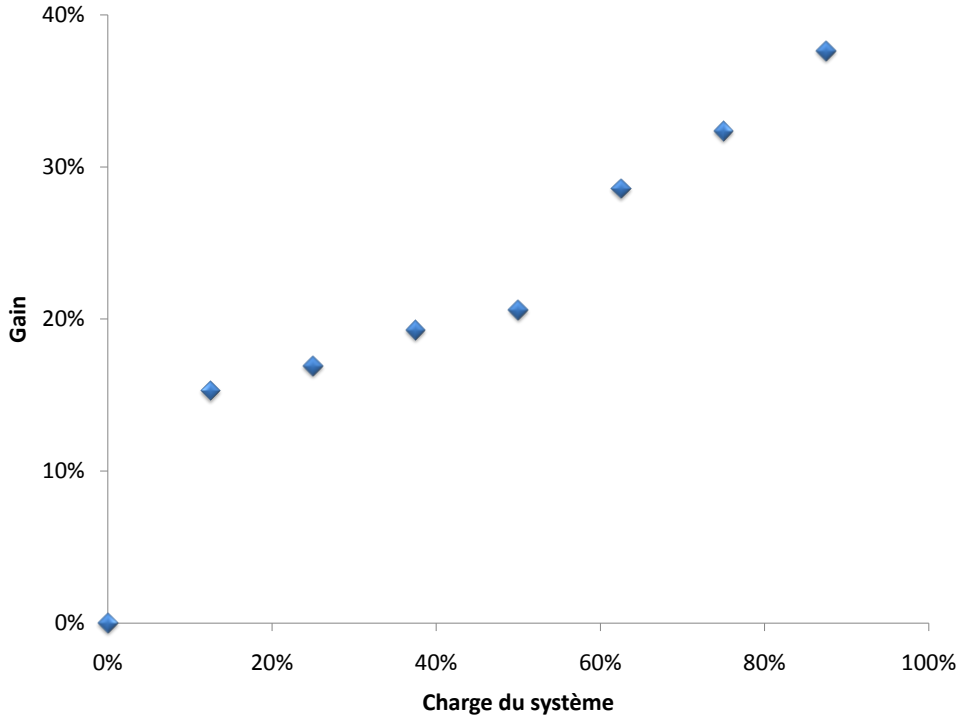

Figure 16 Évolution du gain en fonction de la charge de l'atelier

Ce résultat tend à indiquer que l'impact de l'état initial sur la précision des simulations croît à mesure que la charge de l'atelier augmente. En d'autres termes, cela conforte l'idée

selon laquelle plus l'état de l'atelier diffère de l'état libre et vide habituellement employé dans les simulations classiques, plus la prise en compte de cet état sera importante dans une simulation en ligne.

Les deux résultats exprimés ici nous semblent indiquer l'importance capitale de la prise en compte de l'état de l'atelier dans l'initialisation des simulations : son influence varie entre 20 et 40% sur le résultat final des simulations en ligne selon la charge de l'atelier.

II.2.2 L'initialisation

Les parties précédentes ont montré à quel point la prise en compte de l'état de l'atelier au moment de la prise de décision dans les simulations en ligne bénéficie à la précision des prédictions et donc aux décisions. Le but de cette partie est de définir les notions d'état du système et d'état du simulateur, notions qui doivent se rejoindre pour permettre cette prise en compte.

II.2.2.1 L'état du système de production

La définition suivante, classiquement adoptée, peut être trouvée dans [Banks *et al.*, 1996] : l'état d'un système consiste en une collection de variables contenant toutes les informations nécessaires pour décrire le système à tout instant.

Cette définition reste vague. En effet, la notion de description du système est floue, notamment dû au fait que le niveau de détail n'est pas explicité. À celle-ci, nous préférerons alors utiliser la définition suivante, qui peut être trouvée dans [Pooch et Wall, 1993] :

> ***État du système*** *: Collection minimale d'informations avec lesquelles le comportement futur du système peut être déterminé de façon unique en l'absence de hasard.*

L'utilisateur doit donc récolter toutes les informations nécessaires à la prédiction, ce qui peut aller des informations de l'ERP à l'état d'un capteur. Le problème est alors de définir cet ensemble nécessaire et suffisant d'informations. En effet, il est nécessaire de connaître parfaitement la commande du système complet pour pouvoir envisager de le construire. Et même si l'utilisateur arrive à le définir, il lui faudra encore faire le lien entre ces données et les données constituant l'état du simulateur.

II.2.2.2 État d'un simulateur

Les modèles de simulation, quel que soit le progiciel utilisé, utilisent à peu près les mêmes éléments de base (à la terminologie près) :

- Entités : ce sont les objets circulant du modèle – chaque entité a une identification unique ;
- Attributs : chaque entité possède un nombre fini identique d'attributs. Chaque attribut a un identifiant unique, et possède une valeur (entier, réel, chaine de caractères, etc.) ;
- Files d'attente : elles permettent de stocker les entités dans un certain ordre – elles possèdent un identifiant unique, une règle de gestion (FIFO, LIFO, etc.) et une liste ordonnée des identifiants des entités contenues dans cette file ;

- Transporteurs : permettent de modéliser des transports d'entité par unités indépendantes – ils possèdent un identifiant unique, un statut (occupé, libre, etc.), une position sur le réseau, une destination et éventuellement un identifiant d'entité contrôlante s'il est actuellement piloté ;
- Convoyeurs : permettent de modéliser un transport d'entités à vitesse constante sur un chemin prédéfini – ils possèdent un identifiant unique, un statut (arrêté, démarré, etc.), une longueur et une liste de toutes les entités qu'ils transportent avec la distance restant à parcourir associée ;
- Variables : sont des données globales au modèle. A l'inverse des attributs, elles possèdent à un instant donné une valeur unique. Chaque variable a un identifiant unique, et possède une valeur (entier, réel, chaine de caractères, etc.) ;
- Ressources : sont des objets pouvant être réservées par les entités. Elles ne sont dépendantes des entités que par le fait que ce sont elles qui les réservent ou les libèrent – sans que ce soit obligatoirement la même qui fasse les deux.

La Figure 17 résume tout ceci et présente une modélisation des liens entre ces éléments sous la forme d'un diagramme de classe UML. Il est important de noter que ce méta-modèle ne prend pas en compte la partie « statistiques » inhérente à tout modèle de simulation, partie qui n'a pas encore été complètement étudiée dans ces travaux. En effet, il semble que les statistiques soient reconstructibles à partir de leurs valeurs initiales et de leur valeur en fin de simulation. Prenons l'exemple d'un taux d'utilisation de ressource. S'il vaut T_1 à la date D_1 d'initialisation de la simulation, et qu'à la fin de cette simulation, on relève un taux T_2 sur une durée de simulation D_2. La durée globale de fonctionnement de la ressource est bien $T_1D_1 + T_2D_2$ et le taux global peut s'écrire $\frac{T_1D_1+T_2D_2}{D_1+D_2}$. On pourrait faire un raisonnement analogue pour la mesure d'un MFT (*Mean Flow Time*). N'ayant pas étudié cette possibilité à l'ensemble des statistiques envisageables dans une simulation, nous ne développerons pas plus cette partie qui n'entrera donc pas en ligne de compte dans le reste de nos travaux.

Pour assigner l'état d'un simulateur à une date donnée, il faut donc forcer (assigner à une valeur donnée) l'état de l'instance de la classe Modèle correspondante. Pour cela, il faudra forcer l'état des instances de toutes les classes qui lui sont reliées : pour chaque modèle, l'état sera donc décomposé en six parties distinctes.

La Figure 18 présente l'état obtenu en arrêtant un modèle comportant 3 variables, 3 ressources, 2 files d'attente, 2 convoyeurs et 3 transporteurs à une date aléatoire.

Si ceci était le modèle de simulation en ligne, il est nécessaire de pouvoir complètement reconstruire ces données à partir de celles de l'atelier. Nous proposons de formater ces données en autant de fichiers de données intermédiaires qui seront lus par le simulateur au moment de son utilisation. Ce choix est guidé par le fait que tous les logiciels actuels savent lire et écrire de façon relativement identique dans des fichiers texte. Chaque fichier contiendra les informations relatives à une classe définie ci-dessus.

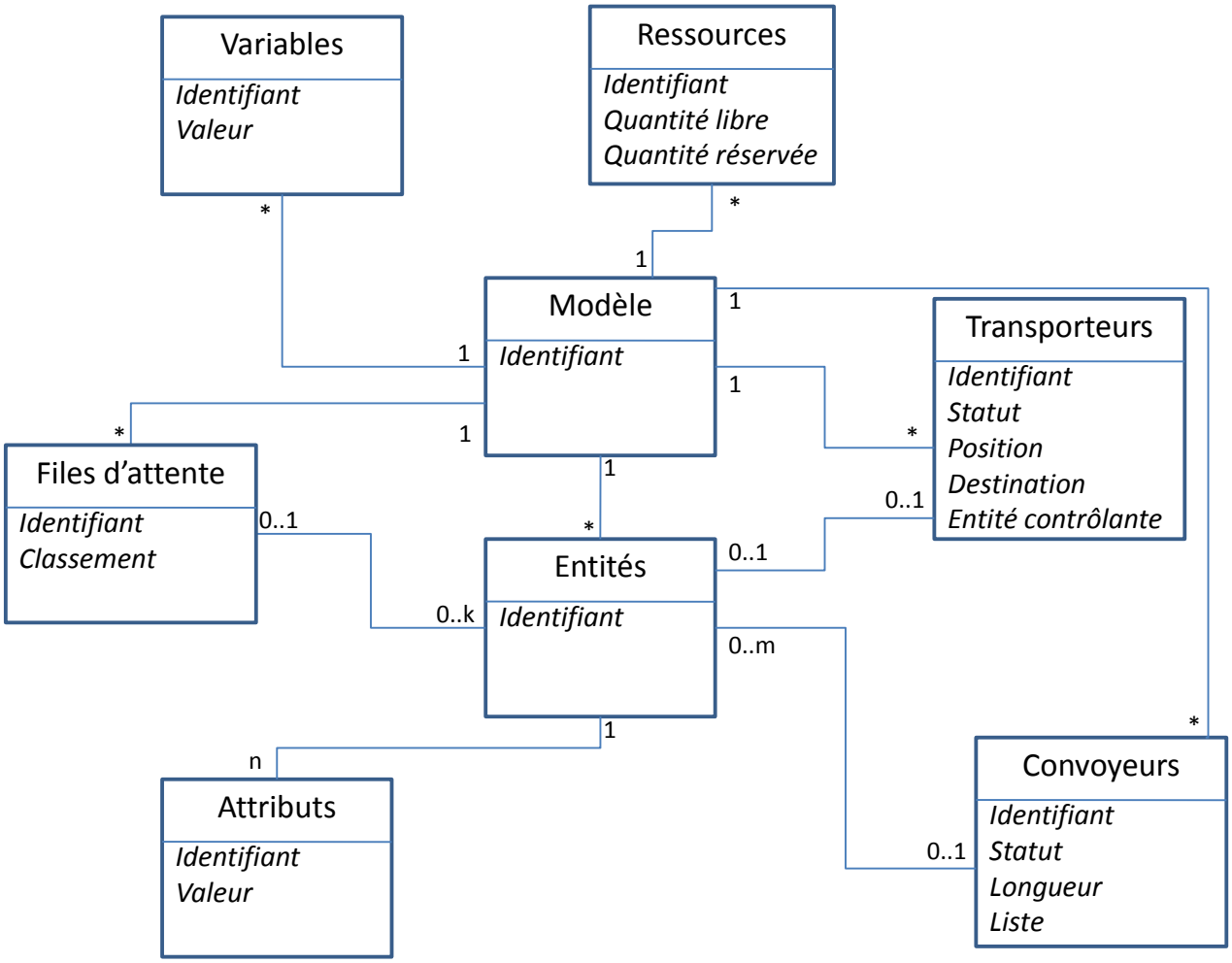

Figure 17 Méta-modélisation UML d'un modèle de simulation

Lorsque le simulateur est arrêté pour nous permettre d'enregistrer son état (Figure 18), les entités ne peuvent être que : dans une file d'attente, assignées à un transporteur, sur un convoyeur, dans une machine (caractérisé par la réservation d'une ressource) ou dans une partie logique de la programmation. Cette dernière catégorie est représentée par le fichier Entités. Des points d'interrogation ont été mis dans ce fichier, car il est en général impossible d'obtenir simplement toutes les informations nécessaires au traitement de ces entités. Par exemple, sur Arena, une entité de ce type est dans un bloc Delay, et va bientôt sortir. On peut savoir quand elle va sortir, mais il ne nous a pas été possible de développer une méthode générique permettant de connaitre, à coups sûr, sa prochaine destination. Au contraire, des progiciels tels Quest ou Witness ont une structure qui ne laisse pas cette ambiguïté : les « parts » ou « articles », qui jouent le rôle d'entités, sont toujours dans des endroits identifiables du modèle (stocks, machines, etc.).

De plus, ces entités n'ont en général pas leur équivalent dans l'atelier, ce qui rend impossible leur connexion avec le système réel. De ce fait, il est impossible de prendre en compte de telles structures de modèles et de les initialiser avec les données de l'atelier.

Variables	
Identifiant	Valeur
1	5
2	'FIFO'
3	2.5

Ressources	
Identifiant	Quantité réservée
1	2
2	0
3	8

Entités	
Identifiant	Action
5	???
8	???
9	???

Files d'attente						
Identifiant	Rang	Entité	Attribut 1	Attribut 2	…	Attribut n
1	1	7	0	11	…	8
1	2	3	14	27	…	1.4
2	1	10	12.5	9	…	0

Convoyeurs							
Identifiant	Statut	Longueur	Entité	Distance	Attribut 1	…	Attribut n
1	'arrêté'	10	11	1	0	…	8
1	'arrêté'	10	2	9	174.25	…	99
2	'démarré'	74	6	131	11	…	10

Transporteurs							
Identifiant	Statut	Position	Destination	Entité	Attribut 1	…	Attribut n
1	'busy'	13	14	4	0	…	8
2	'idle'	71	0	0	0	…	0
3	'busy'	17	18	1	12.6	…	13

Figure 18 L'état d'un simulateur

II.2.2.3 Les informations dans l'atelier

Reprenons l'exemple décrit Figure 18 et remontons jusqu'à l'atelier. Si ces fichiers sont l'état initial de la simulation, alors les informations qu'ils contiennent représentent l'état de l'atelier réel. Ainsi, il y a un produit sur le convoyeur n°2, à une distance de 31 unités de

distance (disons pour simplifier des mètres) de la fin du convoyeur. Puisque ce chiffre est forcément un entier sur beaucoup de progiciels, que se passe-t-il si, après avoir mesuré sur le convoyeur, le produit est en fait à 30.5 mètres de la fin du convoyeur ? Il faut alors arrondir. Mais en faisant cela, l'information est dégradée, et le résultat peut être faussé (ce qui peut avoir une grande influence si le temps passé sur le convoyeur détermine le temps de cuisson d'un produit par exemple). La solution est alors de passer à l'unité inférieure, par exemple au millimètre, en risquant de faire largement diminuer la rapidité de la simulation du fait du grand nombre d'éléments à animer. Les spécificités de chacun des progiciels de simulation imposent également certaines contraintes de modélisation. Par exemple, sous SIMAN Arena, la longueur du convoyeur est forcément un multiple de la taille des produits qu'il est censé transporter.

C'est ce que [Roy, 1989] décrit comme étant l'une des quatre sources d'incertitudes et imprécisions dans les modèles de décision : « les données ne sont pas le résultat d'une mesure exacte ». Cette analyse est tout-à-fait transposable à la problématique de la construction des modèles de simulation.

Les trois dernières sources évoquées traitent des éventuelles imprécisions du modèle par rapport à la réalité, notamment au niveau de la subjectivité introduite dans la construction de tout modèle par son auteur et de la différence inévitable entre le résultat d'une simulation et le déroulement réel de la production due au fait que le comportement futur du système de production n'est pas obligatoirement le prolongement de son comportement actuel (ou passé). Dans ce cadre, l'application de données venant du système réel, même exactes, à une modélisation imprécise ne peuvent pas donner les résultats escomptés.

II.2.2.4 Initialiser le modèle

Considérons pour ce paragraphe que l'utilisateur a en sa possession toutes les informations nécessaires à l'initialisation de son modèle. Deux stratégies s'offrent alors à lui : une initialisation *directe* ou en *warm-up*.

II.2.2.4.1 Initialisation en warm-up

Tous les progiciels de simulation utilisent déjà cette notion à des fins statistiques. En effet, la simulation d'un atelier par exemple commence classiquement avec un atelier vide. Or, si l'on veut connaître la cadence moyenne de l'atelier au cours de son fonctionnement, toute la phase initiale de remplissage de l'atelier va faire faussement baisser la moyenne. De ce fait, les progiciels proposent tous la possibilité d'éliminer le suivi statistique durant une phase transitoire de fonctionnement.

Le principe ici est légèrement différent. À partir de la collecte des informations sur le système réel, il est nécessaire de construire une trace que doit suivre le simulateur depuis son état initial. Cet état est toujours le même à chaque démarrage de la simulation, mais peut être choisi au mieux pour faciliter l'initialisation sans influer sur la justesse de la méthode. Lorsque le simulateur a fini de dérouler la trace déterminée, l'état du simulateur et la date courante doivent correspondre au mieux à l'état du système réel au moment de la collecte.

Cette méthode a deux gros désavantages : la construction de la trace peut s'avérer très complexe et les durées de construction puis de déroulement de la trace ralentissent son utilisation.

II.2.2.4.2 Initialisation directe

L'autre solution est d'utiliser une initialisation directe. Pour cela, il faut injecter à la date souhaitée tous les éléments dans la simulation et artificiellement recréer un calendrier pour que la simulation continue. En effet, si le calendrier est vide, la simulation s'arrête, n'ayant rien à faire dans la phase PMH (voir chapitre I.2.1.3 pour plus de précisions). Certains cas sont assez faciles, comme réinsérer des entités dans une file d'attente. D'autres sont plus compliqués, comme reprendre un temps d'attente en cours. Pour ces derniers, de nouvelles architectures doivent souvent être construites pour permettre de redémarrer correctement. Outre ceci, le principal écueil reste souvent encore l'impossibilité d'insérer simplement les entités dans le modèle (par exemple au milieu d'un convoyeur) sur beaucoup de progiciels du commerce.

Malgré tous ces problèmes, cette méthode est vivement à conseiller. En effet, son principal avantage est de se faire en une durée négligeable, alors que l'initialisation en warm-up oblige un temps de computation puis un temps de simulation préalables. Or, le principe de la simulation en ligne est de délivrer ses résultats en un délai très restreint, ce qui rend l'utilisation du warm-up rédhibitoire.

II.3 Conclusion

Dans sa première partie, ce chapitre a permis de positionner les rôles de l'humain et de la simulation en ligne dans le pilotage d'un système de production. Nous avons tout d'abord étudié les différentes phases se succédant lors de la prise de décision pour en déduire les possibilités de répartition de tâches entre l'humain et la machine dans la phase d'Identification/Prévision. Les deux approches qui nous ont semblées les mieux adaptées à l'utilisation de la simulation en ligne dans le pilotage des systèmes de production sont dénommées *Aide à la décision* et *Prise de décision mixte*.

La suite du chapitre s'est ensuite intéressée à l'initialisation de la simulation en ligne sur l'état du système réel. La première partie de nos travaux a permis d'étudier l'impact de cette initialisation pour décider de la nécessité de son utilisation. Les résultats que nous avons obtenus tendent à montrer un impact important et non-négligeable, ce que les travaux de la littérature sur le sujet laissaient entendre. Nous avons ensuite mis en relation les notions d'état du simulateur et d'état de l'atelier. À la comparaison de ces deux états, il nous semble clair qu'aucune identification directe ne saurait être réalisée dans le cas général.

Le chapitre prochain a pour objectif de présenter les différentes approches que nous avons étudiées, et notamment celle utilisant un observateur par simulation que nous avons particulièrement développée.

Chapitre III.

Utilisation de la simulation comme observateur d'un système de production

Nous avons montré dans le Chapitre II l'importance d'initialiser notre modèle de simulation en ligne pour que son état en début de fonctionnement corresponde le plus exactement possible à la situation du système réel. Dans ce chapitre, nous cherchons à voir comment connaître cette situation du système réel.

Dans un premier temps, nous envisagerons deux possibilités pour connaître la situation actuelle de notre unité de production. La première consiste à récupérer directement à partir du système réel les informations nécessaires à l'initialisation de notre simulateur. Nous montrerons que ces informations pourront être en partie fournies par le MES. Se pose alors un problème important : l'état complet du système n'est que partiellement connu par le MES, à un instant donné. En effet, du fait de la position et de la nature des capteurs et détecteurs présents sur le système de production, l'état pouvant être donné par le MES comporte des incertitudes à la fois spatiales et temporelles. C'est pourquoi nous envisagerons une deuxième solution consistant à utiliser un simulateur temps-réel reproduisant le comportement du système réel. Ainsi, l'état de ce simulateur à un instant *t* correspond à l'état supposé du système de production à ce même instant. Nous verrons que la nature stochastique de la plupart des ateliers de production limite beaucoup cette approche : très rapidement, l'état du simulateur temps-réel diffère de la situation réelle.

Nous montrerons ensuite comment, pour résoudre ce problème, nous avons proposé d'utiliser conjointement les deux possibilités en proposant le concept d'observation par simulation.

III.1 Initialisation utilisant directement l'état du système de production

III.1.1 Notion d'état du système

La première remarque que nous pouvons faire est que l'état d'un système n'a de sens que par rapport à l'utilisation que l'on veut en faire. Ainsi, il existe un état du système par rapport à sa commande. C'est, à un instant t, l'ensemble des informations directement « prélevées » sur le système réel, ou calculées à partir de ces informations prélevées, nécessaires pour générer sa commande.

Il existe de même un état du système par rapport à l'initialisation d'une simulation en ligne. C'est l'ensemble des informations qui sont nécessaires pour initialiser notre simulateur. Ces deux états peuvent différer. Prenons l'exemple très simple d'un vérin pneumatique. Imaginons que le cahier des charges du système de commande n'impose pas la maitrise de la trajectoire du vérin au cours du temps mais seulement les commandes de sortir et de rentrer complètement la tige de ce vérin. Le système de commande n'a donc besoin de connaitre que trois états concernant la tige de ce vérin :

- La tige est en position rentrée ;
- La tige est en position sortie ;
- La tige n'est ni rentrée, ni sortie.

Le concepteur du système de commande se contentera donc de placer deux détecteurs binaires délivrant une quantité d'information au sens de Shannon égale à deux bits. En général, il suffira de placer un détecteur nous informant que la tige est sortie et un détecteur nous informant que la tige est rentrée. L'absence de détection des deux détecteurs correspond alors à l'information « la tige n'est ni rentrée ni sortie ».

En réalité, la position de la tige du vérin correspond à une information continue $x(t)$ indiquant la position exacte du vérin au cours du temps. Si cette information n'est pas utile pour la commande, elle l'est pour l'initialisation de notre simulation. En effet, en cours de déplacement, la position actuelle de la tige et son sens de déplacement peut permettre de prévoir la date d'arrivée en butée, d'un côté ou de l'autre. Nous remarquons donc sur cet exemple que la quantité d'informations nécessaires à l'initialisation de la simulation en ligne peut être plus importante que celle nécessaire à la commande. La question importante qui se pose alors est : Peut-on utiliser l'état par rapport à la commande pour générer l'état par rapport à la simulation en ligne ?

Avant d'aborder cette question, précisons quelle est la nature des informations que nous pourrons trouver dans l'état par rapport à la commande du système. Nous distinguerons deux types d'informations :

- Des informations directement mesurées sur le système physique par l'intermédiaire de capteurs ou de détecteurs ;
- Des informations connues par le système de commande, soit parce que ce dernier les à calculées, soit parce qu'elles correspondent à des consignes.

La question qui se pose ensuite est la disponibilité des données nécessaires à la connaissance de l'état du système de production. Ces données doivent être à jour et refléter

complètement l'état du système de production tel que nous l'avons défini au chapitre précédent. Nous proposons de décomposer ces données en deux catégories.

Tout d'abord, les *données de production*, nécessaires au pilotage du système de production, en général contenues dans la base de données du MES et/ou de l'ERP, sont faciles à obtenir. Il s'agira par exemple des ordres de fabrication en cours, du paramétrage des postes de travail ou des gammes associées aux références de produits, constituant une grande partie du système d'information inhérente au système de production. Il faut permettre au simulateur de travailler avec les données issues de cette base de données, possibilité que la plupart des progiciels de simulation proposent par défaut. Les deux acteurs que sont la simulation et le système de production travaillant alors sur les mêmes données, la question de l'homogénéité de ces dernières ne se pose pas. Reste alors à les intégrer dans le simulateur, ce qui ne pose en principe pas de difficulté particulière.

Ces données n'évoluent que relativement peu au cours du temps, sur des évènements très bien maitrisés et ne présentent donc pas de problèmes particuliers de mise à jour. Au contraire, certaines données évoluent souvent, et leur disponibilité n'est pas aussi évidente que pour celles du système d'information. Ces données sont à récupérer directement sur le système de production. Il s'agit de toutes les *données relatives à l'état du système*. On pourra par exemple citer la position des produits dans l'atelier, l'état d'une ressource opérateur ou encore le contenu d'un stock. Ce sont ces données qui constituent le cœur de notre propos dans le reste de ce chapitre.

III.1.2 Utilisation des informations directement issues du système réel.

La première solution que nous avons envisagée est d'initialiser directement le simulateur avec les données issues du système réel, au travers de la commande, du MES ou des composantes supérieures de la décomposition CIM, telles que l'ERP ou l'APS (Figure 19). Cette solution a plusieurs avantages. Les données récupérées sont fiables et représentent l'état réel du système de production, la simulation s'initialise relativement rapidement et elle requiert une architecture matérielle minimale pour pouvoir être efficace.

La composante la plus importante de cette collecte de données est la performance des capteurs et détecteurs installés sur le système. Certains capteurs pourront nous renseigner en continu sur l'évolution d'une grandeur du système. Un capteur de température pourra par exemple nous donner en continu les conditions à l'intérieur d'un four. Toutefois, en général, beaucoup de données ne sont pas mesurées de cette manière. Prenons l'exemple de la position de véhicules autoguidés (AGV) dans un atelier. Des détecteurs disposés à certains points du circuit (intersections, postes de travail, etc.) suffisent pour piloter la flotte. Le problème est qu'une grande incertitude nait de l'utilisation de tels dispositifs. En effet, lorsqu'un AGV est en face d'un détecteur, on connaît à la fois sa position et la date courante. Dès qu'il se déplace entre deux détecteurs, sa position est totalement inconnue (exceptée l'information disant qu'il se situe sur le tronçon entre les deux détecteurs), et sa date de passage devant le prochain détecteur est indéterminée. Il est impossible que l'ensemble de la flotte d'AGV ne se situe en face des détecteurs présents sur le système à chaque moment où le simulateur a besoin des informations. De ce fait, le MES n'a accès, à chaque instant, qu'à une vue partielle de l'état complet de l'atelier : c'est ce que l'on a pu définir comme étant les incertitudes spatiales et temporelles du MES dans [Cardin et Castagna, 2006a].

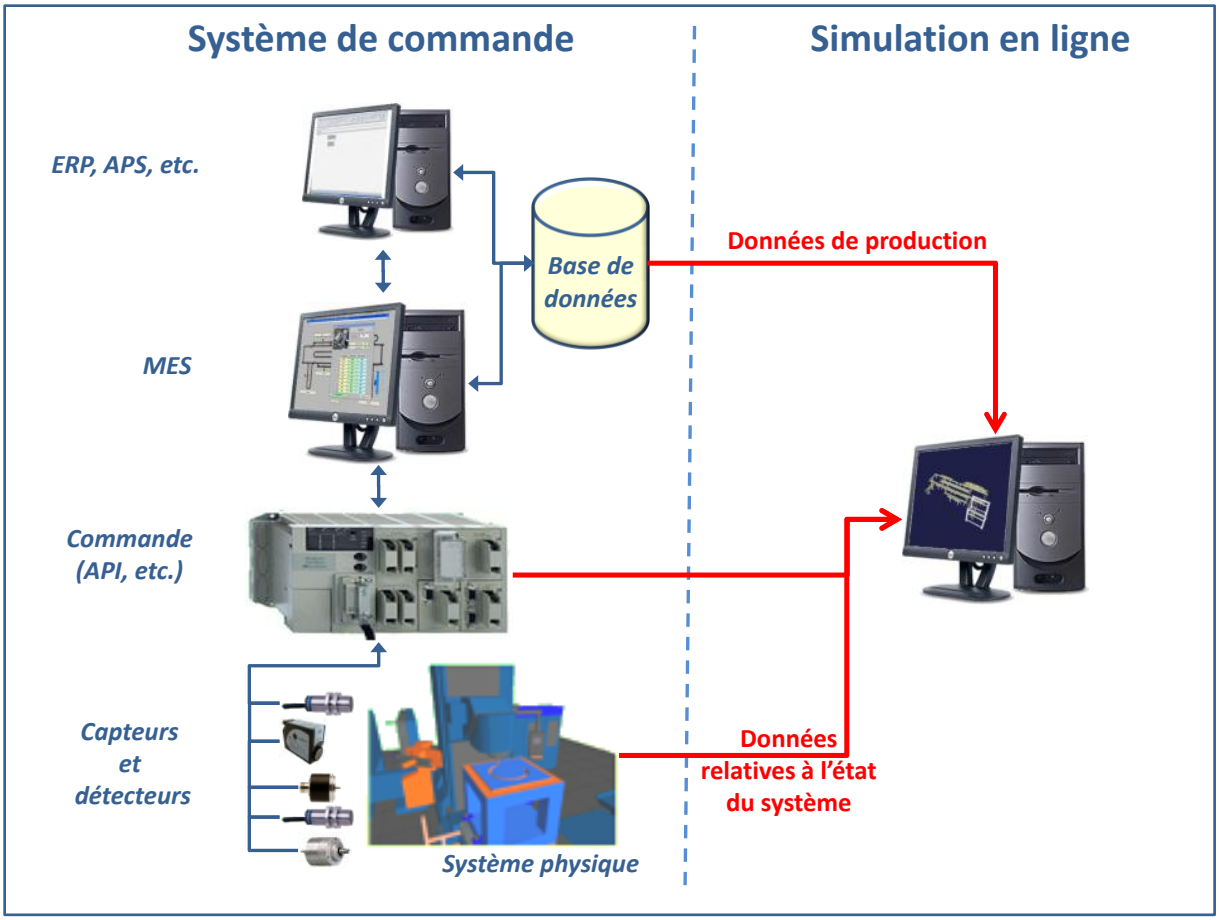

Figure 19 Utilisation des informations directement issues du système réel.

Nous avons de plus pris le parti dans ces travaux de ne strictement utiliser que les capteurs et détecteurs utilisés par la commande. Pour reprendre l'exemple précédent, l'utilisation des technologies telles qu'un module GPS ou un système de reconnaissance d'image serait envisageable pour obtenir la position exacte des AGV, mais peut également s'avérer très coûteuse, d'autant qu'il sera souvent indispensable de coupler plusieurs de ces technologies pour obtenir l'ensemble des informations nécessaires à la définition de l'état du système. Cette contrainte permet de limiter les coûts d'installation de la simulation en ligne sur une installation en production.

La conclusion que nous avons tiré de l'utilisation de cette solution est qu'elle n'est pas applicable en l'état dans le cas général, du fait du manque de disponibilité de certaines des données, prépondérantes dans la définition de l'état du système. Toutefois, elle est à conseiller dans les rares cas particuliers permettant son utilisation principalement grâce à la fiabilité des données obtenues et à la simplicité de mise en place de la solution. Nous citerons par exemple le contrôle aérien où la position, le plan de vol et la trajectoire actuelle de chaque appareil sont parfaitement connus à chaque instant (voir par exemple [Rogers et Flanagan, 1991]).

III.2 Utilisation d'un simulateur temps-réel

Depuis quelques années, la plupart des éditeurs de logiciels de simulation proposent des versions dites « temps-réel » de leurs outils. Ces simulateurs ont la particularité de ne

pas tourner à vitesse maximale mais de caler leur horloge sur le temps du système réel (souvent appelé *wall clock*), d'où le nom de simulateur temps-réel. Ces simulateurs ont été développés pour répondre à des besoins d'utilisation de la simulation en temps qu'émulateur de système physique. L'idée initiale est de réduire la durée de mise au point de systèmes complexes de production en réalisant les tests du système de commande en le couplant sur cet émulateur, sans attendre l'existence du système physique.

Notre idée est d'utiliser un tel simulateur, tournant parallèlement à la production. Ce simulateur réagirait aux sollicitations de production venant du MES ou de l'ERP, tout comme le système de production réel. Ainsi, il permet de créer une « image » permanente du système réel. L'idée est d'utiliser l'état de ce simulateur temps-réel pour initialiser le simulateur en ligne (Figure 20).

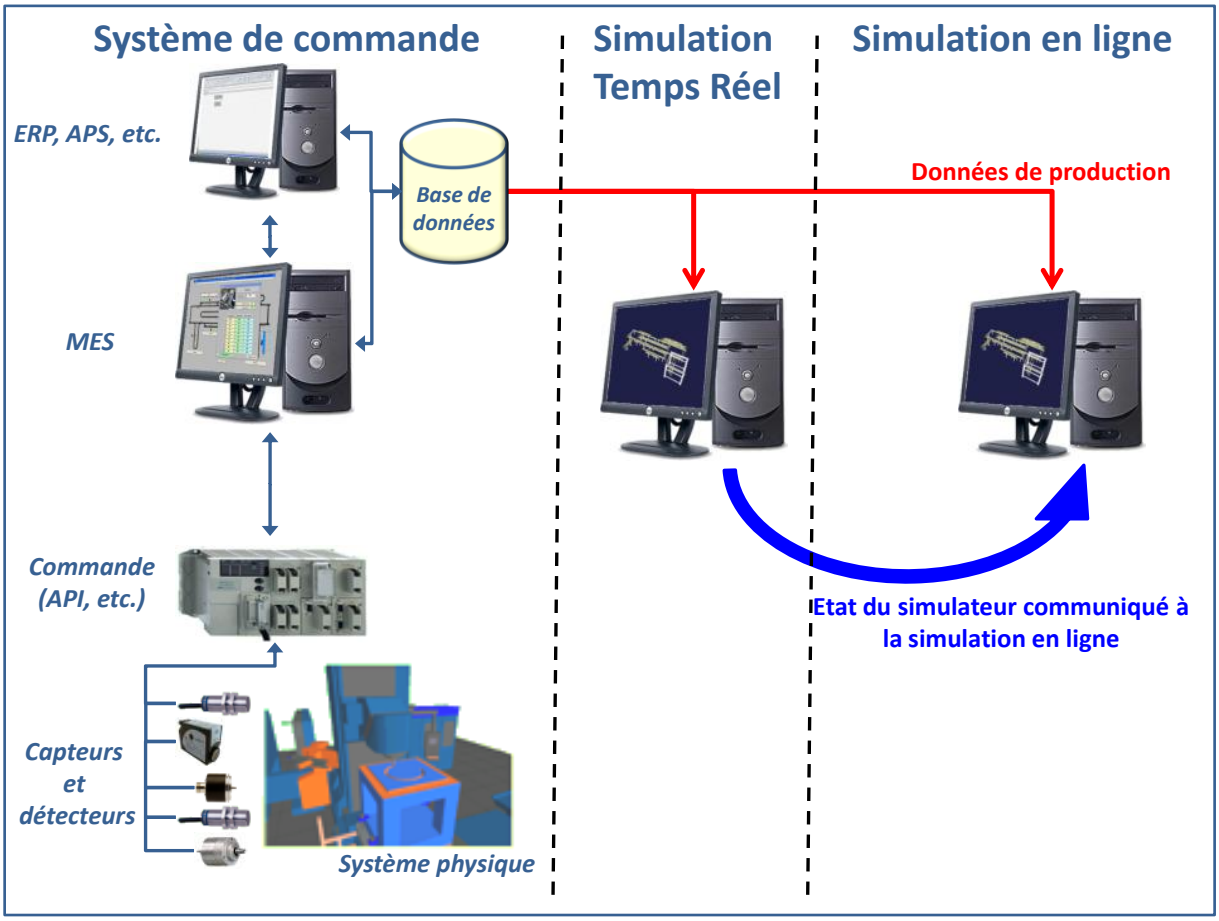

Figure 20 Utilisation d'un simulateur temps-réel

Cette solution offre deux avantages importants par rapport à l'utilisation directe des informations issues du système réel. Le premier est qu'elle supprime le problème des incertitudes spatiales et temporelles vues précédemment. Nous reviendrons largement sur cet aspect dans le chapitre suivant. Le second est la grande simplicité de définition de l'état du système, qui serait directement une « recopie » de l'état d'un simulateur (le temps-réel) vers un autre simulateur (en ligne).

Remarquons que notre proposition rappelle l'approche de type modèle parent/modèle enfant développée dans [Hanisch *et al.*, 2005]. Cette approche consiste en l'utilisation de

modèles dits *enfants* pour la prévision, démarrant avec des données initiales collectées depuis un modèle dit *parent*. Celui-ci tourne en parallèle du système réel. L'application proposée par les auteurs est réalisée sous un tableur, ce qui permet de proposer en plus une génération automatique des modèles enfants, ce qui ne nous semble pas approprié ici, au vu de la complexité de ces derniers. Un simple paramétrage des modèles est suffisant à notre application. L'hypothèse implicitement associée à cette solution est le fait que l'état du simulateur est à tout instant considéré comme une estimation suffisante de l'état du système réel.

Plusieurs problèmes se posent néanmoins lors de l'application de cette solution. Tout d'abord, l'initialisation de ce simulateur : celui-ci doit démarrer sur un état connu du système de production, qui sera typiquement choisi vide d'entité, l'ensemble des ressources étant libres (*empty and idle*). De plus, les défauts dans la modélisation du système (voir au chapitre I.3.1) et les différents aléas qui surviennent au cours de toute production entraînent une déviation sur le résultat final. L'aspect additif de ces écarts au cours du temps finirait par avoir un effet qui peut être loin d'être négligeable.

Pour répondre à cela, plusieurs contraintes doivent être imposées pour garantir l'applicabilité de la solution. Tout d'abord, le simulateur doit être extrêmement fidèle pour limiter au maximum les erreurs dues à la modélisation. Ensuite, le système étudié devra avoir une fréquence d'occurrence d'aléas faible pour limiter leur impact sur l'écart entre le réel et le simulé. Enfin, il sera préférable de considérer des systèmes qui reviennent régulièrement à un état connu (généralement vide). On pourra citer par exemple des systèmes de production qui se vident chaque jour en fin de journée. On parle de systèmes à cycles régénératifs. Ceci permettra à la simulation d'être réinitialisée, et donc de rattraper les erreurs qui auront été commises.

La classe de systèmes qui peut être considérée est donc restreinte, mais cette solution a l'avantage de constituer une alternative possible et crédible pour des applications de taille limitée qui ne peuvent se permettre les investissements nécessaires à l'utilisation complète de la simulation en ligne.

III.3 Le concept d'observation par simulation

Comme nous avons pu le voir dans les paragraphes précédents, les deux solutions que nous avons initialement envisagées se sont avérées bien adaptées pour des classes restreintes d'application. Dans un souci de construire une solution applicable à la classe de systèmes la plus élargie possible, nous avons donc eu l'idée d'hybrider les deux solutions proposées précédemment comme présenté dans la Figure 21. Ainsi, nous pensons gagner les avantages du simulateur temps-réel au niveau de la disponibilité des données et ceux de l'utilisation directe des données du système réel au niveau de la fiabilité des données.

III.3.1 La reconstruction d'état

III.3.1.1 Définition

Un système automatisé est classiquement décomposé en :[Merlaud *et al.*, 1995]

- Une partie opérative, qui assure les transformations (d'espace, de temps ou de forme) des matières d'œuvre permettant d'élaborer la valeur ajoutée recherchée ;
- Une partie commande, capable de reproduire un modèle de fonctionnement exprimant le savoir-faire humain et qui commande la partie opérative pour obtenir les effets voulus.

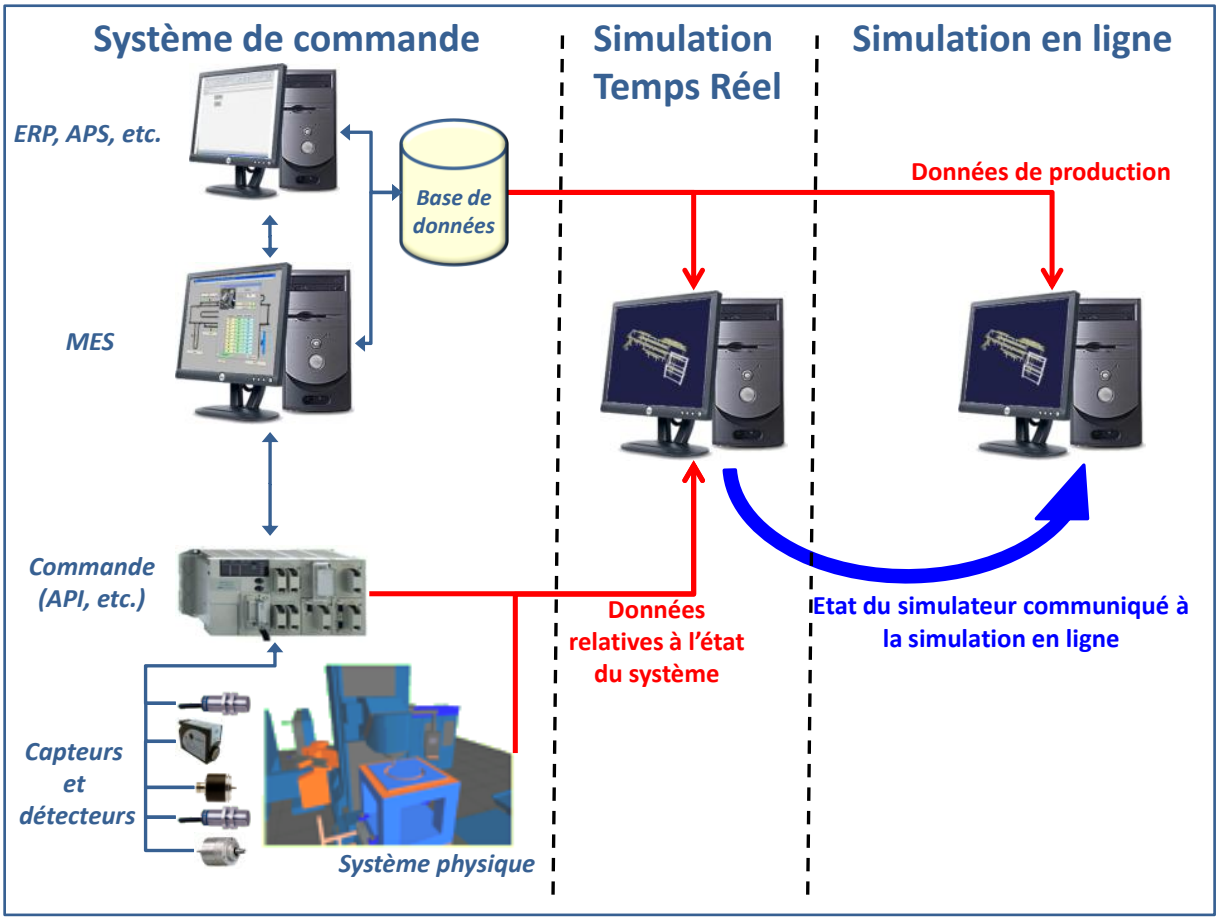

Figure 21 L'observation par simulation

La Figure 22, inspirée d'une décomposition disponible dans [Merlaud *et al.*, 1995], présente les relations existant entre ces deux parties.

L'ensemble des variables d'entrée du système peut être décomposé en consignes envoyés par les opérateurs humains et en messages envoyés depuis les autres composantes de l'architecture (MES, etc.). Ces variables sont considérées comme connues dans notre approche. Les flux de matière d'œuvre sont en général associés à un flux d'informations, qui est inclus dans l'ensemble des variables de sortie du système. Cet ensemble peut être décomposé en deux sous-ensembles :

- $y_m = y_{om} \cup y_{cm}$: sous-ensemble union du sous-ensemble des variables mesurées via des capteurs ou détecteurs du système au niveau de la partie opérative y_{om}, et du sous-ensemble des variables directement contenues dans la partie commande y_{cm}. Cet ensemble comprend en général également les signalisations à direction de l'opérateur humain et les messages envoyés par la commande aux autres

composantes de l'architecture. Cet ensemble sera par la suite appelé *ensemble des variables mesurées* ;

- y_{o_u}: sous-ensemble des variables non-mesurables sur la partie opérative y_{o_u}. Cet ensemble contient l'ensemble des variables qui ne sont pas mesurables technologiquement et celles qui ne sont pas mesurées car elles ne sont pas nécessaires à la partie commande. Cet ensemble sera par la suite appelé *ensemble des variables non-mesurables*.

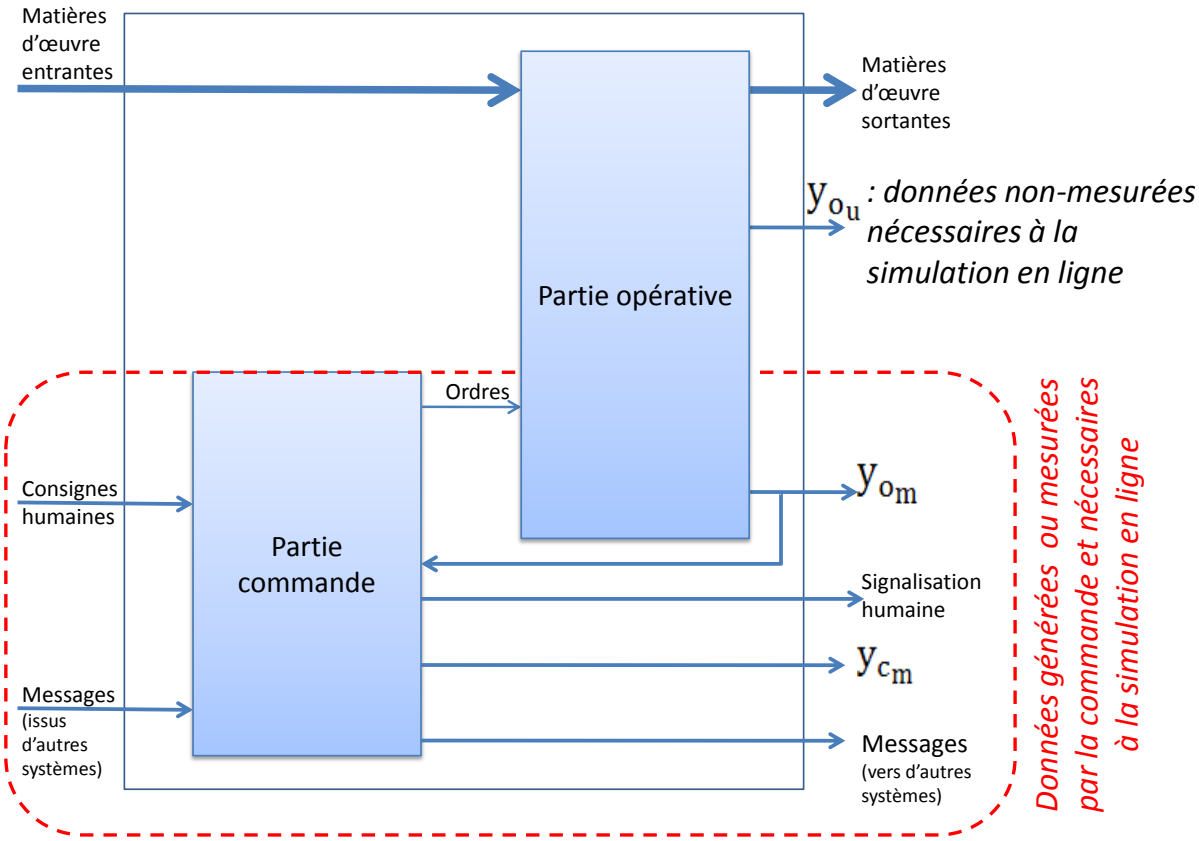

Figure 22 Décomposition d'un système automatisé

Le principe d'un observateur est de reconstruire une partie des variables non-mesurables du système en appliquant l'ensemble des variables connues du système à un modèle mathématique de celui-ci. Une condition *sine qua non* est donc de pouvoir construire un modèle du système. Celui-ci est en général basé sur une modélisation des lois physique le régissant et intégrant le paramétrage et l'ensemble des perturbations connues. En reprenant la notation précédente, l'objectif de l'observateur est donc de reconstruire tout ou partie du sous-ensemble y_{o_u}.

Le Chapitre II nous a permis de proposer l'utilisation de la définition suivante pour la notion d'état d'un système : « collection minimale d'informations avec lesquelles le comportement futur du système peut être déterminé de façon unique en l'absence de hasard ». Si nous notons $x(t)$ l'état de notre système à l'instant t, la première question est donc de savoir quelles sont les informations contenues dans ce vecteur x. Généralement, l'ensemble y_m est entièrement inclus dans l'état du système. Il est souvent indispensable de

connaitre également les consignes et messages provenant des autres composantes de l'architecture. Enfin, une partie de l'ensemble des variables non-mesurables y_{o_u} peut également être nécessaire. Ceci pose un problème particulier, puisque, par définition, ces variables ne sont pas connues.

Un système est dit observable si l'observation de ses entrées et sorties pendant un intervalle de temps fini $[t_0; t]$ permet de retrouver l'état initial $x(t_0)$. Un système est dit reconstructible si l'observation de ses entrées et sorties pendant un intervalle de temps fini $[t_0; t]$ permet de retrouver l'état final $x(t)$. Un observateur peut donc :

- soit observer l'évolution de certaines variables sur un intervalle pour estimer la valeur d'autres variables au début de l'observation ;
- soit reconstruire des variables non-mesurables (champ de température à l'intérieur d'un four, etc.) pour construire le vecteur x, vecteur d'état du système regroupant toutes les variables nécessaires à la définition de l'état actuel du système.

Au vu de notre objectif, c'est principalement la seconde possibilité que nous allons aborder. Le but est alors de trouver une approximation x' du vecteur d'état x à partir des variables observées y_m. En général, dans notre cas, il s'agit au temps τ de trouver une fonctionnelle F,

$$x'(\tau) = F[y_m(t), \tau], \qquad t_0 \leq t \leq \tau$$

telle que $x'(\tau) \approx x(\tau)$, où t_0 représente la date initiale des observations et $x'(\tau)$ représente *l'état reconstruit*. Il est à noter que $F[y_m(t), \tau], t_0 \leq t \leq \tau$ est une fonction des observations passées $y_m(t)$, $t_0 \leq t \leq \tau$, et ne dépend pas des observations futures $y_m(t)$, $t > \tau$. La fonctionnelle est construite grâce à la connaissance des lois physiques régissant la partie opérative et de la modélisation de la partie commande. Étant donné que la variable d'état à reconstruire est en général un vecteur, la fonctionnelle est de la même dimension.

Une fois l'état reconstruit obtenu, les méthodes classiques peuvent s'employer en remplaçant l'état actuel par l'état reconstruit, considéré alors comme une estimation suffisante de l'état réel. De récents travaux dans des thématiques différentes font état de l'utilisation en ligne de la simulation au sens large (on pourra citer par exemple la régulation thermique d'un four [Jaklic *et al.*, 2007]), souvent pour la commande de processus continus. À partir du jeu de capteurs installés sur l'installation, les modèles formels (ici un jeu d'équations thermiques) reconstruisent l'état du système selon les théories d'observation et de reconstruction d'état. Grâce à cela, les auteurs peuvent obtenir des valeurs non mesurables qui sont réinjectées dans les modèles pour obtenir des prévisions plus fiables sur le devenir à court terme de l'installation selon la décision prise (variation de la température du four).

III.3.1.2 Analogie dans le cas des modèles de simulation

Cette définition a déjà été appliquée dans le cas des systèmes linéaires, non-linéaires ou encore des systèmes à retard. Dans le cas des modèles de simulation, les concepts sont transposables, même s'il n'est pas toujours possible d'écrire les relations sous une forme mathématique simple du fait des lois de comportement régissant le système.

Dans notre cas, le système physique est perçu à travers un nombre fini n de capteurs ou détecteurs. De ce fait, le vecteur des variables de sortie mesurées est de la forme :

$$y_m = \begin{bmatrix} y_{m,1}(t) \\ y_{m,2}(t) \\ \vdots \\ y_{m,i}(t) \\ \vdots \\ y_{m,n}(t) \end{bmatrix}$$

L'observateur n'a donc à sa disposition qu'un ensemble de mesures localisées à inclure dans la fonctionnelle, localisation qui influera bien évidemment dans la construction de cette dernière. Le changement d'état de chacune de ces variables à des dates notées θ_{i,d_i} (avec i l'identifiant unique du capteur correspondant et d_i le numéro de l'évènement – plusieurs évènements concernant le même capteur pouvant intervenir, il faut pouvoir les différencier) correspond à un évènement qui a fait le système évoluer.

À chaque θ_{i,d_i}, c'est-à-dire à chaque évolution du système, l'expression de chaque composante de la fonctionnelle est susceptible d'être modifiée. Ces composantes pourront par exemple être une régression linéaire sur la position d'un objet entre sa vitesse moyenne estimée et la distance à parcourir entre deux capteurs, puis à partir de la date d'évolution du capteur de fin du tronçon, une seconde régression linéaire entre les capteurs suivants.

Les paragraphes suivants donnent deux exemples d'application de cette méthode.

III.3.1.2.1 Application au déplacement d'un colis sur un convoyeur

Prenons le cas simple du déplacement d'un colis entre deux capteurs sur un convoyeur de vitesse moyenne estimée V (Figure 23). La variable d'état que nous souhaitons reconstruire est notée $x'(t)$. Cette variable est une estimation du vecteur d'état noté $x(t)$, qui correspond dans notre cas simplement à la position $P(t)$ de ce colis au cours du temps :

$$x(t) = P(t), \qquad 0 \leq P(t) \leq 1$$

$P = 0$ correspond à la position face au capteur 1, alors que $P = 1$ correspond à la position face au capteur 2. La variable observée est constituée de l'état des deux capteurs :

$$y_m = \begin{bmatrix} y_{m,1}(t) \\ y_{m,2}(t) \end{bmatrix}$$

avec $y_i(t) = 1$ si le colis est devant le capteur i, $y_i(t) = 0$ sinon. Nous souhaitons reconstruire la variable d'état au temps τ. À cet instant, les trois propriétés suivantes sont vérifiées :

$\exists! \, t_1 \in [0, \tau[\;\; tel\ que \;\; y_{m,1}(t_1) = 1$ $\equiv$ *Un colis est rentré à la date $t = t_1$ sur le convoyeur ;*

$\forall \, t \in [0, \tau], t \neq t_1, \quad y_{m,1}(t) = 0$ $\equiv$ *Un seul colis est entré sur le convoyeur ;*

$\forall \, t \in [0, \tau], \quad y_{m,2}(t) = 0$ $\equiv$ *Aucun colis n'est sorti du convoyeur.*

Nous définissons alors une fonctionnelle F telle que :

$$x'(t) = F(y(t)) = \begin{cases} 0 & si\ y_{m,1}(t) = 1 \\ 1 & si\ y_{m,2}(t) = 1 \\ V \times (t - t_1) & sinon \end{cases}$$

avec t_1 la dernière occurrence de $y_1(t) = 1$. Comme explicité précédemment, on peut s'apercevoir que cette fonctionnelle ne contient que des évènements passés ($t \leq \tau$).

On obtient donc :

$$x'(\tau) = F(\tau) = V \times (\tau - t_1)$$

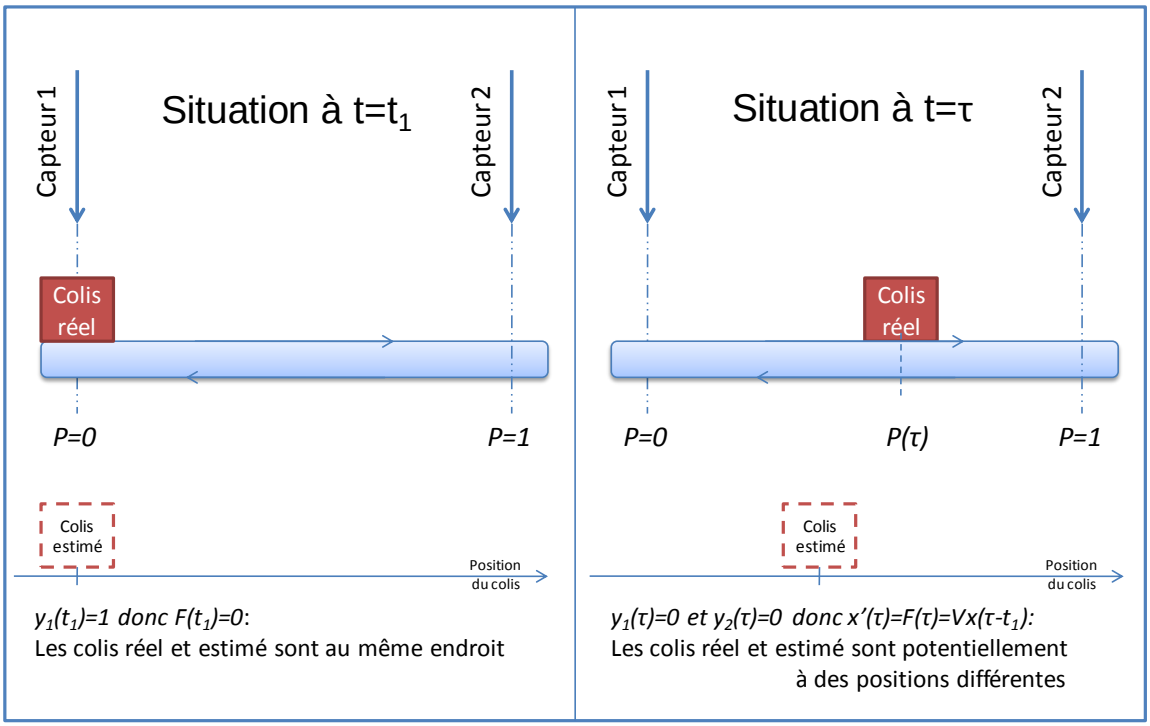

Figure 23 Évolution d'un colis sur un convoyeur entre deux capteurs

III.3.1.2.2 Application au remplissage d'un réservoir

Considérons désormais le réservoir de section S muni de trois capteurs de niveaux répartis équitablement (deux aux extrémités et un au milieu) présenté Figure 24. Le conduit supérieur permet de le remplir grâce à des vannes disposées en amont avec un débit D_r, tandis qu'une évacuation permet d'alimenter les clients en liquide selon leur demande avec un débit D_e. Comme dans l'exemple précédent, on ne peut pas connaître exactement le niveau de l'eau à la seule lecture de la variable observée, qui est ici le vecteur d'état des trois capteurs de niveaux :

$$y_m = \begin{bmatrix} y_{m,1}(t) \\ y_{m,2}(t) \\ y_{m,3}(t) \end{bmatrix}$$

La variable d'état correspond au niveau $N(t)\epsilon[0;1]$ de liquide dans le réservoir. Par application de la méthode proposée précédemment, nous obtenons $\forall\tau \geq t_1 > 0$l'état reconstitué :

$$x'(\tau) = F(y(\tau)) = \begin{cases} 1 & si\ y_{m,1}(\tau) = 1 \\ 0.5 & si\ \uparrow y_{m,2}(\tau) = 1\ ou\ \downarrow y_{m,2}(\tau) = 1 \\ 0 & si\ y_{m,3}(\tau) = 0 \\ \dfrac{D_r(\tau) - D_e(\tau)}{S} \times (\tau - t_1) + x'(t_1) & sinon \end{cases}$$

avec t_1 la dernière date d'évolution du vecteur des variables observées.

Ces deux exemples ne revêtent pas de caractère exhaustif, mais illustrent bien la difficulté de construction de la fonctionnelle. La différence majeure entre les deux est au niveau du capteur 2 du réservoir. En effet, celui-ci pouvant se remplir ou se vider, le capteur peut avoir deux comportements différents. L'utilisation des fronts dans la définition de la fonctionnelle traduit le besoin d'une fonction de mémoire.

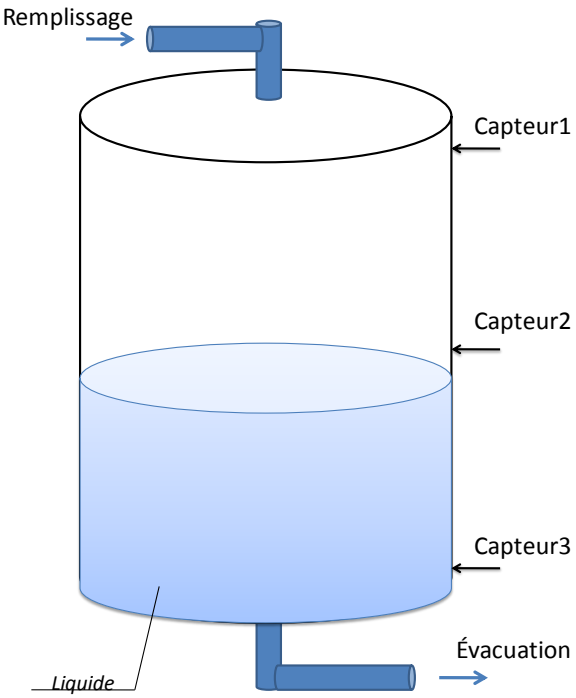

Figure 24 Réservoir muni de trois capteurs de niveaux

III.3.2 Les données nécessaires

Il est évident que la précision de l'observateur dépendra du nombre et de la disposition des capteurs et détecteurs présents sur le système étudié. Si l'on avait par exemple un nombre infini de capteurs tout ou rien sur un convoyeur, alors l'observateur ne serait que le reflet de la valeur de ceux-ci, et serait donc capable de donner la position exacte de tous les éléments sur celui-ci. Bien sûr, ce nombre est en général bien moins important. La question est alors de déterminer le nombre et la disposition adéquats de capteurs à utiliser pour assurer une mise en place correcte de l'observateur.

Une condition suffisante est d'avoir des capteurs à tous les points de décision : un capteur pour savoir qu'une décision va être prise, puis suffisamment de capteurs pour savoir quelle décision l'a été. La Figure 25 explicite ce point sur l'exemple d'un colis arrivant à une intersection et pouvant aller sur deux convoyeurs. La Figure 25a montre une implantation satisfaisante de capteurs de présence. Cette architecture a pour inconvénient de ralentir la prise en compte de l'information : on ne connait le choix du colis qu'après que celui-ci soit sorti de la zone de bifurcation. La Figure 25b montre une implantation différente, n'utilisant qu'un capteur de présence mais également un capteur sur l'actionneur permettant le choix (ici un vérin rotatif). Cette architecture résout le problème décrit précédemment, mais n'est pas toujours applicable suivant les solutions techniques retenues.

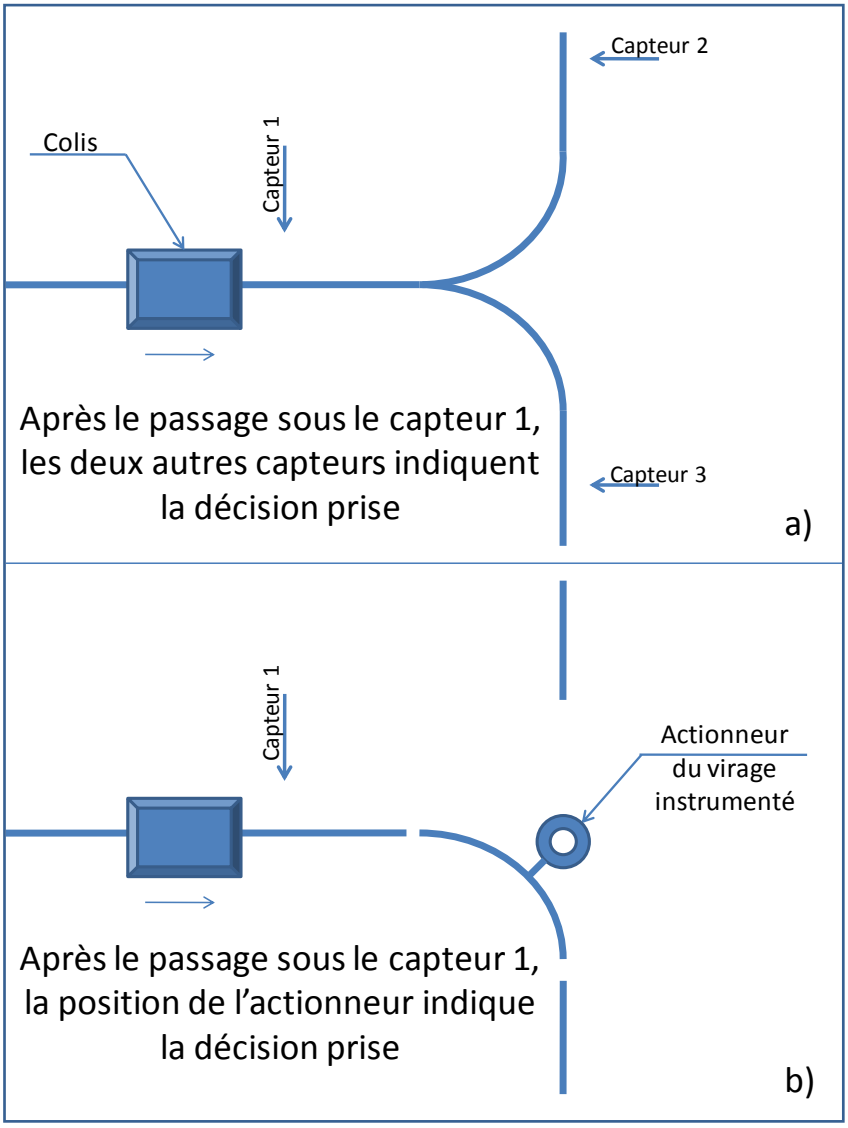

Figure 25 Condition suffisante à l'implantation de l'observateur
a) Prise d'information après la décision
b) Prise d'information au moment de la décision

Malheureusement, il n'est pas toujours possible de respecter cette condition – le système est déjà existant et n'est pas modifiable, le prix des capteurs concernés est prohibitif, etc. La solution la plus efficace dans ce cas extrême est de dialoguer directement avec la commande du système. En effet, celle-ci connaît l'information qui nous intéresse

puisqu'elle est le résultat de sa décision. L'inconvénient majeur de cette solution est le temps de cycle de la commande qui est en général très court (quelques millisecondes), ce qui impose une collecte de l'information très performante. Il faut donc s'assurer que l'observateur ait le temps d'obtenir l'information avant qu'elle ne disparaisse de la commande. Ce que nous préconisons alors est un système de file d'attente classique inscrit dans la mémoire de la commande. À chaque fois qu'une décision est prise, elle est mémorisée en dernière position dans la file d'attente correspondante. Lorsque l'observateur a besoin de connaitre la décision prise à cet endroit, il récupère la première décision contenue dans la file d'attente, et la supprime de cette file. Ainsi, toutes les décisions sont prises en compte sans risque d'en oublier. Cette solution a toutefois l'inconvénient d'être difficile à mettre en place, et d'influer sur la programmation de la commande du système. La suite de ce document montrera notamment comment nous avons sorti cette file d'attente des programmes de la commande pour ne pas influer sur celle-ci lors de la construction de notre observateur.

III.3.3 Recalages

Les paragraphes précédents ont détaillé le mode opératoire qu'il est préférable d'utiliser pour assurer que l'observateur mis en place puisse contenir toutes les informations nécessaires à la définition de l'état complet du système. Ces données sont de deux ordres :

1. Les données directement observées issues des capteurs, reflétant les décisions de la commande ou contenues dans le système d'information ;
2. Les données reconstruites.

Afin d'assurer la cohérence entre le comportement du système réel et l'état du système réel, ces dernières sont le plus souvent possible recalées sur leur valeur réelle tout au long du fonctionnement de l'observateur. Pour les données provenant du système d'information, cela ne pose pas de problème, puisqu'il suffit de lire la valeur et de l'assigner à la variable de l'observateur correspondant. Reste à s'affranchir des problèmes de cohérence de données : nous avons vu au chapitre II qu'il est préférable dans cette optique de faire travailler tous les éléments de l'architecture de commande sur les mêmes données.

Pour ce qui est des données du système physique, on a pu voir précédemment qu'il en était tout autrement. Ce chapitre s'intéresse plus particulièrement au traitement du recalage de ces données à l'intérieur de l'observateur. En effet, il n'est pas du tout trivial de prendre en compte des évènements extérieurs dans le déroulement en temps-réel d'une simulation.

Pour cela, il faut définir la manière dont on gère la dualité existant entre l'évènement réel et ce même évènement simulé : lors de l'occurrence de l'un des deux, que fait-on de l'autre ?

Tout d'abord, éliminons la possibilité de l'occurrence simultanée : nous nous plaçons sous une des hypothèses habituelles de la théorie des automates : deux évènements d'origine distincte ne peuvent se dérouler au même instant. La durée de cet instant est définie par la durée d'un cycle automate. Ici, elle sera définie par la durée de prise en compte des évènements extérieurs par la simulation.

L'analyse nous pousse ensuite à considérer deux cas :

1. L'évènement simulé est en avance par rapport au réel ;
2. Le réel est en avance par rapport au simulé.

Nous proposons de construire un observateur sous le formalisme des réseaux de Petri sur un exemple très simple pour pouvoir en extraire le principe de fonctionnement du recalage. Nous n'observerons ici qu'une seule variable du système qui est la position d'un vérin.

III.3.3.1 Un exemple : le vérin

Reprenons plus en détail l'exemple évoqué en début de chapitre. Considérons le système présenté Figure 26 constitué de :

- Un vérin double effet ;
- Deux capteurs de fin de course du vérin ;
- Une commande implantée en Ladder Diagram (LD) sur un Automate Programmable Industriel (API).

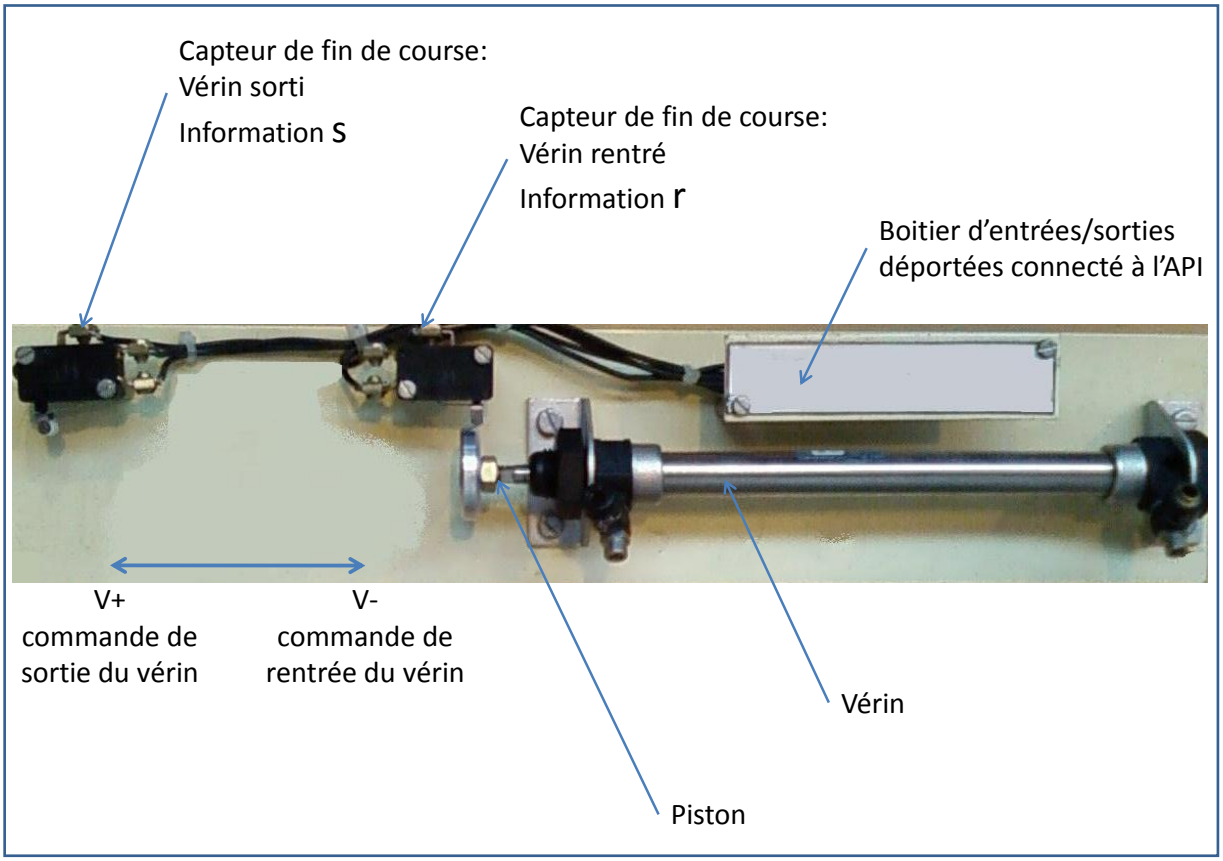

Figure 26 Le vérin instrumenté

La Figure 27 présente la commande du système. Le comportement attendu de ce système est un va-et-vient incessant entre les deux capteurs dès la mise en *run* de l'automate.

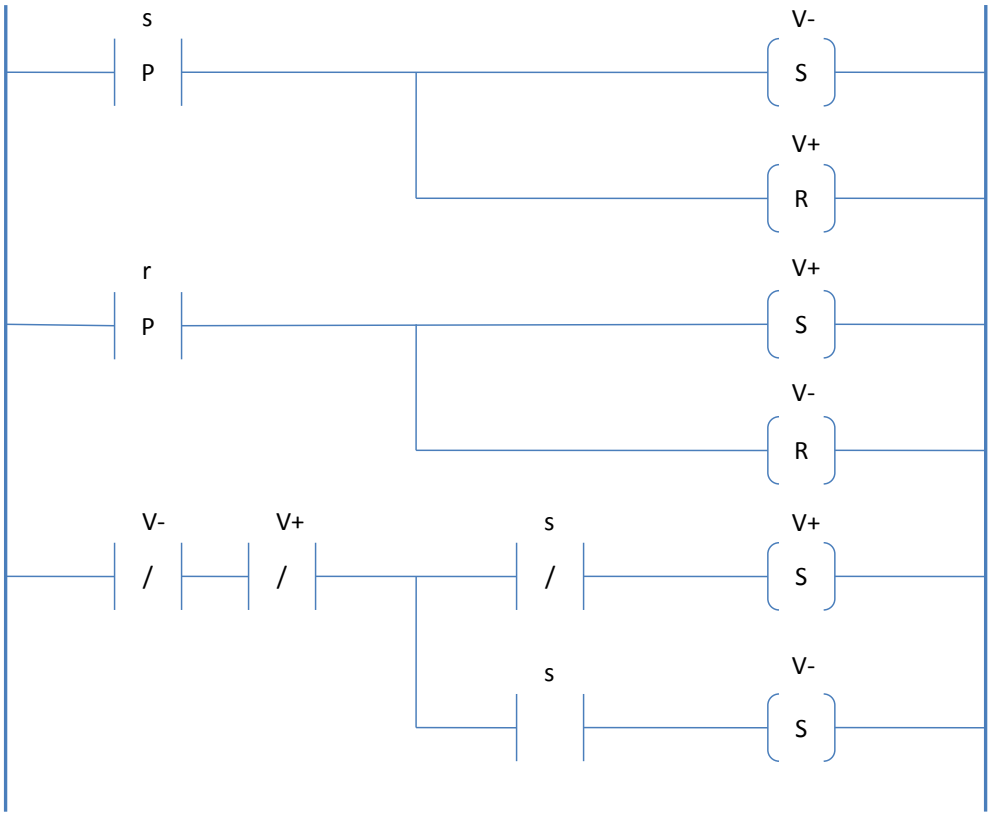

Figure 27 La commande implantée en Ladder Diagram

Des mesures temporelles ont été réalisées sur le système pour réaliser l'observateur :

- Temps de rentrée du vérin : $T_R = 10\ secondes$;
- Temps de sortie du vérin : $T_S = 10\ secondes$;

On définit comme précédemment la position du vérin par une variable $P \in [0; 1]$ avec $P = 0$ lorsque $r = 1$ et $P = 1$ lorsque $s = 1$. La position initiale du vérin au lancement de l'observateur est définie en $P = 0$.

III.3.3.2 Observateur en avance

Si l'observateur est en avance, cela signifie que le vérin simulé est déjà arrivé sur un capteur alors que le réel est toujours dans sa course. Pour bien comprendre les « mécanismes de recalage » nous proposons de les représenter en utilisant le formalisme des réseaux de Petri [Murata, 1989]. Le réseau de Petri de la Figure 28 présente la solution que nous avons retenue pour faire l'observateur attendre le simulateur.

Puisque l'observateur est en avance, la transition T2 est tirée la première. Un jeton passe alors de la place P1 où il permettait au vérin simulé de bouger (rentrer ou sortir) à la place P2 où il ne bouge plus car en bout de course. Lorsque l'évènement réel apparaît, un jeton passe alors en P3, et la synchronisation se réalise alors par le tir de la transition T3. L'observateur peut alors continuer.

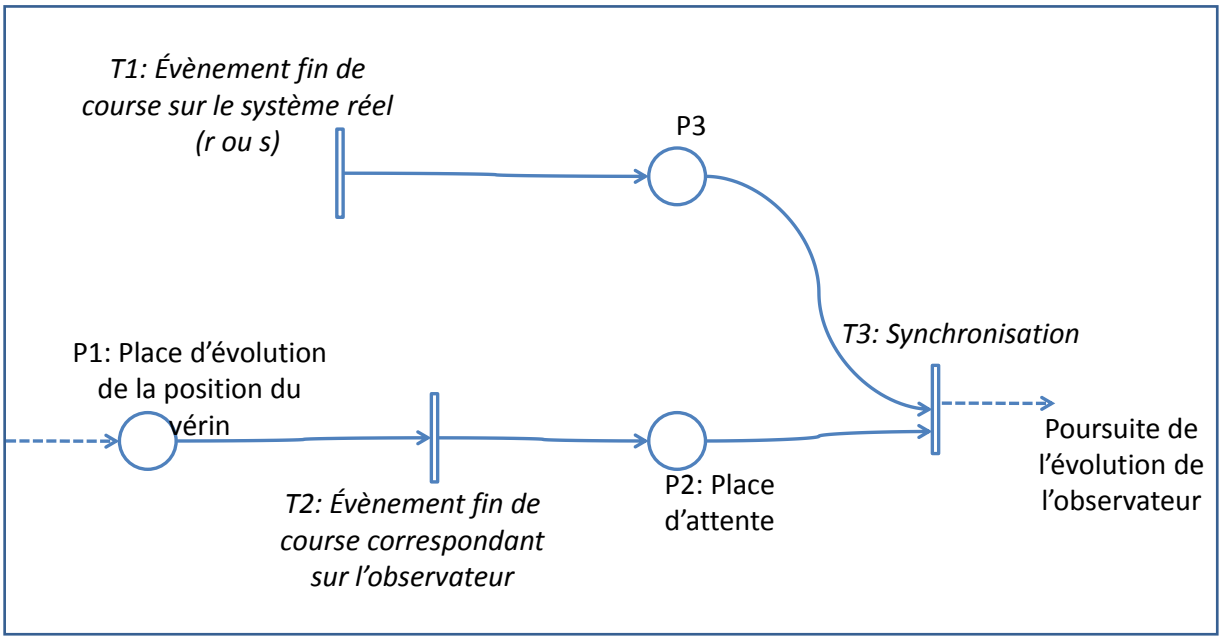

Figure 28 Gérer l'avance de l'observateur

III.3.3.3 Observateur en retard

La Figure 29 présente la solution que nous avons retenue pour le cas où l'observateur est en retard. Un jeton est donc présent en P1 lorsque la transition T1 est tirée. Une synchronisation s'effectue alors sur le tir de la transition T4. Un jeton passe alors en P4, où le recalage par rapport à la réalité s'effectue (ici on impose une position au vérin). Une fois cette action effectuée, l'observateur peut continuer d'évoluer par la transition T5. Remarquons que le traitement du retard impose la possibilité d'agir en temps-réel sur l'observateur. Dans cet exemple, cette action signifie la possibilité d'imposer à tout moment la position rentrée ou sortie au vérin dans l'observateur.

III.3.3.4 Observateur complet

Pour chaque évènement réel à prendre en compte dans l'observateur, il faut donc paramétrer puis composer les deux réseaux de Petri proposés ci-dessus, puis encore composer les réseaux obtenus pour obtenir un observateur reflétant le comportement attendu du système. Ces compositions ne peuvent évidemment s'effectuer sans une connaissance approfondie de la commande, ou tout au moins du comportement exact du système.

La Figure 30 présente l'observateur que nous avons obtenu sur cet exemple. La variable PX.t représente le temps depuis lequel un jeton est présent à la place PX.

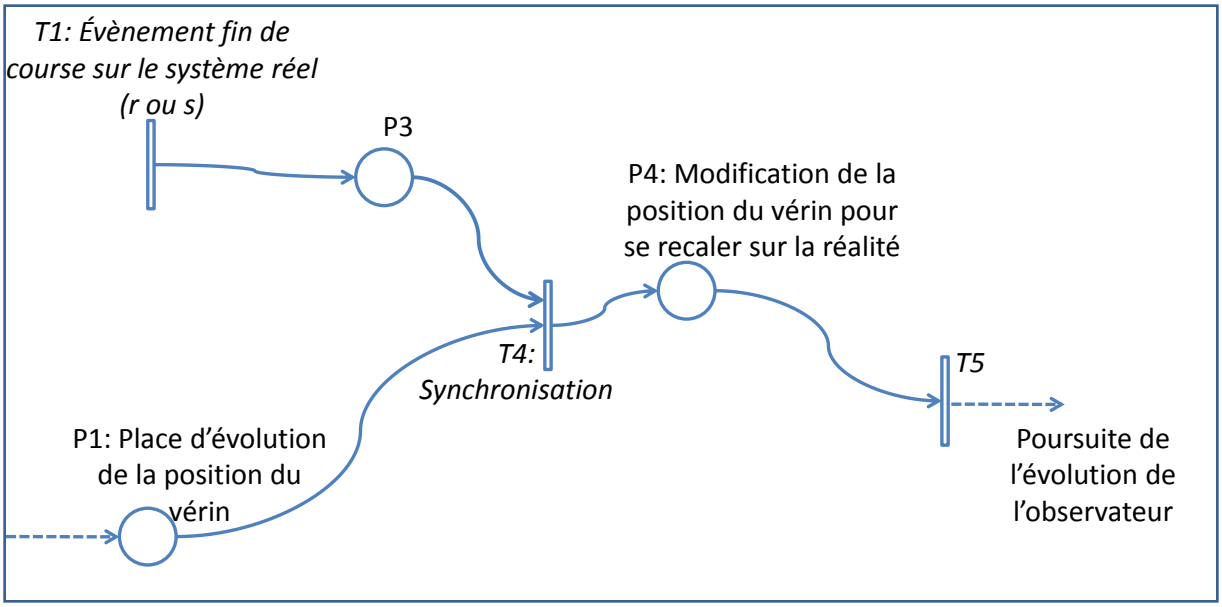

Figure 29 Gérer le retard de l'observateur

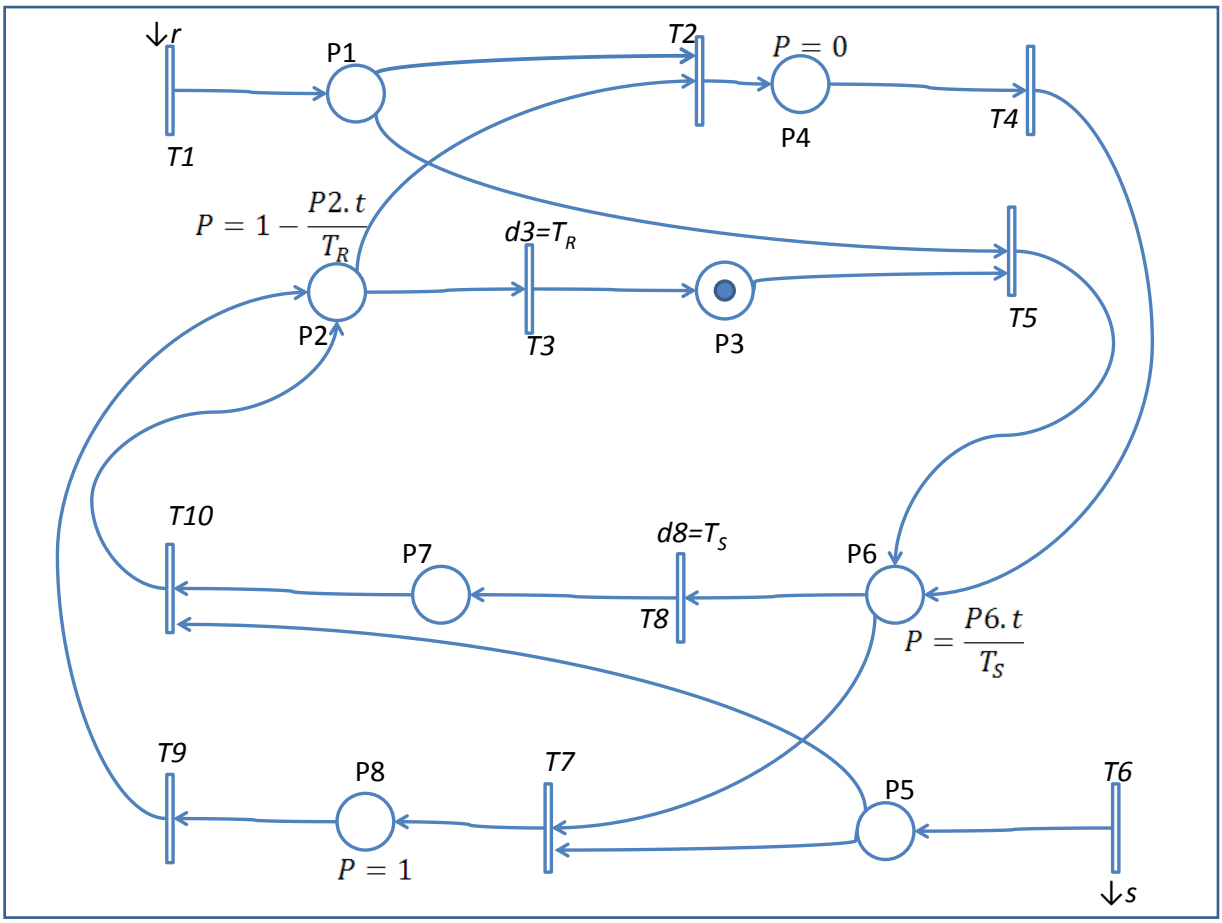

Figure 30 L'observateur complet du système vérin

Le chronogramme présenté Figure 31 regroupe toutes les combinaisons d'évolutions possibles de l'observateur de la Figure 30, ainsi que les marquages correspondant du réseau

de Petri complet. L'observateur n'évolue pas sur l'intervalle [0 ; t_1] car le système réel est dans son état initial. Lorsque celui-ci démarre (en t_1) et génère donc un « front descendant du capteur r », les transitions T1 puis T5 sont tirées.

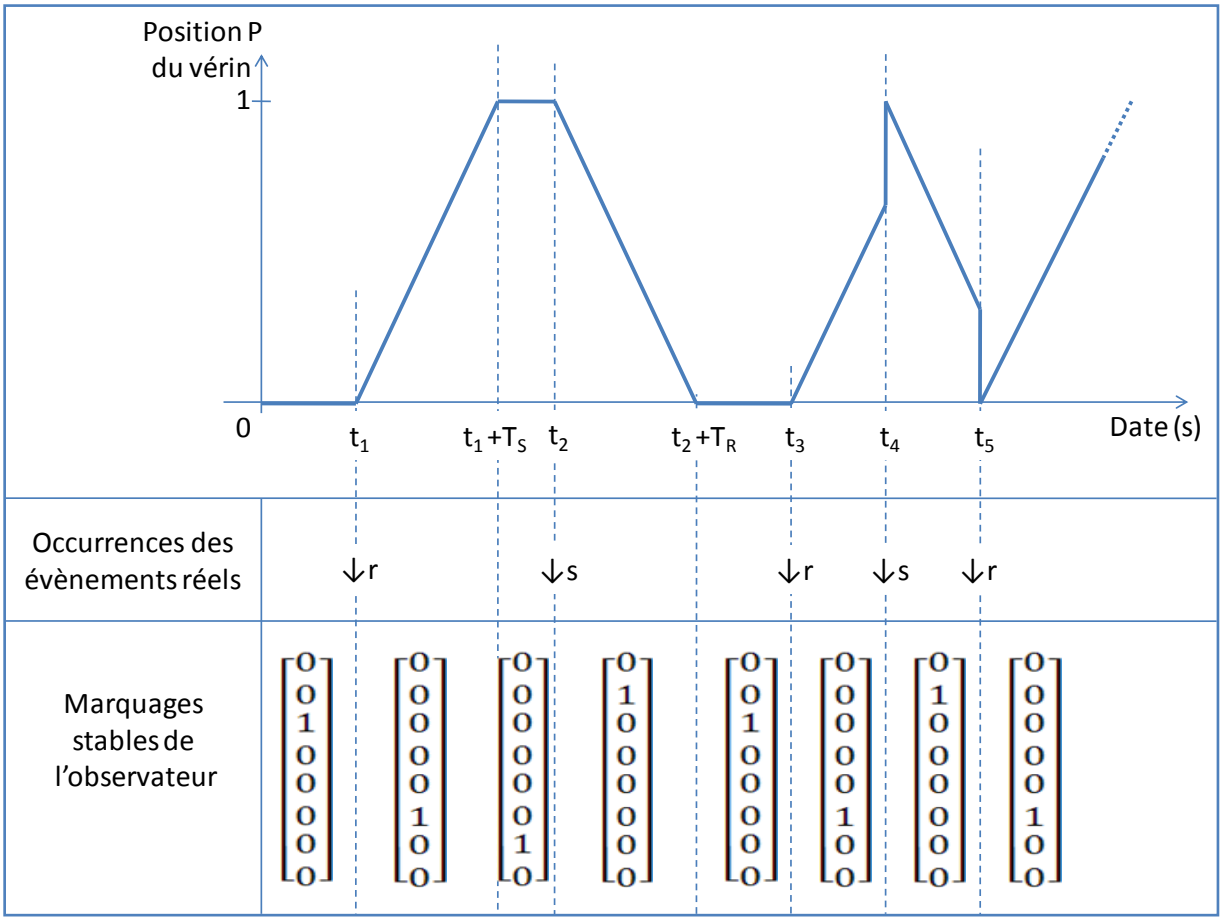

Figure 31 Évolution de l'observateur du vérin

Sur l'intervalle $[t_1 ; t_1 + T_S]$, l'observateur en P6 fait évoluer linéairement la position simulée du vérin (choix qui a été fait lors de la modélisation du système). En $t_1 + T_S$, la transition T8 est tirée car elle est franchissable depuis une durée T_S. L'observateur est alors en attente de l'évènement « front descendant du capteur s » du système réel, et la position simulée du vérin n'évolue plus ($P = 1$).

À l'occurrence de cet évènement (en t_2), les transitions T6 puis T10 sont tirées, et l'observateur estime la position du vérin par une nouvelle fonction linéaire jusqu'à la date $t_2 + T_R$ où la transition T3 est franchie. On se retrouve alors dans un état identique à l'état initial. Ces comportements sont représentatifs d'un observateur en avance sur la réalité.

Lors de la phase suivante d'estimation de l'état (place P6), l'évènement qui se déroule en t_4 implique le tir de la transition T6. Des jetons étant présents en P5 et P6, la transition T7 est tirée. La place P8 implique le forçage de la position à une position connue. En pratique, cette valeur correspond à celle qu'a le vérin lors de l'occurrence de l'évènement déclenchant le recalage. Lorsque ce recalage est terminé, l'observateur reprend son fonctionnement normal par le tir de la transition T9. Ce comportement est représentatif d'un observateur en retard sur la réalité.

III.3.3.5 Initialisation de l'observateur

Après la modélisation de l'observateur, une question importante doit être posée au sujet de l'initialisation de celui-ci. Dans notre exemple, cet exemple consiste à définir le marquage initial du réseau de Petri constitutif de notre observateur. En effet, le modèle présenté Figure 30 a résolu implicitement cette question en proposant un marquage initial de la place P3. Or, d'autres choix pourraient être faits, dont la place P7 est le plus simple représentant. Le choix de la place P3 a été fait par la connaissance que nous avions du système, qui nous indiquait une tige de vérin rentrée au démarrage du système.

Posons-nous alors cette question : et si la place P7 avait été choisie ? La première transition tirée après le démarrage du système est toujours T1, correspondant au début de la sortie de la tige. Un jeton est donc disponible en place P1. Puis, au bout du temps T_S, la transition T6 est tirée, permettant à l'observateur d'évoluer par le tir de la transition T10 (un jeton étant présent en P5 et P7). Le problème qui se pose alors est la présence du jeton en P1. En effet, la transition T2 peut être tirée, ce qui correspond à un recalage de la tige observée en position rentrée, alors que la tige réelle vient de commencer à rentrer.

Cet exemple montre que d'une mauvaise initialisation de l'observateur peut résulter une prise en compte erronée des évènements se déroulant sur le système réel. Bien évidemment, il serait possible de modifier le modèle pour trier les évènements, mais cette modification est relativement difficile à mettre en place, principalement au niveau de la robustesse du tri. De plus, comme on peut le voir sur l'exemple du vérin, il existe un laps de temps entre le début d'évolution des variables observées du système réel et le début d'évolution de l'observateur lors d'une mauvaise initialisation. Durant ce temps, l'état de l'observateur n'est pas une image de l'état du système réel. Ceci est préjudiciable, puisque l'observateur n'est pas utilisable pendant ce laps de temps, éventuellement long suivant la complexité du système.

Lorsque cela est possible, il est à notre sens beaucoup plus simple d'initialiser l'observateur sur un état parfaitement connu du système. Ainsi, aucun évènement ne doit être trié, et l'observateur commence à évoluer dès le début de l'évolution des variables observées. Il est toutefois à noter que l'initialisation de l'observateur se fait en une durée qui ne peut généralement pas être considérée comme nulle. Cette contrainte impose qu'aucune variable observée ne doit évoluer durant cette initialisation au risque de se retrouver dans le cas précédent.

Même si elle est plus simple et plus efficace, il faut noter que cette dernière solution n'est pas toujours applicable, et la solution de tri des évènements peut avoir à être tout de même développée.

III.3.4 L'observateur dans l'architecture de commande

III.3.4.1 Technologie de l'observateur

Pour résumer, et au regard des observations précédentes, on peut définir le besoin au niveau de l'observateur en quatre points :

1. L'observateur est constitué d'un ensemble de fonctionnelles liées au synoptique du système qu'il doit être capable de faire évoluer. L'observateur doit être capable,

entre deux recalages avec le système réel, de *prévoir le comportement* de notre système ;
2. L'observateur doit être capable d'intégrer des mécanismes de recalage pour corriger ses prévisions ;
3. L'observateur doit être capable de communiquer avec les autres composantes de l'architecture de commande ;
4. L'observateur doit pouvoir fournir à tout instant une image de son état, exploitable pour l'initialisation de simulateurs en ligne.

Nous proposons ici de revenir sur le choix d'un outil de simulation pour la réalisation de notre observateur. Nous avons, dans le chapitre précédent, donné un certain nombre d'exemples très simples, qui pourraient laisser à penser que l'observateur peut être simplement réalisé en utilisant un langage de programmation connu (Pascal, VB, …). Par exemple, dans le cas de notre vérin, il est évident que cette solution serait tout à fait réalisable. Mais la réalité des systèmes auxquels nous voulons appliquer nos travaux est beaucoup plus complexe. Reprenons l'exemple du convoyeur décrit en III.3.1.2.1. Nous avons indiqué que, si la position d'un colis est connue aux instants t_i (où elle est égale à x_i) et t_{i+1}(ou elle est égale à x_{i+1}), alors nous pouvons prévoir sa position $x(t)$ entre les instants t_i et t_{i+1} en utilisant la fonction linéaire :

$$x(t) = x_i + V(t - t_i) \qquad t \in [t_i; t_{i+1}]$$

Cette modélisation est tout à fait exacte dans le cas simple où l'accumulation est impossible sur le convoyeur. Dans le cas où le convoyeur permet l'accumulation, la fonction $x(t)$ devient beaucoup plus complexe parce qu'elle dépend du comportement des autres produits se trouvant aussi sur le convoyeur, en position amont. Si nous observons l'exemple de la Figure 32, nous voyons que l'utilisation du modèle simpliste du convoyeur sans accumulation nous conduirait à prévoir la position de notre colis rouge en x_{i+1} alors qu'en réalité, il est venu s'accumuler derrière la file des produits déjà accumulés.

Cet exemple simple montre que notre observateur est bien un outil de prévision (point 1 défini précédemment), mais qu'il doit permettre une prévision la plus précise possible dans le contexte d'un système complexe, où, nous l'avons déjà dit, la simulation s'impose comme le seul outil utilisable aujourd'hui.

Il faudra par contre que notre simulateur/observateur intègre les mécanismes de recalage (Point 2). Cette intégration se fera au niveau du modèle de simulation et c'est sans aucun doute le point le plus délicat de l'implémentation de notre observateur.

Concernant le point 3, on peut remarquer qu'une évolution actuelle de tous les outils de simulation de flux est de les rendre de plus en plus communicants. Cette évolution va tout à fait dans le sens de notre application et nous pourront en voir un exemple détaillé dans le Chapitre 4.

Concernant le point 4, nous pouvons simplement remarquer que le fait d'utiliser un même outil de simulation de flux pour réaliser l'observateur et le simulateur en ligne facilitera grandement la communication entre les deux et permettra une initialisation aisée du simulateur en ligne à partir de l'état du simulateur observateur.

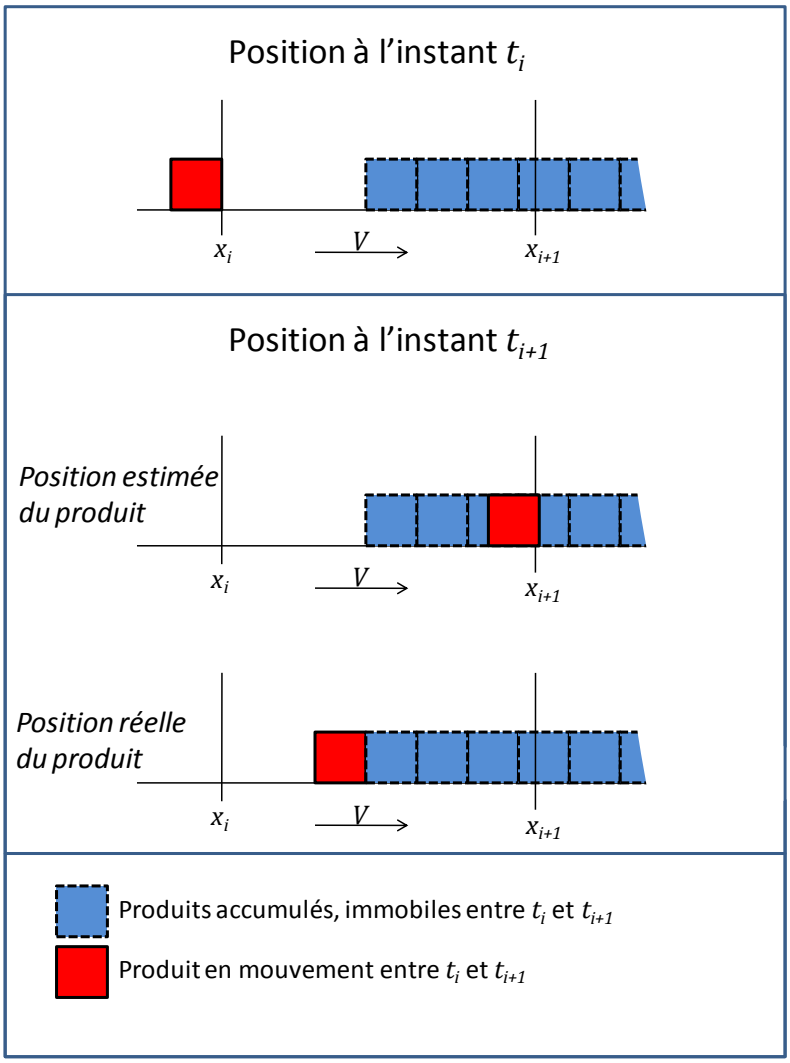

Figure 32 Prévisions sur un convoyeur à accumulation

III.3.4.2 Position de l'observateur dans l'architecture de commande

Nous avons présenté au début de ce document l'architecture que nous considérions, centrée autour du MES. La Figure 33 propose une vue de l'intégration du module d'aide à la décision comme partie intégrante du MES et de l'observateur en communication avec la commande du système et le MES. Les flèches représentées sur la figure correspondent aux communications directes possibles entre les éléments de l'architecture.

À celles-ci, il faut ajouter les communications internes aux éléments, comme à l'intérieur du MES par exemple, qui ne sont pas représentées par souci de clarté.

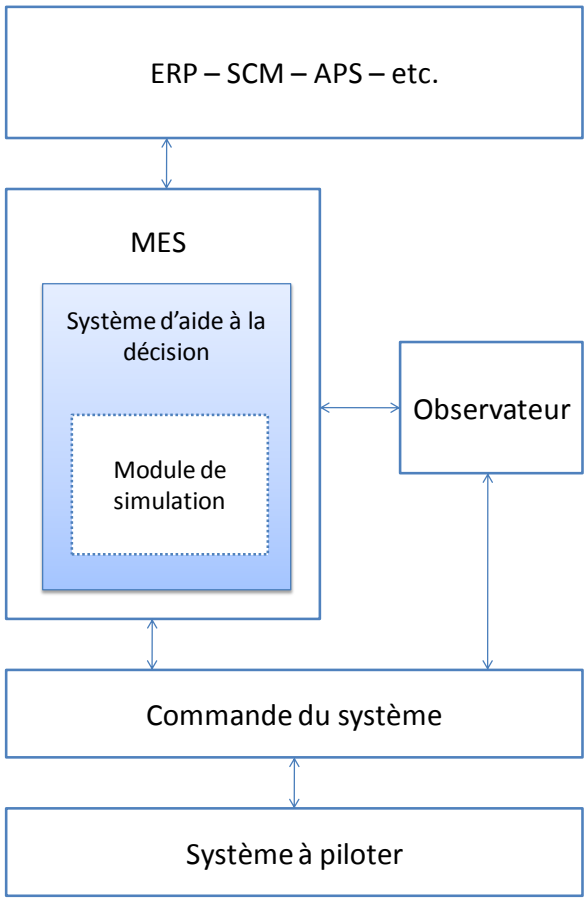

Figure 33 Intégration de l'observateur et du système d'aide à la décision dans l'architecture de pilotage du système de production

Nous avons choisi d'inclure la simulation en ligne à l'intérieur du MES comme aide à la décision pour les fonctions orientées pilotage : ordonnancement et cheminement des produits et des lots (voir paragraphe I.1.).

L'observateur se situe au même niveau CIM que le MES, c'est-à-dire au niveau de l'exécution. L'observateur ne remplit pas à proprement parler une des fonctions du MES. La seule relation nécessaire entre l'observateur et le MES est donc le lien entre le simulateur et l'observateur. Toutefois, plusieurs fonctions du MES sont nécessaires au bon fonctionnement de l'observateur. Deux choix sont alors possibles : les reconstruire dans l'observateur (parti pris dans la Figure 33), ou réutiliser celles du MES. La fonction principale dont nous parlons ici est la collecte et l'acquisition de données. D'autres peuvent être concernées, notamment la gestion des ressources. À terme, il est possible de supprimer la relation directe entre l'observateur et la commande du système, relation qui se fait par l'intermédiaire du MES.

III.4 Conclusion

Ce chapitre nous a permis de présenter trois alternatives à la collecte d'informations sur le système de production pour l'initialisation des simulations en ligne. Tout d'abord, la collecte directe depuis les capteurs, les détecteurs et la commande a montré sa performance dans la fiabilité des données, mais également des limites au niveau des

incertitudes nées de l'utilisation de technologies suffisantes pour la commande du système, mais insuffisante pour cette procédure.

Ensuite, l'utilisation d'un simulateur temps-réel tournant en parallèle du système de production, mais sans aucun lien direct avec celui-ci a permis de corriger ces problèmes d'incertitude, mais du caractère aléatoire de ce type de systèmes découle une divergence toujours plus importante du comportement du simulateur par rapport à celui du système.

La dernière partie de ce chapitre nous a permis de présenter le concept d'observateur réalisé par simulation, hybride des deux solutions présentées précédemment. Cet observateur est en constante relation avec le système de production, ce qui permet de recaler ses prévisions dès qu'une information lui parvient. Un exemple simple de mouvement alternatif de vérin nous a permis de présenter les mécanismes de recalage que nous proposons notamment de mettre en œuvre dans le prochain chapitre.

Chapitre IV.

Validation sur une plateforme expérimentale

Lors des trois chapitres précédents, nous avons vu d'une part quels sont les intérêts potentiels mais également les difficultés d'application de la simulation en ligne, et d'autre part les concepts que nous avons développé pour répondre à ces problèmes. Nous avons choisi, pour valider ces concepts, de les appliquer sur un système de production suffisamment proche d'une installation industrielle pour pouvoir y transposer directement les solutions développées.

Lors de ce chapitre, nous allons tout d'abord présenter ce système et les décisions à prendre au cours de son fonctionnement pour lesquelles nous proposons de tester l'utilisation de la simulation en ligne. Puis, nous allons étudier l'intégration de la simulation et de tous les éléments nécessaires à son bon fonctionnement dans l'architecture de commande du système. Enfin, nous présenterons les modèles de l'observateur et du simulateur réalisées sous Arena puis sous Quest.

IV.1 La ligne d'assemblage

Le système de production étudié (Figure 34) est situé au département Qualité, Logistique Industrielle et Organisation (QLIO) de l'Institut Universitaire de Technologie (IUT) de Nantes. Il a été conjointement financé par l'IUT et L'Atelier Inter-établissement de Productique (AIP PRIMECA) des Pays-de-Loire, puis développé au sein du département QLIO. Ce système, une ligne d'assemblage automatisée, fait partie d'un ensemble de production plus large, comprenant un magasin dynamique automatisé, et un ensemble de transport comportant deux chariots optoguidés. La finalité de cet ensemble est bien sûr la formation des étudiants de l'IUT mais aussi la recherche. L'application de cette thèse en est un exemple. Nous pouvons aussi citer des travaux concernant la comparaison d'ordonnancements prédictif, réactif et prédictif-réactif calculé en utilisant la théorie des ordonnancements de groupe [Pinot *et al.*, 2007].

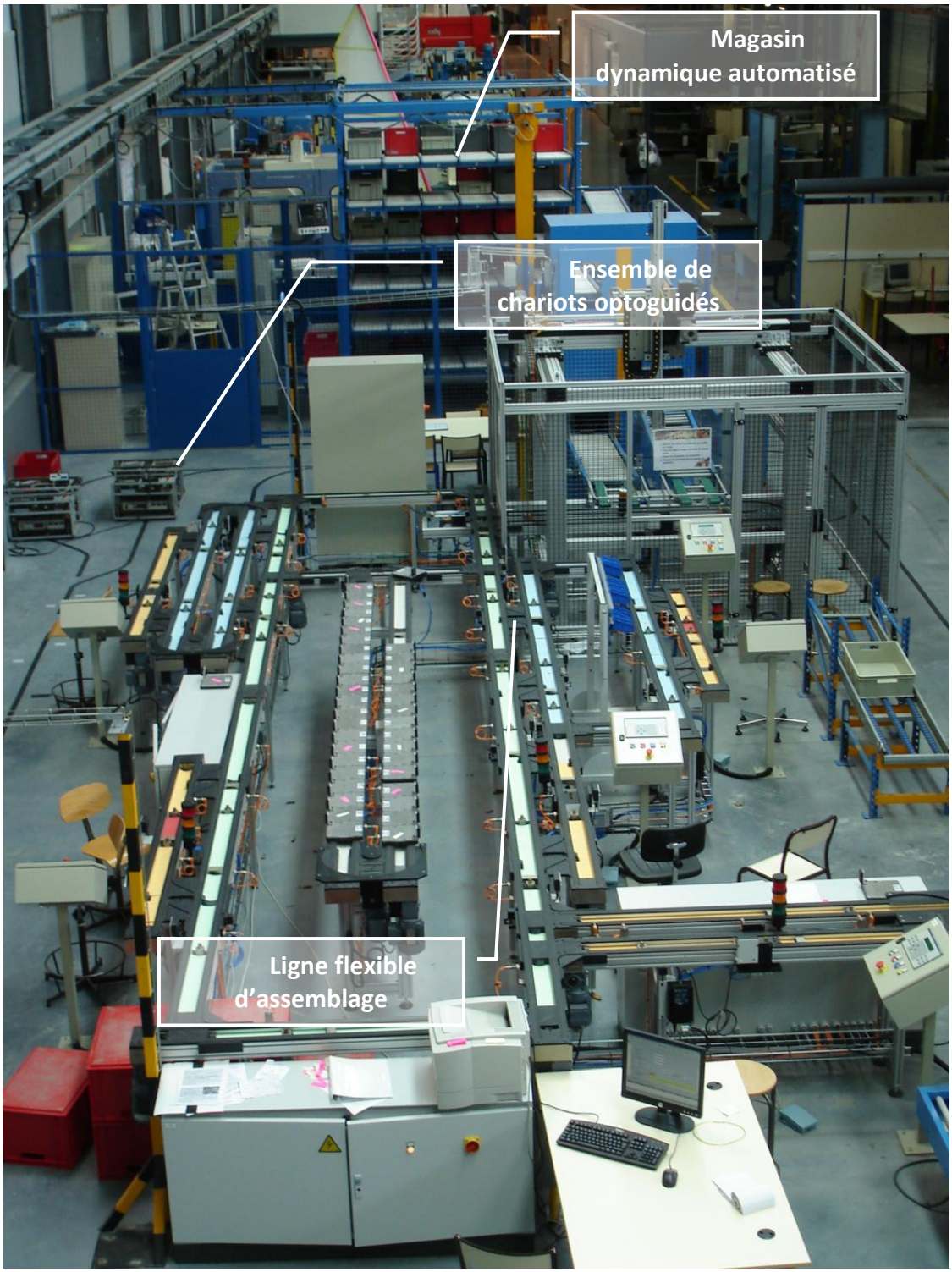

Figure 34 Le système de production en situation

IV.1.1 La structure physique

Ce système, déjà partiellement présenté au Chapitre II, est un système de production flexible à transferts automatisés. Les produits sont transportés par des palettes libres ; chaque palette ne peut transporter qu'un seul produit à la fois, et il y a toujours exactement

42 palettes sur le système. Le système peut être décomposé en trois classes d'éléments (Figure 35) :

1. Le magasin : il permet le stockage des palettes qui ne sont pas en production;
2. La boucle centrale de transfert : elle permet le transport des palettes entre les postes de travail ;
3. Les six postes de travail (machines) en dérivation : ils permettent la transformation des produits transportés par les palettes.

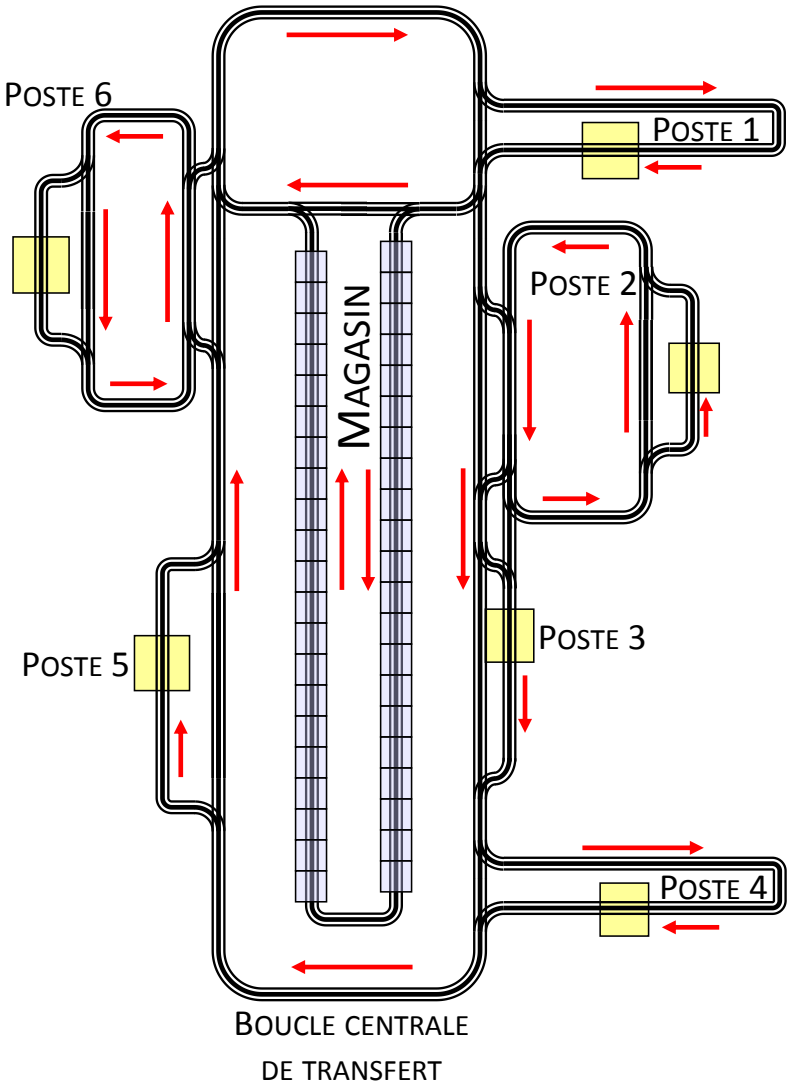

Figure 35 Disposition spatiale des éléments du système de production

Le transfert est assuré par des convoyeurs à ruban. Cette technologie impose que les palettes soient constamment en mouvement tant qu'elles ne sont pas bloquées par une butée ou par une palette elle-même bloquée. En effet, ces convoyeurs autorisent l'accumulation. Les butées sont actionnées par la commande du système.

Notons que la structure constituée d'un boucle centrale autour de laquelle sont situés les poste fait de ce système une unité de production de type job-shop. Cela signifie que les gammes, ou ordre de passage, sur les postes peuvent être absolument quelconques.

À l'initialisation du système et à chaque instant où aucune production n'est en cours, toutes les palettes sont vides et dans le magasin. Lors d'une production, elles circulent sur la boucle centrale d'entrée de poste en entrée de poste jusqu'à être dérivées dans celui qui leur permettra de poursuivre la production du produit qu'elles transportent. Les postes peuvent être classés en deux catégories :

1. Les postes FIFO : ces postes sont tous construits de manière identique, seule la taille de la zone de stockage change. La Figure 36 présente cette disposition. Son nom vient du stock amont au poste qui est de type FIFO (First In, First Out).

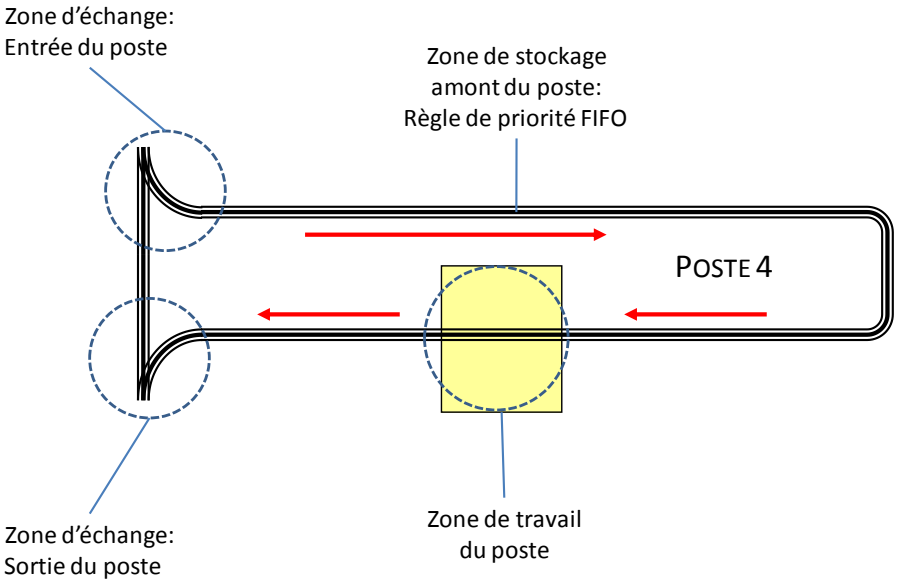

Figure 36 Un poste en dérivation de type FIFO

2. Les postes non-FIFO : ils se distinguent par un stock amont formé d'une boucle dont la règle de gestion peut-être quelconque. Un petit stock amont FIFO est de plus disposé à l'entrée immédiate du poste pour permettre d'augmenter la cadence (Figure 37).

À chaque poste est assignée par le gestionnaire de production une liste d'opérations qu'il est capable de réaliser. Cette liste n'est pas exclusive, ce qui implique que plusieurs postes peuvent être capables de réaliser la même opération. À chaque couple (opération ; poste), une durée opératoire est assignée. Chaque poste étant capable d'évoluer en manuel ou en automatique, c'est l'action de l'opérateur sur une pédale ou l'égalité entre le temps passé par la palette sur le poste et cette durée opératoire qui peut conditionner le départ de la palette du poste. De plus, à chaque couple (opération ; poste) est associée une durée de réglage. Cette durée est ajoutée à la durée opératoire si l'opération réalisée diffère de celle précédemment exécutée sur le poste.

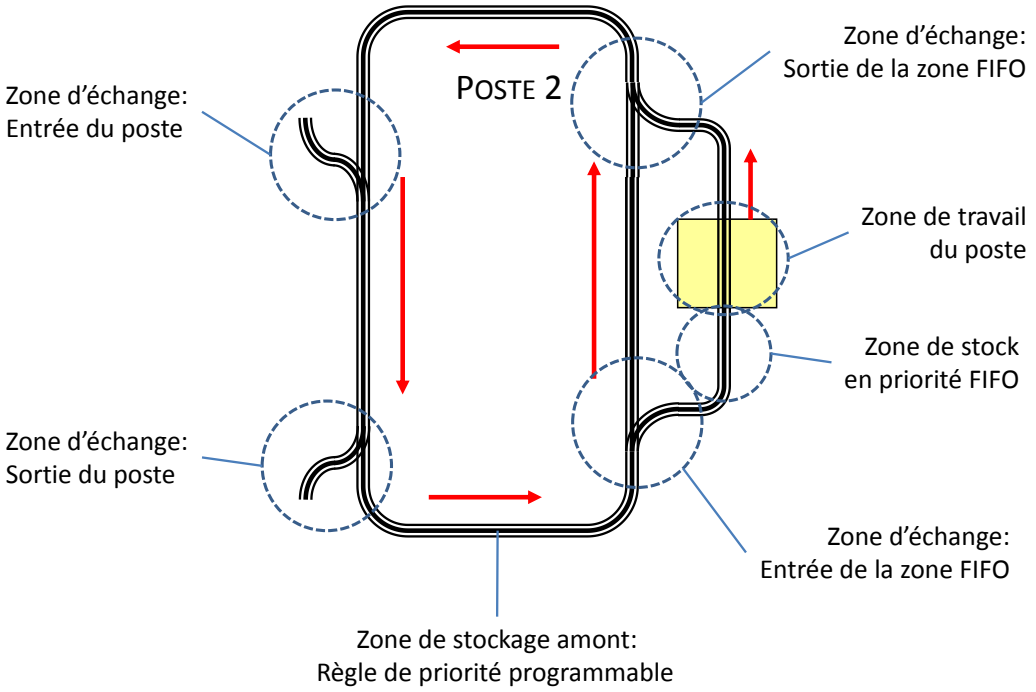

Figure 37 Un poste en dérivation de type non-FIFO

IV.1.2 La structure informationnelle

Le système est commandé par quatre automates programmables industriels disposés sur un réseau de terrain type Ethernet. Toutes les entrées et sorties classiques (capteurs, butées, systèmes de dérivation, etc.) sont reliés directement aux automates concernés. Des pupitres sont également disposés devant chaque poste de travail pour la communication directe entre un opérateur et les automates. Ceux-ci sont reliés aux automates par une liaison type série. Enfin, 18 modules de lecture/écriture d'étiquettes électroniques sont disposés sur un réseau FipIO dont font partie les automates.

Le Manufacturing Execution System est également disposé sur le réseau Ethernet. Nous avons construit un MES simple, puisqu'il ne nous semblait pas opportun de charger outre mesure l'architecture de commande avec des éléments non-indispensables à notre propos. De ce fait, le MES n'est constitué que d'une base de données Microsoft SQL Server (fonctionnant par requêtes SQL) et d'une application de supervision élaborée sur Wonderware Intouch. Depuis plus de 15 ans, InTouch est le logiciel de supervision de référence grâce, notamment, à sa légendaire simplicité et rapidité de mise en œuvre. À ce jour, plus de 325 000 installations à travers le monde sont exploitées avec InTouch, dont plus de 30 000 en France.

La base de données a pour fonction de stocker les informations (opérations réalisables par les postes, ordres de fabrication, etc.) et les résultats de production (temps de production de chaque produit, temps de production total de chaque ordre, etc.).

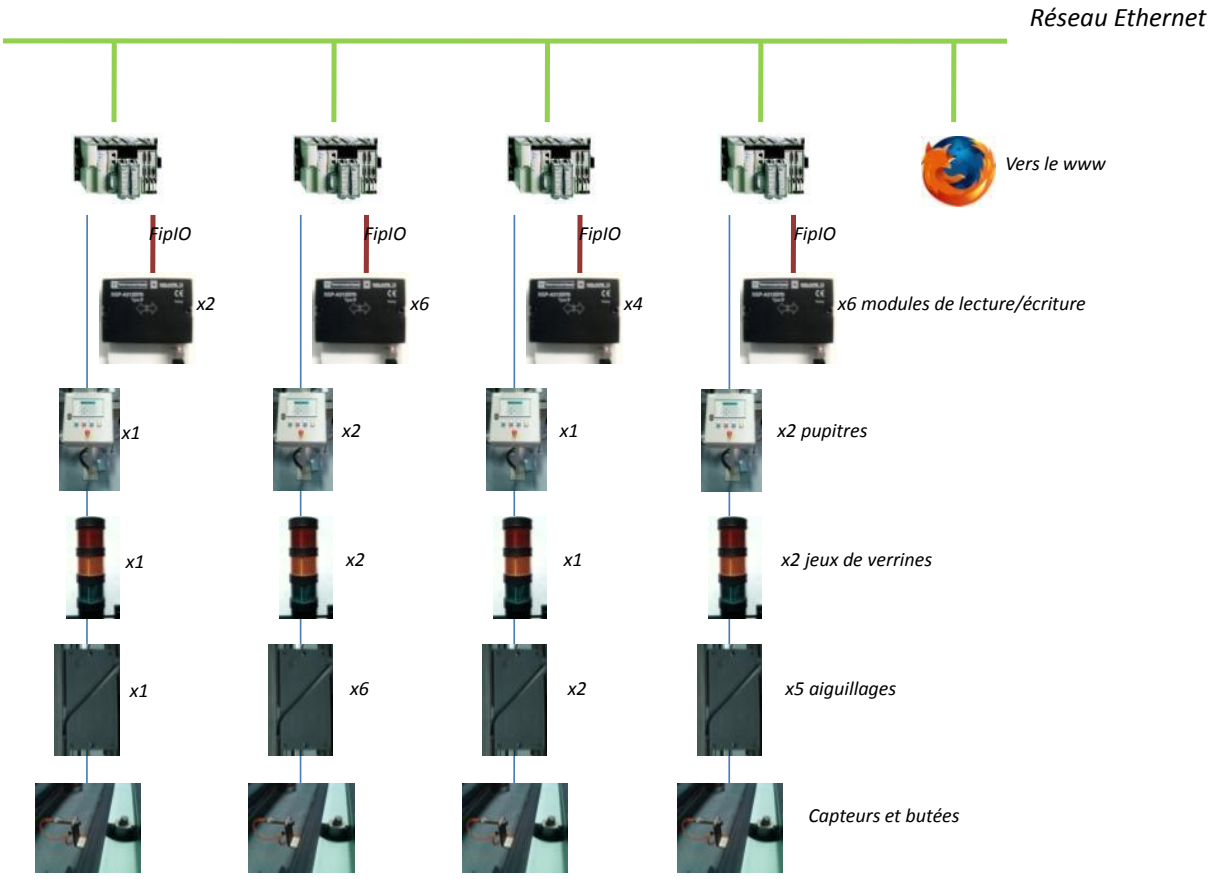

Figure 38 La commande bas-niveau du système

L'utilitaire de supervision a principalement une fonction d'Interface Homme-Machine destinée au gestionnaire de production (aussi appelé pilote de production). Il permet de configurer le système et les ordres de fabrication, de traiter les résultats de production, etc.

IV.1.3 Le fonctionnement

Nous avons largement utilisé, pour le fonctionnement de la ligne, le concept de pilotage par le produit (Chapitre I). Chaque palette est munie d'une étiquette électronique pouvant stocker 2 kilo-octets de données. Ces étiquettes permettent une décentralisation de l'information. En effet, elles sont utilisées pour embarquer les données nécessaires à la production du produit transporté. Les décisions concernant le devenir d'un produit sont prises après lecture des informations le concernant par les automates. De ce fait, les modules sont placés à tous les endroits impliquant une prise de décision : sortie du magasin pour le lancement des productions, points de dérivation de l'entrée des postes, zones de travail des postes. Les informations stockées dans les étiquettes sont de deux types. Les premières correspondent à l'ensemble des données nécessaires au fonctionnement du système, comme par exemple la gamme de fabrication ou l'ordre de fabrication. Les deuxièmes permettent d'assurer la traçabilité de la production réalisée. Ces dernières informations pourront, pour chaque production réalisée, permettre le calcul d'indicateurs de performances caractérisant cette production.

Lors du début d'une production, le module en sortie du magasin inscrit sur l'étiquette de chaque palette nécessaire à l'accomplissement de l'ordre de fabrication (OF) quelle est la gamme à réaliser et le nombre de produits à réaliser. Remarquons que les palettes transportent bien un seul produit à la fois mais elles peuvent parcourir plusieurs fois successivement le chemin correspondant à la gamme de fabrication et donc réaliser plusieurs produits, les uns après les autres. Les palettes sont ensuite libérées pour débuter la production. En sortie de magasin, les palettes circulent sur la boucle centrale de transfert. Le principe de fonctionnement est très simple. A l'entrée de chaque poste, un lecteur permet de lire un pointeur d'opération pour connaitre l'état d'avancement du produit se trouvant sur la palette. Puis, la gamme de production se trouvant aussi sur la palette, le lecteur permet de connaitre la prochaine opération à exécuter. Si l'opération à faire fait partie des opérations réalisables sur le poste, et si la règle de gestion du stock permet d'accepter cette nouvelle palette, alors la palette est dérivée vers la zone de stockage du poste. La palette passe ensuite sur la zone de travail du poste. Là, une autre unité de lecture/écriture va permettre, si l'opération s'est bien déroulée, d'incrémenter le pointeur d'opération. Ensuite, la palette retourne sur la boucle centrale de transfert pour rechercher le poste qui exécutera la prochaine opération. Remarquons que l'utilisation de ce pointeur autorise des gammes non linéaires avec par exemple des retours en arrière. Si une opération comporte une phase de contrôle, le contrôle peut conduire à ré-exécuter une opération pour par exemple retoucher un produit. La palette partant vide du magasin, la première opération consistera généralement à déposer sur la palette le premier composant du produit et la dernière opération consistera à démonter de la palette le produit terminé. Si la palette a terminé les produits qui lui ont été demandés, elle revient en fin de production au magasin.

IV.2 Quelques exemples de décisions à prendre par le pilote

Ce système de production à un comportement très hétérarchique. En effet, chaque poste de travail est très autonome. Il dialogue avec la palette qui se présente à l'entrée pour décider s'il accepte ou non de la prendre. Ce fonctionnement à l'avantage d'être totalement réactif. Nous avons pu voir précédemment que ceci a l'avantage d'éviter de lancer un calcul d'ordonnancement avant un lancement en production et d'être plus robuste aux aléas, mais a également l'inconvénient de rendre difficile l'obtention d'une vision précise sur le comportement du système. Cet inconvénient a une influence particulière sur les deux acteurs principaux du pilotage en temps-réel du système : l'opérateur humain, aussi appelé pilote, et la commande du système. Ceux-ci sont alors limités dans leur prise de décision.

Nous avons également pu voir (Chapitre II) que deux types de prises de décision sont principalement susceptibles d'accueillir la simulation en ligne : l'une excluant totalement l'opérateur humain de la boucle de décision, et l'autre le mettant au cœur de cette décision. Étant donnée la complexité des problèmes traités, nous avons vu dans le Chapitre II que l'automatisation complète était souvent limitée à une échelle plus restreinte de par l'exclusion de l'opérateur de la boucle de décision.

Nous nous sommes intéressés sur ce système à deux exemples de décision couvrant ces deux types :

1. Une décision menée par l'opérateur concernant le comportement global du système de production ;

2. Une décision totalement automatisée concernant le comportement local d'une petite partie du système.

IV.2.1 Lancement de production

Notre système fonctionne sur le principe d'Ordres de Fabrication (OF). Ces ordres sont caractérisés par :

- Une gamme de fabrication des produits ;
- Un nombre de produits à réaliser, décomposé en :
  - Un nombre de palettes allouées ;
  - Un nombre de produits à réaliser par palette ;
- Une date de lancement.

Il est très utile, pour le pilote de la ligne, de prévoir la date de fin estimée de l'ordre de fabrication en préparation. Cet indicateur est lié au paramétrage de l'ordre, et l'idée est qu'il serait intéressant de pouvoir paramétrer l'ordre en fonction de l'évolution de cet indicateur.

Toutefois, le comportement du système est difficilement modélisable par des méthodes autres que la simulation et l'état actuel de la production influe énormément sur le résultat final. Tout ceci rend le calcul précis de l'indicateur très difficile sans simulation en ligne.

Nous avons donc développé une application de calcul de la date de fin des ordres de fabrication selon leur paramétrage. Le paramétrage des ordres déjà lancés est évidemment fixé et ne peut être modifié, mais le paramétrage des ordres encore à lancer pourra changer en fonction des résultats obtenus. La perturbation déclenchant l'intervention de l'opérateur est donc ici l'ajout ou la modification d'un ordre dans la table des ordres à lancer.

Dans cette configuration, l'opérateur entre le paramétrage des simulations ; en fonction du résultat des simulations, il est libre de modifier ou confirmer son paramétrage sans intervention de la machine. On est donc dans le cas d'une *Aide à la décision* comme définie par [Endsley et Kaber, 1999].

Concrètement, nous avons dans la Figure 39a l'écran de supervision permettant le lancement des ordres. Nous voyons que les ordres 100 et 110 sont en cours. Lors du lancement de l'ordre 110, le pilote a lancé une première simulation en ligne qui a prévu la fin des ordres 100 à t=1960 secondes et 110 à t=2750 secondes. Le pilote désire lancer un nouvel ordre (120), devant être réalisé avant la date t=4000 secondes. Cet ordre doit permettre de fabriquer 20 produits correspondant à la gamme 3. Le pilote a décidé de lui affecter 10 palettes, réalisant donc chacune 2 produits. Après avoir renseigné les caractéristiques de son ordre, le pilote peut lancer la simulation par le bouton « Simulation en ligne ». Sur cette application, la durée de décision prise en compte dans la simulation est fixe et réglée à 20 secondes.

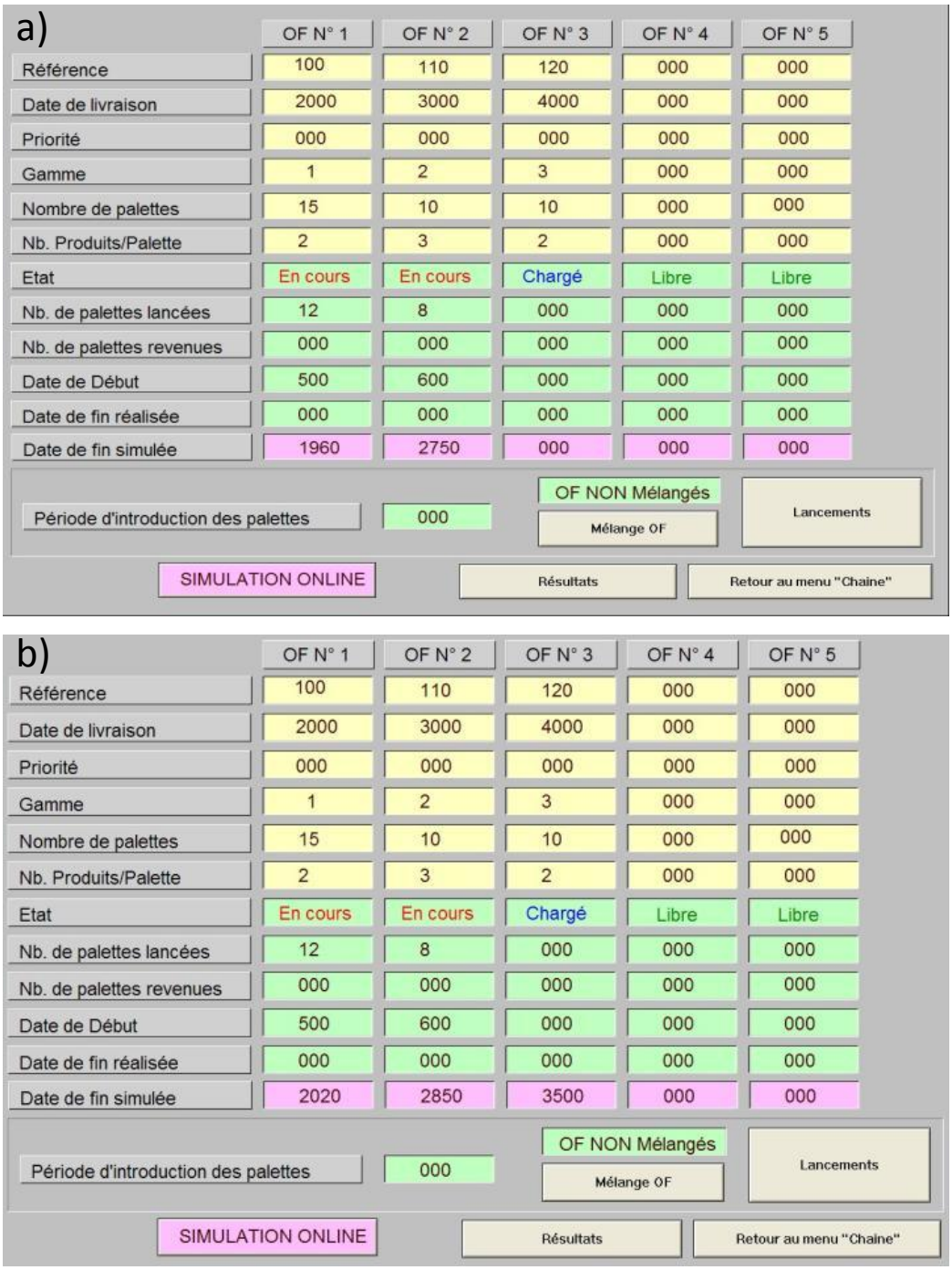

Figure 39 Le lancement de production vu du MES
a) Avant la simulation en ligne
b) Les résultats de la simulation en ligne

Il obtiendra alors les résultats de la Figure 39b. Nous voyons que le simulateur, en partant de la situation actuelle, a simulé la fin de réalisation des ordres 100 et 110 et la réalisation de l'ordre 120. Le pilote obtient, en fin de simulation, les dates de réalisation prévues de trois ordres. Il peut constater que le réglage qu'il a affecté à l'ordre 120 entraîne un léger dépassement de la date de réalisation de l'ordre 100. Le pilote peut alors modifier les données concernant le lancement de l'ordre 120 et relancer une simulation. Il peut aussi accepter ce résultat et lancer le nouvel ordre 120 en cliquant sur la touche « Lancement ».

Concernant les résultats obtenus, les ordres de fabrication qui arriveront ultérieurement ont bien sûr une influence sur le résultat, mais il est impossible de prendre cette donnée totalement inconnue en compte au moment du calcul. C'est pourquoi, à chaque fois qu'un nouvel ordre est entré par l'utilisateur, l'ensemble des dates de fin de tous les ordres sont réévaluées pour permettre de détecter l'influence du nouvel ordre sur le comportement de ceux préalablement entrés.

IV.2.2 Règle locale d'ordonnancement

Dans la structure du système étudié, deux postes nous permettent de faire un choix dynamique des produits à traiter : le poste de travail choisit dans un stock tournant. Le problème est que ce choix doit être fait très rapidement, car la dynamique du système créerait des engorgements et donc des ralentissements néfastes à la production.

Ainsi, nous avons développé sur ces postes plusieurs règles dynamiques de fonctionnement. On s'intéressera plus particulièrement ici à la règle qui a été définie par [Kumar et Seidman, 1990] comme suit :

Considérons un système de production avec les caractéristiques suivantes :

- Il y a P types d'éléments numérotés $p = 1,2, \dots, P$ et M machines numérotées $m = 1,2, \dots, M$.
- Les éléments de type p entrent dans le système à un taux de d_p éléments par unité de temps dans une machine $\mu_{p,1}$. Ils visitent ensuite les machines $\mu_{p,2}$, $\mu_{p,3}$, …, μ_{p,n_p} dans l'ordre avec chaque $\mu_{p,i} \in \{1,2, \dots, M\}$, et sortent du système depuis la machine μ_{p,n_p} comme produit fini. Il est possible qu'une pièce visite une machine donnée plus d'une fois ; ceci arrive lorsque $\mu_{p,i} = \mu_{p,j}$ avec $i \neq j$.
- À la i[ème] machine $\mu_{p,i}$ visitée par les éléments de type p, un temps opératoire $\tau_{p,i}$ doit être respecté. En attendant d'être traités par la machine, ils attendent dans un stock $b_{p,i}$.
- Les stocks $B_m := \{b_{p,i} : \mu_{p,i} = m\}$ desservent la machine m. Lorsqu'une machine m passe du traitement des éléments du stock $b \in B_m$ au traitement des éléments du stock $b' \in B_m$, un temps de réglage $\delta_{b,b'}$ doit être respecté.

Notons $x_{p,i}(t)$ le nombre d'éléments p dans le stock $b_{p,i}$ au temps t,

Une politique d'ordonnancement sera appelée *politique de vidage* si elle se comporte ainsi : dès qu'une machine m traite des éléments dans un stock $b \in B_m$, alors elle reste réglée pour ce type d'élément et continue de traiter les éléments de b jusqu'à ce que le stock b soit vide et qu'il existe simultanément au minimum un stock $b' \in B_m$ non-vide. À cet instant, la machine commence alors le réglage pour un des stocks non-vides de B_m.

Une sous-classe spéciale des politiques de vidage est la classe de la politique *clear-a-fraction* (CaF) définie par : une politique de vidage sera appelée une politique *clear-a-fraction*, si pour chaque machine m il existe une constante $\varepsilon_m > 0$, et une constante K_m telles que si la machine m commence un réglage pour le stock $b_{p,i} \in B_m$ au temps t, alors

$$x_{p,i}(t) \geq \varepsilon_m \sum_{\{(q,j):\mu_{q,j}=m\}} x_{q,j}(t) - K_m$$

Il est à noter que l'on peut aisément construire une politique CaF qui possède également les 2 propriétés suivantes :

- Il est possible de l'implanter de manière distribuée sur les machines, sans avoir besoin d'un quelconque échange d'informations ou de coordination d'actions entre les machines ;
- Il est facile de l'implanter en temps-réel sans nécessiter un quelconque calcul pour déterminer quand changer de réglage, et pour quel stock.

Cette politique a la particularité de moduler de façon réactive la taille des lots à passer sur chaque machine. Ainsi, lorsque la machine est peu chargée, elle a tendance à régler très souvent, alors que lorsqu'elle est très chargée, elle a tendance à régler très peu souvent. De ce fait, elle a pour caractéristique de diminuer le nombre de réglages réalisés lors d'une production, tout en obtenant un temps de production total très bon en comparaison de règles telles FIFO ou LIFO.

La question que nous nous sommes posés est de savoir si la simulation ne pouvait pas améliorer la règle dans le cas de petits ordres de fabrication. En effet, cette règle est très bien adaptée lorsqu'il s'agit de grandes productions. Toutefois, lorsque l'on arrive en fin de production d'une référence, il est tout-à-fait envisageable d'avoir les derniers produits très retardés par le passage d'une autre référence avant eux.

Prenons l'exemple de la Figure 40, qui est tiré de [Cardin et Castagna, 2006a]. Le poste 6 est amené à traiter deux types de palettes repérées 1 et 3. Nous imaginons que le passage du traitement d'une palette 1 à 3 ou de 3 à 1 s'accompagne d'une durée de réglage relativement longue. La production actuelle concerne des palettes 1. Or, en application de la règle *Clear a Fraction*, la palette 3 se trouvant en A a été dérivée vers la zone de travail puisqu'il n'y a plus de palette 1 dans le stock. On va donc provoquer un réglage pour ensuite traiter des palettes 3. Or, nous voyons en B que juste à l'entrée du poste se trouve une palette 1, qui devra attendre le traitement de toutes les palettes 3 et un nouveau réglage avant d'être traitée à son tour. Il aurait été sans doute plus astucieux d'attendre le traitement de cette palette 1 avant de provoquer un réglage.

Notre idée est donc d'étendre la vision de la règle à l'ensemble du système et non au seul stock amont de la machine par la simulation. Ainsi, la règle pourrait prendre en compte les produits étant proches d'entrer dans le stock, et ainsi les faire passer avant de changer de réglage.

Du fait de la structure des postes considérés de notre système de production, le choix de la référence à produire se fait différemment du cas général. À chaque fois qu'un produit passe au point A, la commande évalue s'il est judicieux de laisser entrer le produit sur la machine. Lorsque la règle CaF est appliquée, la décision peut être écrite de la forme :

Le produit entre si :

1. *La machine est disponible ;*
2. *L'opération à réaliser est la même que celle pour lequel la machine est réglée ou si aucun produit du stock ne remplit cette condition.*

Dans notre cas, nous avons modifié le deuxième point et déclaré la règle :

Le produit entre si :

1. *La machine est disponible ;*
2. *L'opération à réaliser est la même que celle pour lequel la machine est réglée ou si aucun produit actuellement dans le stock ou rentrant avant un délai donné dans le stock ne remplit cette condition.*

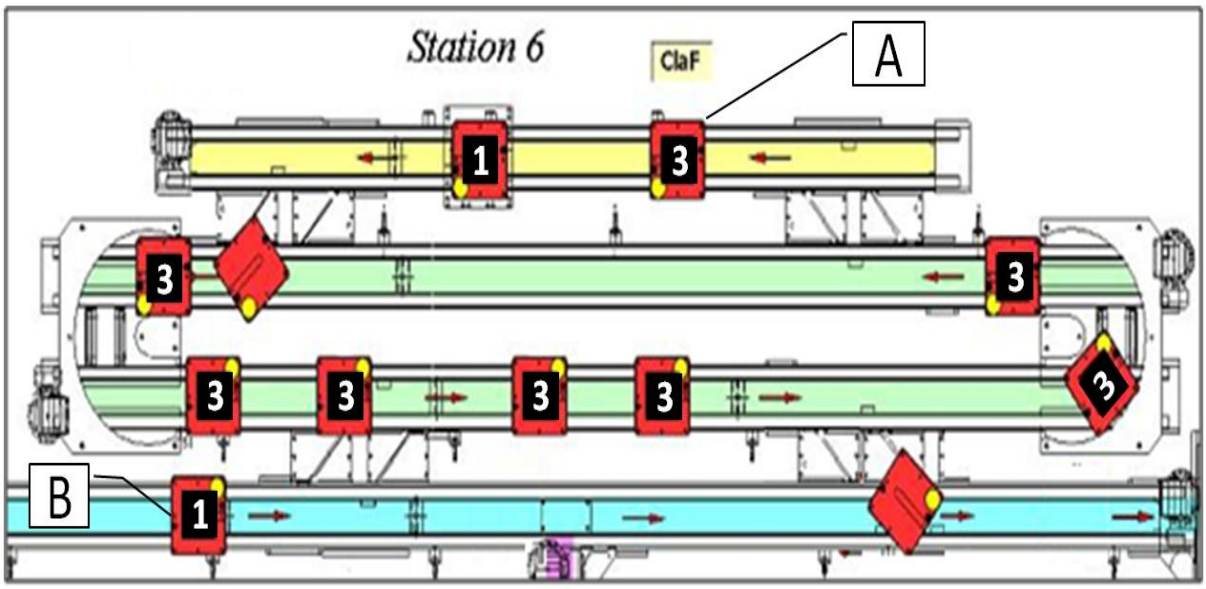

Figure 40 Le poste 6 en production

Il est clair que, plus le nombre de produits attendus sera grand, plus le nombre de réglage diminuera. Il est également évident que l'augmentation du délai laissé aux produits pour se présenter à l'entrée du poste aura pour effet l'augmentation du nombre de produits attendus. Notre étude [Cardin et Castagna, 2006b] a porté sur l'évaluation du délai maximal à laisser aux produits pour rentrer dans le stock pour ne pas ralentir la production. Pour cela, nous avons lancé des ordres de fabrication successifs et avons analysé les résultats en termes de nombre de réglages et de temps de production. Sur cet exemple particulier, nous avons obtenu une chute d'un tiers du nombre de réglages, tout en gardant un makespan globalement constant (légèrement inférieur). Ces résultats ont été obtenus pour un horizon de simulation d'environ 150 à 200 secondes.

La simulation en ligne a été insérée comme suit dans la commande :

1. Lorsqu'un produit se présente à l'entrée de la machine, la commande évalue s'il peut rentrer sur le poste ;
2. S'il est autorisé à entrer, mais qu'il génère alors un nouveau réglage, alors la commande envoie une demande de confirmation au MES et bloque le produit ;
3. Le MES ordonne au simulateur une simulation démarrant à la date actuelle avec un horizon fixé au préalable correspondant au délai laissé aux produits pour entrer dans le stock ;

4. Le simulateur demande l'état actuel de l'atelier à l'observateur ;
5. L'observateur stocke son état dans des fichiers de données et renvoie un acquittement au simulateur ;
6. Le simulateur s'initialise à partir des données des fichiers et démarre la simulation ;
7. Pendant la simulation, la première entrée d'un produit de la référence pour laquelle la machine est réglée est détectée ;
8. À la fin de la simulation, la simulation envoie sa réponse au MES (un produit est arrivé ou non) ;
9. Le MES transmet la décision à la commande ;
10. La commande applique la décision et débloque le produit.

Pour permettre de réaliser les tests en un temps raisonnable, nous avons substitué à la commande un modèle de simulation représentant le système de production émulé.

Contrairement à précédemment, l'opérateur n'entre en rien dans la mise en œuvre de la décision. Les 10 étapes se déroulent donc de manière totalement automatisée, ce qui est caractéristique du niveau d'automatisation *Automatisation Complète* défini par [Endsley et Kaber, 1999]. De manière générale, on réservera l'automatisation complète aux décisions où la définition des alternatives est très bien maitrisée et où la décision se fait sur un indicateur de performance très bien défini. Dans notre cas, le temps de réponse nécessaire est un critère supplémentaire à l'automatisation de la décision alors qu'il n'est pas obligatoire dans le cas général. Notre exemple nous a permis d'évaluer une durée globale inférieure à 5 secondes pour l'obtention de la réponse. Comparativement aux valeurs des durées en jeu dans ce système, et aux horizons de simulation trouvés, celle-ci semble largement acceptable.

IV.3 Intégration dans l'architecture de commande

Nous allons désormais aborder l'intégration physique de notre module d'aide à la décision dans l'architecture de commande du système. Nous avons vu au chapitre précédent que celui-ci se situait au même niveau CIM que le MES, c'est-à-dire au niveau de l'exécution. Nous allons détailler ici les solutions techniques qui ont été mises en place pour assurer les communications entre les différents éléments de l'architecture.

La Figure 41 présente un diagramme de séquence UML représentant les communications entre les éléments de l'architecture de commande nécessaires à une simulation en ligne. Sur ce diagramme sont représentés chronologiquement les différents messages échangés par les acteurs de cette simulation. Le MES initie la simulation en en référant au simulateur. Celui-ci demande à l'observateur de lui fournir son état, tout en interrogeant la base de données du MES pour obtenir les données de production actuelles du système. Lorsque l'observateur acquitte la création des fichiers contenant son état, le simulateur peut aller les consulter. Lorsqu'il a toutes les informations à disposition, le simulateur peut débuter les réplications qu'il a à effectuer. Lorsque toutes les simulations sont terminées, le simulateur communique les résultats au MES.

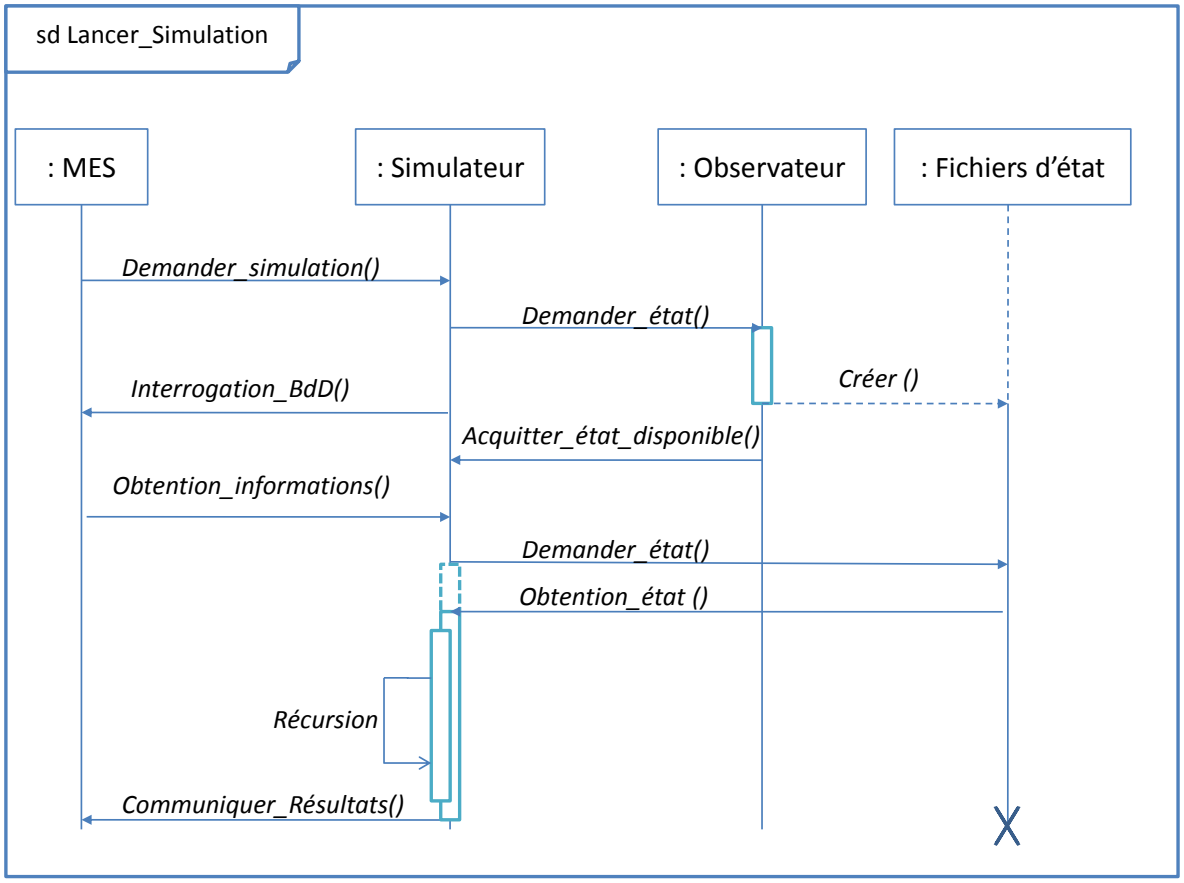

Figure 41 Diagramme de séquence UML représentant les communications entre les éléments de l'architecture de commande nécessaires à une simulation en ligne

IV.3.1 Le simulateur

Le simulateur en tant que tel est finalement partie intégrante du MES, comme peut l'être la base de donnée par exemple. De ce fait, il est préférable d'utiliser les moyens de communication déjà existants au sein du MES. Le simulateur doit être capable de communiquer à trois niveaux au sein de l'architecture :

- Une communication avec la base de données, qui se fera entièrement par requêtes SQL ;
- Une communication par fichiers communs avec l'observateur, détaillée dans le prochain paragraphe ;
- Une communication avec les autres composantes de l'architecture sous forme de questions/réponses/acquittement pour permettre de déclencher simplement les calculs et autres routines.

Pour cette dernière communication, nous avons majoritairement utilisé les Sockets Windows, globalement acceptées par la plupart des logiciels du commerce (MES et simulation). La Socket Windows, ou plus couramment appelé Winsock, est une interface d'application procurant des fonctions d'accès aux protocoles réseaux, permettant donc l'utilisation du protocole TCP/IP sur une interface Windows. Winsock 2.0 fournit deux interfaces de programmation. La première, qui est celle qui nous intéresse, est appelé API

(Application Programming Interface) par opposition à l'interface SPI (Service Provider Interface). Elle se situe en couche 5 du modèle OSI (Figure 42) et utilise donc un protocole de couche 3 et 4 pour véhiculer ses informations à travers un réseau.

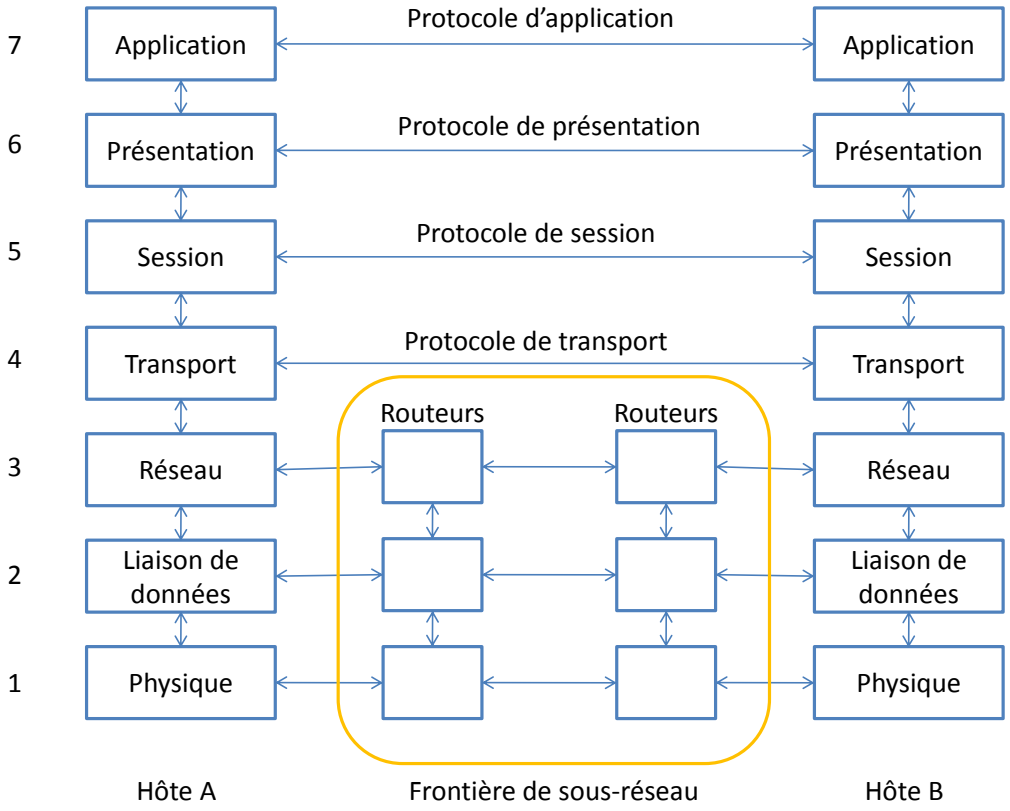

Figure 42 Le modèle OSI dans une configuration à deux hôtes du même sous-réseau

IV.3.2 L'observateur

L'observateur a deux caractéristiques : être synchronisé avec le système réel et communiquer son état à l'application de simulation. Il aura donc deux communications à établir.

Pour la communication avec le simulateur, nous proposons une communication double : une socket TCP/IP permet de transmettre la demande du simulateur, et l'acquittement de l'observateur, alors qu'une communication par fichiers communs permettra de faire transiter les données sur l'état, qui représentent une masse plus importante. Nous avons choisi ici de ne traiter que de simples fichiers type texte du fait de son universalité (la majorité des logiciels savent lire et écrire dans des fichiers type texte) et de la structure simple des données à transmettre. Si la masse de données se complexifie et qu'un traitement plus approfondi de ces données est nécessaire, il pourrait être préférable d'utiliser des structures type XML ou un transit via la base de données du MES.

Pour la communication avec le système réel, nous proposons d'utiliser le standard OPC. La norme OPC, qui signifie OLE for Process Control, est une norme industrielle qui fournit une interface commune de communication permettant à des logiciels indépendants de partager et d'échanger des données. Cette norme est supportée par une fondation,

soutenue par de nombreux grands acteurs industriels[2]. L'objectif d'OPC est de créer une véritable interopérabilité entre différents équipements industriels issus de plusieurs fournisseurs différents sur un réseau Windows [Santos *et al.*, 2005].

La communication sous OPC est réalisée avec une architecture type client/serveur, notre serveur utilisant les communications Microsoft DCOM. Le serveur OPC est la source des données et n'importe quelle application basée sur OPC peut y accéder pour lire/écrire toute variable fournie par le serveur (Figure 43). Ceci constitue une solution simple, ouverte et efficace au classique problème de compatibilité entre drivers propriétaires, puisque presque tous les principaux constructeurs mondiaux de systèmes de commande, d'instrumentation et de systèmes de supervision incluent OPC dans leurs produits.

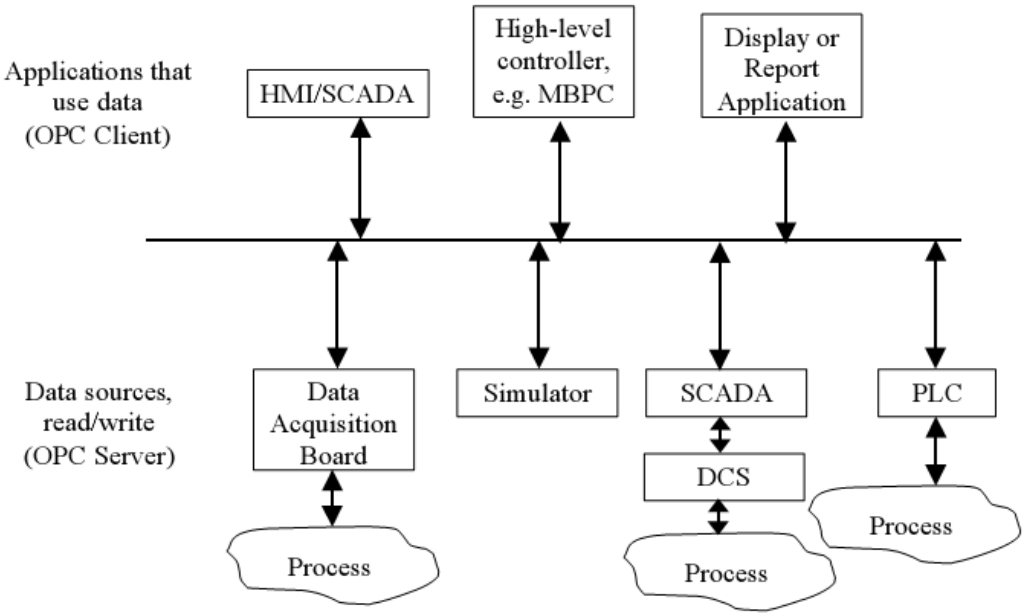

Figure 43 OPC facilite la communication entre les équipements bas niveau, les bases de données et les applications dédiées [Zamarreño *et al.*, 2000]

Plusieurs outils existent permettant de programmer un client OPC. La plupart sont basés sur l'utilisation de bibliothèques de liens dynamiques (dll pour Dynamic Link Library) programmées en C++, généralement fournies avec le serveur. Nous avons choisi comme serveur *OPC Factory Server*, logiciel propriétaire de Schneider Electric nous permettant une configuration très rapide de la communication avec les automates de la même marque.

L'observateur a été réalisé à partir de deux logiciels différents. L'un d'entre eux, Arena, permet une intégration immédiate d'un client OPC via son interface Visual Basic for Applications (VBA). Les dll sont directement intégrées et la programmation par évènement est très bien adaptée à notre problème.

Le second logiciel, Quest, a une interface de communication basée sur la lecture/écriture dans des fichiers ou par socket TCP/IP. Il n'est donc pas possible de directement intégrer les évènements OPC à la programmation. La solution que nous avons

[2] OPC Foundation: http://www.opcfoundation.org/

élaborée a été de construire un logiciel intermédiaire en Visual Basic (VB) communiquant classiquement avec le serveur OPC et fournissant les informations au simulateur via des sockets TCP/IP.

Que ce soit avec un logiciel ou avec l'autre, notre client est donc toujours programmé en VB (ou VBA), seule l'interface entre cet intermédiaire et l'observateur diffère un peu selon que le langage est directement intégré au logiciel ou non (Figure 44).

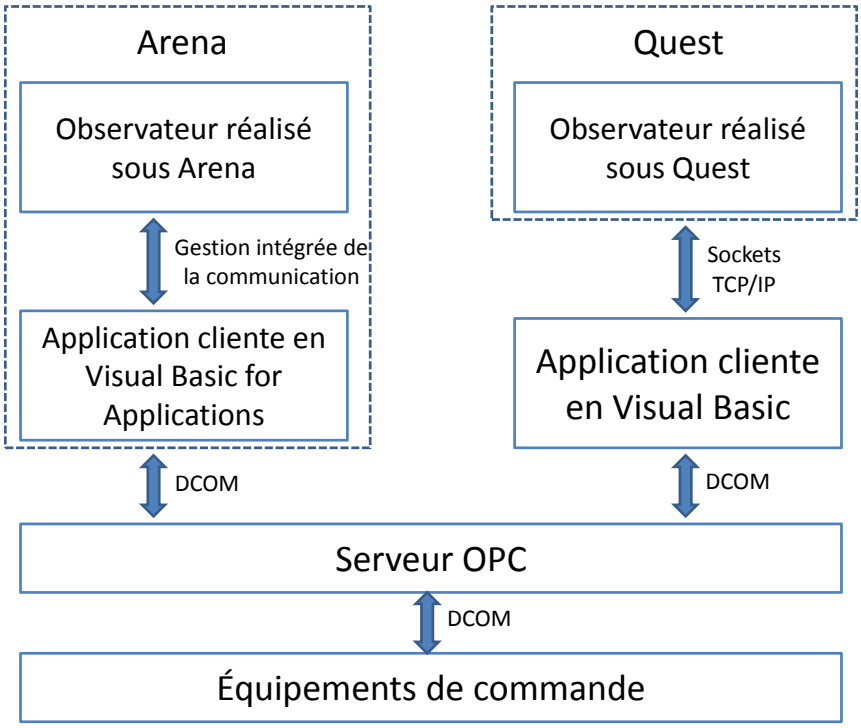

Figure 44 Différences d'intégration des contenus OPC pour deux observateurs réalisés sous Arena et sous Quest

IV.3.3 Performance de la remontée d'informations depuis l'atelier

Nous avons cherché à savoir quelles étaient les performances temporelles que nous pouvions attendre d'une telle architecture de remontée d'informations basée autour d'un serveur OPC. La question était de connaitre le temps séparant un évènement sur le système réel de sa perception par l'application cliente du serveur OPC, programmée en Visual Basic ou en Visual Basic for Applications, selon la configuration du serveur OPC. Un des paramètres de fonctionnement du serveur OPC qu'il nous a semblé primordial d'étudier était sa période de rafraichissements des données. Ce paramètre contrôle la période à laquelle le serveur OPC interroge les équipements de commande pour connaître l'état des variables considérées. Si l'état d'une ou de plusieurs variables a changé depuis le dernier rafraichissement, le serveur envoie un évènement aux applications clientes de ces variables. Ce paramètre peut être réglé à un minimum de 50 ms et sans maximum. Ce minimum implique que les variations de certaines données ne pourront pas être détectées par le serveur OPC. Par exemple, les variables intermédiaires des automates peuvent évoluer à chaque cycle de l'automate. Dans notre application, le temps de cycle moyen est d'environ 3 ms, ce qui empêche toute détection du serveur OPC.

Au vu de la dynamique de notre système, nous avons limité notre étude à un temps de rafraichissement maximal de 1000 ms. Remarquons ici que la vitesse de déplacement des palettes est de 20 cm/s et qu'elles s'inscrivent dans un carré de 20 cm de coté. Ainsi, un capteur détectant la présence d'une palette ne voit celle-ci que durant 1000 ms. Un montage expérimental a été construit pour nous permettre de mesurer précisément le temps de remontée de l'information. Celui-ci est présenté Figure 45.

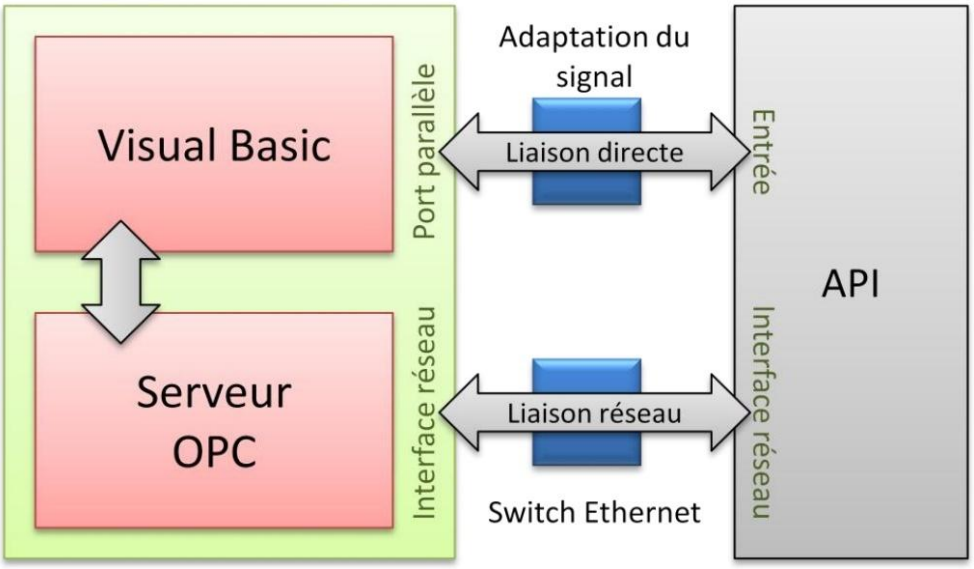

Figure 45 Montage expérimental [Chové, 2007]

L'ensemble des mesures est géré par le programme Visual Basic. Le principe de celui-ci est séquentiel : tout d'abord il génère un front électrique sur l'une des entrées de l'automate en déclenchant une mesure de temps. Pour cela, nous avons relié le port parallèle de l'ordinateur de test à une entrée de l'automate au travers d'un montage d'adaptation de tension. Nous considérons la durée de transmission de ce signal négligeable comparée aux temps de rafraichissement testés. De même, le temps de prise en compte par l'automate de cet évènement est négligé, étant compris entre 0 et 3 ms. Lorsque le serveur OPC détecte cette variation, il en informe l'application cliente VB, qui stoppe alors la mesure. Le temps qui s'est écoulé est alors considéré comme le temps de remontée de l'information.

Le serveur OPC et l'application cliente sont placés sur le même ordinateur pour ne pas avoir à prendre en considération de temps de communication Ethernet entre ces deux applications. Toutefois, étant donné que plusieurs données stochastiques entrent en ligne de compte (temps de remontée au travers du switch Ethernet, position de l'évènement à l'intérieur de la période de rafraichissement, etc.), nous avons réalisé 1000 essais pour chaque période de rafraichissement entre 50 et 1000 ms, avec un incrément de 50 ms. La moyenne des temps de réponse est présentée Figure 46.

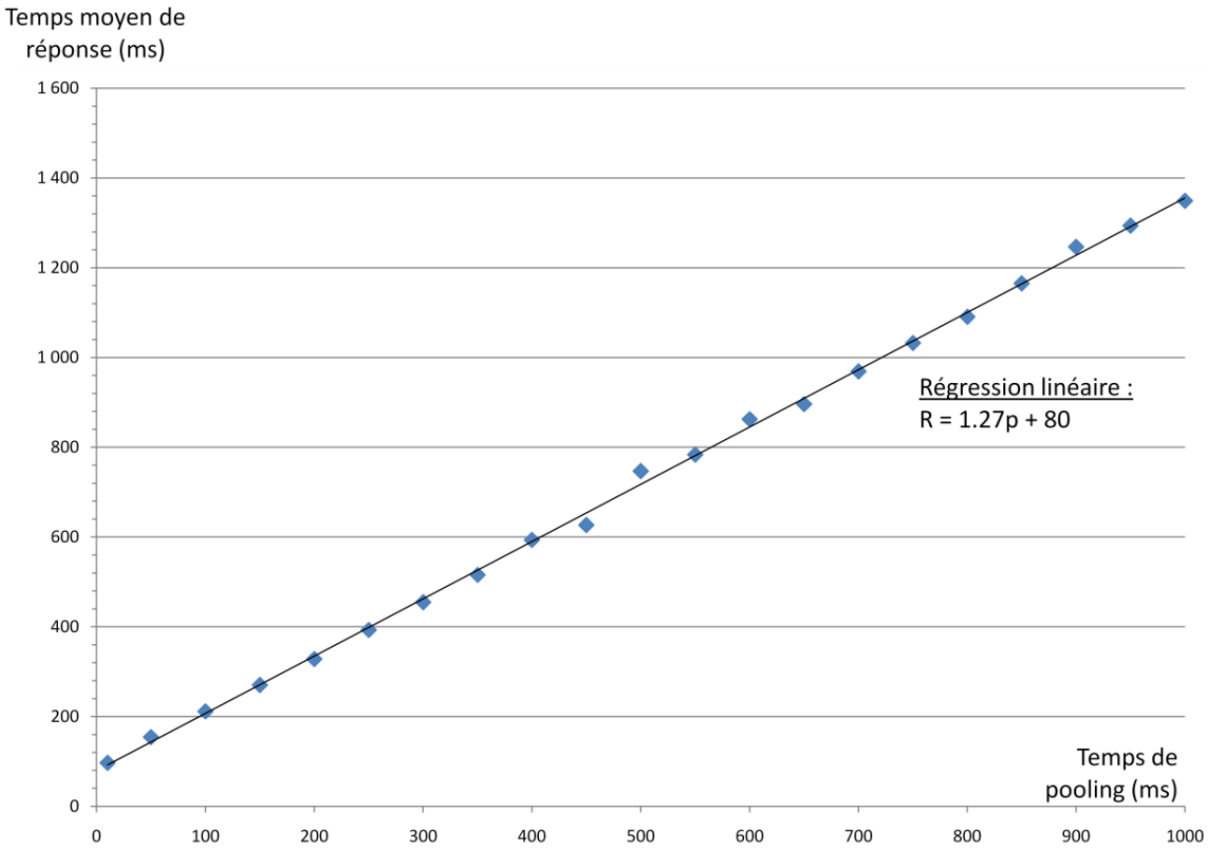

Figure 46 Moyenne des temps de réponse OPC [Chové, 2007]

La courbe obtenue peut être approximée par une droite d'équation :

$$R = 1.27p + 80$$

avec R le temps de remontée de l'information en millisecondes et p la valeur du réglage de la période de rafraichissement du serveur OPC. Nous avons de plus observé sur ces essais un écart type de l'ordre de 0.5 p.

La première indication remarquable est l'ordonnée à l'origine. Ces 80 ms incompressibles sont probablement dues aux différents retards pris par l'information : temps de communication entre l'application et l'automate, temps de cycle automate, temps de communication Ethernet, etc. Cette donnée impose une limite physique à l'utilisation du serveur OPC dans une telle configuration : les systèmes de production haute cadence où deux évènements concernant la même variable peuvent survenir en moins de 80 ms ne peuvent supporter l'utilisation d'un tel serveur.

Nous n'avons pas encore d'explication précise sur la valeur du coefficient directeur de la droite. Alors que nous nous attendions à un coefficient de 1, cette valeur de 1.27 semble indiquer une différence entre le réglage du paramètre sur le serveur et sa valeur réellement utilisée. Toutefois, aucun test réalisé n'a pu venir étayer cette supposition.

Bien évidemment, une faible valeur de la période de rafraichissement implique un temps de remontée de l'information faible. Pour notre application, nous avons retenu un période de rafraichissement de 50ms. Cela correspond à un temps de réponse en moyenne

de 150 ms. Rapporté à la dynamique du système, ce retard implique déjà une imprécision de 3 cm environ sur la position des palettes en mouvement.

Cette relativement faible valeur de paramètre implique également un nombre important de communications entre le serveur et les automates, ce qui augmente considérablement la charge du réseau. Notre application tourne correctement avec une période de 50 ms pour une centaine de variables observées par OPC sur quatre automates, mais il n'est pas évident que ce soit le cas de toutes les applications, notamment celles utilisant déjà massivement les communications entre équipements de commande.

Le réglage de ce paramètre est primordial dans la mise en place du serveur OPC. Trop élevé, le risque de rater des évènements est trop important et la précision de l'observateur décroit linéairement. Trop faible, il risque de surcharger inutilement le réseau et d'éventuellement impliquer des erreurs de communication dues à une latence trop importante. Nous reparlerons de ces problèmes de charge réseau au paragraphe IV.6.

IV.4 Modélisation de l'observateur

Après avoir vu l'environnement dans lequel se situent l'observateur et le simulateur, nous proposons de détailler les deux modélisations en jeu, forcément très différentes. Tout d'abord, intéressons-nous à l'observateur.

Au démarrage de l'observateur, la première tâche à effectuer est l'initialisation de la simulation pour que tous les éléments soient en place, en attente d'évènements extérieurs. Cette initialisation se fait sur un état connu de notre système. En l'occurrence, celui-ci se vidant à la fin des productions, toutes les palettes étant alors rangées dans le magasin, nous avons choisi cet état. Seul l'ordre des palettes dans le magasin est inconnu, information que nous récupérons du système réel au moment de la sortie des palettes. Pour information, cette phase d'initialisation est déclenchée par l'évènement intitulé *RunBeginSimulation* dans Arena, qui survient juste avant la compilation du modèle. À la fin de cette phase, il est possible d'agir sur les entités, et donc sur le fonctionnement de l'observateur.

Il est ensuite nécessaire d'aller chercher l'ensemble des données nécessaires au paramétrage de l'observateur dans la base de données du MES. Le principe que nous avons retenu est de lire chaque donnée une à une et, chaque fois qu'il est possible, de les identifier directement à des variables globales du modèle de simulation.

Lorsque ces données sont intégrées, l'étape suivante est l'établissement de la connexion à toutes les variables des automates nécessaires dans l'application cliente pour la communication avec OPC, qui sont appelés *items* sous OPC. À partir de cet instant, tous les changements de valeur de l'un de ces items déclencheront une routine de recalage dans l'observateur. C'est la raison pour laquelle la connexion doit être réalisée en dernier : les routines ne peuvent en général s'appliquer que sur un simulateur totalement opérationnel.

Il n'y a pas, à ce stade, de différence majeure de modélisation entre les observateurs réalisés due à la différence de logiciel. Nous allons voir dans les paragraphes suivants que la principale distinction se situe au niveau de la modélisation du système, et en particulier du système de transport. La technologie de routage de notre système de production est composée exclusivement de convoyeurs à accumulation. Nous voulons n'utiliser que des

éléments définis par défaut dans les logiciels afin de montrer l'applicabilité de nos concepts à ces logiciels commerciaux, mais le choix de la primitive à utiliser doit se faire dans l'optique du recalage.

IV.4.1 Sous QUEST

IV.4.1.1 Primitive de transport

Nous souhaitons évaluer la pertinence des moyens de déplacement définis dans chacun des logiciels par rapport aux contraintes que nous avons dans ces travaux. Ces contraintes peuvent être définies en cinq points :

1. Modélisation de la topologie du système : lignes droites, virages…
2. Possibilité de donner une vitesse différente aux palettes dans les virages, les lignes droites, les aiguillages…
3. Modélisation des accumulations de palettes sur les convoyeurs ;
4. Possibilité de recaler la position d'une palette sur le convoyeur à l'arrivée d'un évènement extérieur ;
5. Possibilité de connaître la localisation exacte des palettes à tout instant.

Quest propose quatre primitives de transport que sont les humains (*Labors*), les chaines débrayables (*Power and Free*), les véhicules autoguidés (*AGV*) et les convoyeurs (*Conveyors*). Nous avons testé la compatibilité de ces primitives avec les conditions énoncées ci-dessus dans [Chové, 2007]. Le Tableau 2 présente ces résultats sous la forme de conditions *possibles* à réaliser représentées par un P, ou *difficiles* à réaliser représentées par un D.

Critères	1	2	3	4	5	Commentaires
Labor	D	D	D	P	D	Les humains peuvent se doubler facilement, ce qui constitue un avantage pour le recalage.
Power and Free	P	P	P	D	D	Le pilotage d'une chaine débrayable se fait par le produit, ce qui complexifie leur utilisation dans notre cas.
AGV	P	P	P	D	P	Les véhicules sont fortement contraints par la topologie du circuit, ce qui rend le recalage difficile à réaliser.
Conveyor	P	P	P	D	P	Toutes les conditions requises par nos travaux sont réalisables par l'utilisation de convoyeurs sauf le recalage..

Tableau 2 Comparaison des primitives de transport de Quest [Chové, 2007]

À la lecture de ce tableau, notre attention s'est vite focalisée sur les AGV et les convoyeurs. La difficulté réside dans la même condition, ce qui semble peu discriminant. Nous avons finalement opté pour l'utilisation de convoyeurs par analogie avec le système réel, également constitué de convoyeurs.

La particularité des convoyeurs sous Quest est la présence des points de décision. Les points de décision sont des éléments qui se positionnent sur un système de transport (Automatic Guided Vehicule, Power & Free, convoyeurs et humains) et qui permettent de contrôler les pièces qui passent à ce point. Tout comme les éléments de base, ces points sont contrôlés par des logiques qui permettent de définir leur comportement (retenue d'une pièce, routage, destruction…). La Figure 47 présente un exemple de routage de pièces. Les convoyeurs se déplacent de gauche à droite, et les pièces sont triées selon leur couleur au niveau du point de décision représenté par une croix. Une logic (routine) est déclenchée au passage d'une pièce sur le point. Il est possible de construire des classes de points de décision, contenant autant de points que nécessaires, qui utiliseront toutes la même logic.

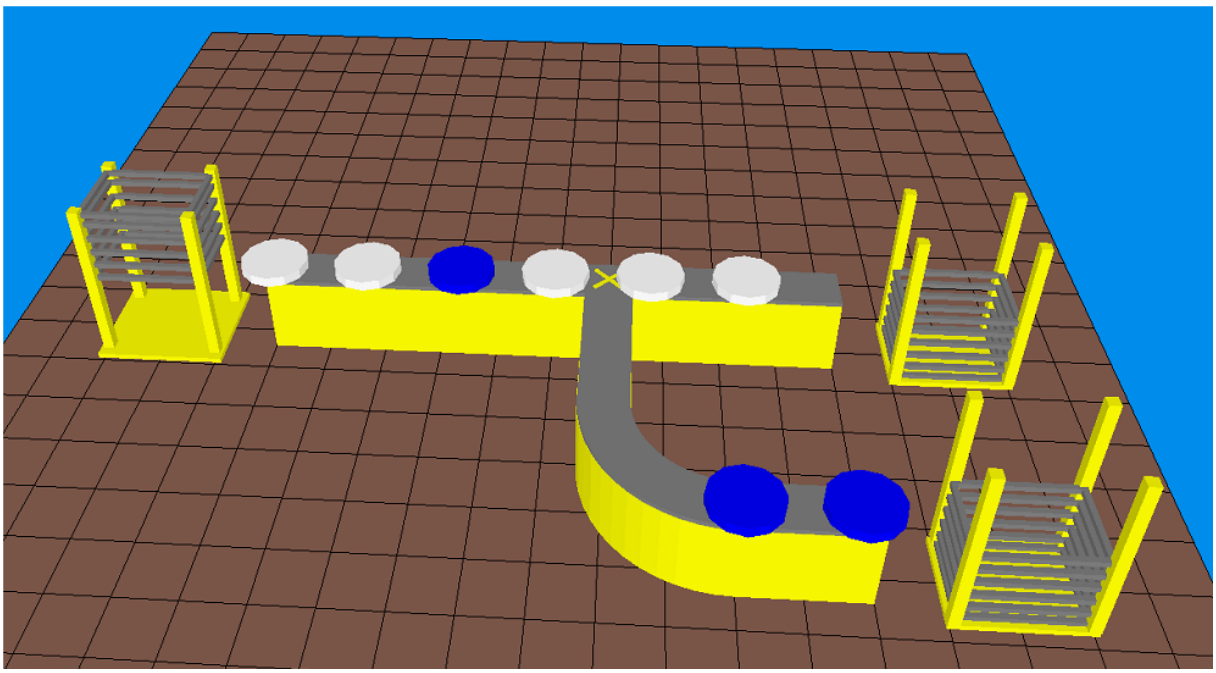

Figure 47 Exemple de routage de pièces sur un convoyeur

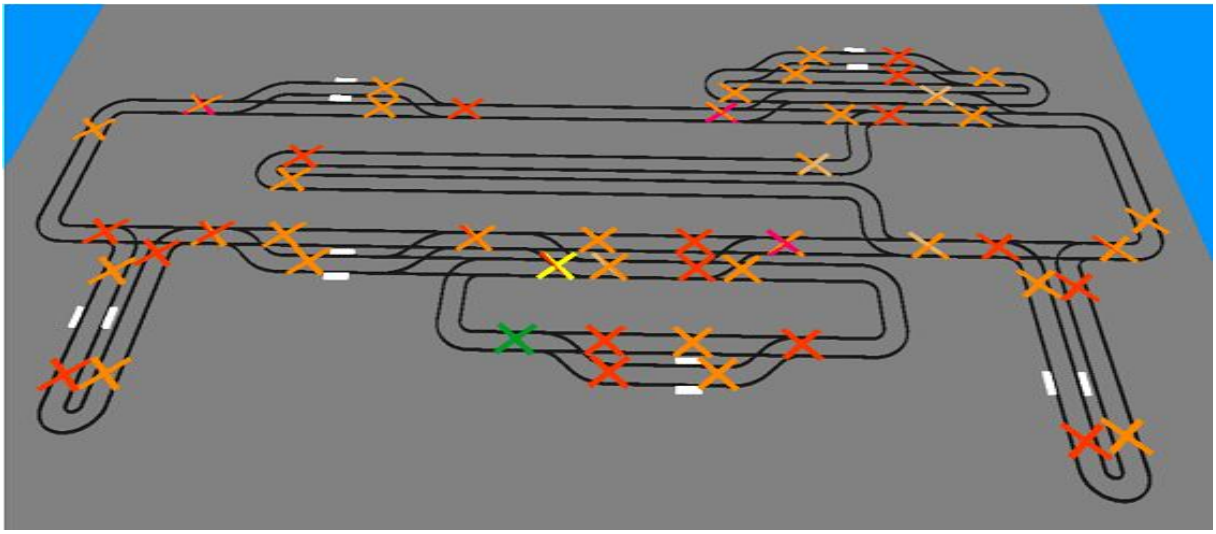

Figure 48 Les points de décision sur notre système

Les points de décision permettent d'ajouter sur un système de convoyage des entrées et des sorties qui ne sont pas situées aux extrémités des segments. Ils sont utiles pour réaliser des aiguillages, des convergences ou des points d'arrêts par exemple. La Figure 48 présente la disposition des convoyeurs et la position des points de décision concernant les convergences, divergences et postes de travail.

IV.4.1.2 Le recalage

La dynamique du système est principalement basée sur les palettes : ce sont elles qui déclenchent la plupart des évènements de par leur présence, leur mouvement ou bien encore par le temps passé devant un poste de travail. Le recalage le plus important concerne donc, sur ce système, le positionnement des palettes. Nous avons vu dans le paragraphe précédent que nous avions porté un soin particulier aux éléments de convergence pour permettre de conserver l'ordre exact des palettes sur les convoyeurs.

À chaque passage d'une palette devant un capteur, deux informations sont disponibles : sa date d'arrivée et sa date de départ. Ces informations parviennent à l'observateur sous la forme d'évènements. Le premier évènement aura pour conséquence de récupérer la palette dans le simulateur à l'endroit où elle se trouve actuellement et de l'insérer à l'endroit du capteur. L'extraction de palettes depuis un convoyeur ne peut se faire qu'à la sortie de celui-ci ou à un point de décision. Afin d'extraire le plus rapidement possible la palette, nous proposons de placer des points de décision tout au long des convoyeurs. Plus les points seront rapprochés, plus il sera possible d'extraire rapidement la palette, mais en contrepartie plus le modèle sera chargé. Nous ajoutons aux palettes un attribut (que nous nommons *gotopt*) qui correspond au numéro du capteur où la palette a été vue dans le système piloté. Par défaut, cet attribut a une valeur nulle. Lorsqu'un capteur détecte une palette, l'observateur modifiera l'attribut de la palette correspondante. Dès que la palette passe devant un point de décision, celui-ci commandera l'extraction de cette palette pour l'envoyer vers un stock intermédiaire. Celui-ci se charge ensuite de réinjecter la palette au bon endroit dans le système (grâce à la lecture de l'attribut) avec son attribut remis à zéro.

Cette technique impose deux contraintes : tout d'abord, on ne peut recaler que lorsque l'on connait la palette concernée, ce qui restreint son application aux seuls lecteurs d'étiquette et non aux simples détecteurs de présence. Par contre, elle permet de détecter au niveau de l'observateur des inversions de palettes. Si l'on inverse manuellement deux palettes sur la chaine, cette inversion sera prise en compte par l'observateur lors des prochains passages des palettes devant une unité de lecture/écriture. Pour utiliser les simples détecteurs de présence palette pour faire du recalage, il est possible d'ajouter une recherche de palette dans l'observateur en prenant comme hypothèse que l'ordre est correctement respecté. Toutefois, cela fait intervenir des techniques d'algorithmes de graphe pour trouver la palette la plus près du détecteur, ce qui complexifie nettement la technique que nous venons de proposer. Nous avons évalué que notre système ne nécessitait pas de telles possibilités grâce à la répartition suffisante des lecteurs tout autour de celui-ci : l'information n'est pas dégradée sans la prise en compte du recalage sur les détecteurs.

La seconde contrainte est que tous les points de décisions doivent être connectés au stock tampon (pour l'extraction des palettes), et que le stock tampon doit à son tour être connecté à tous les points de décision correspondant aux lecteurs (pour leur réinsertion).

Cette contrainte est évacuée par la notion de classe contenue dans Quest. Tous les points de décision ne sont pas regroupés dans la même classe, car leurs fonctions ne sont pas toujours identiques. Toutefois, le nombre de classes est très limité. Il suffit alors de connecter ces classes au stock tampon, puis de connecter le stock à la classe des points correspondant aux lecteurs.

IV.4.1.3 Modélisation des divergences

Sur le système réel (Figure 49a), une palette est bloquée à l'entrée d'une divergence par une butée, puis est relâchée vers un aiguillage l'envoyant vers un convoyeur ou la laissant sur l'actuel. La décision sur l'avenir de la palette est prise par l'intermédiaire des informations contenues sur l'étiquette de la palette, ce qui impose la présence d'un lecteur d'étiquette en face de la butée. La particularité de ce système est que la palette est relâchée par la butée après que la décision ait été appliquée, c'est-à-dire après que l'aiguillage ait pris sa position.

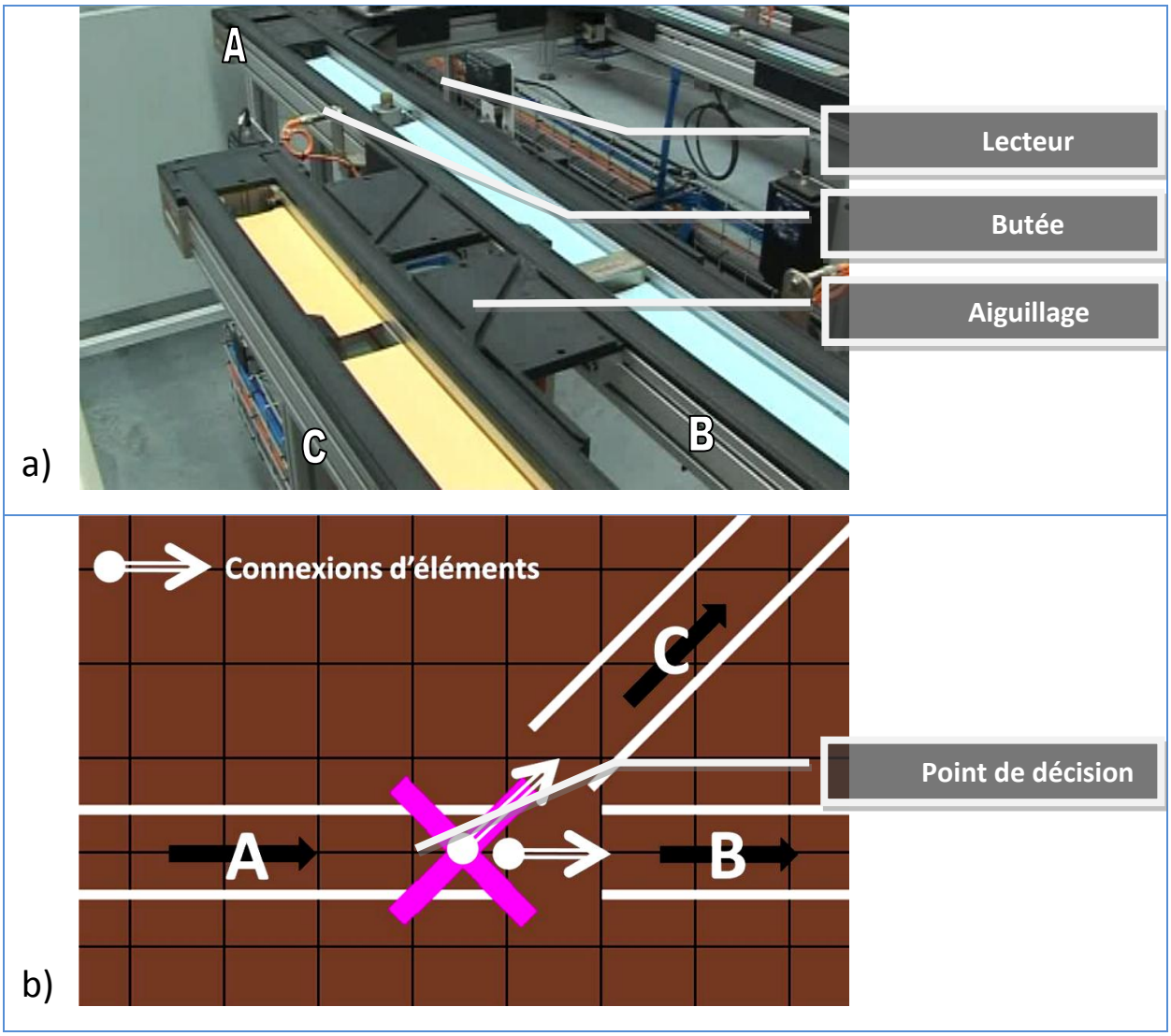

Figure 49 Modélisation d'une divergence sous Quest
a) La divergence réelle b) Modèle Quest

Nous proposons, pour modéliser les divergences, d'utiliser un point de décision correspondant à la butée réelle (Figure 49b). Lorsqu'une palette arrive à la divergence, un mécanisme de recalage est exécuté tel que défini dans le chapitre précédent, grâce à la présence du lecteur. La palette est ensuite bloquée dans l'observateur au point de décision (dans sa *process logic*) jusqu'à ce que l'évènement de descente de la butée arrive. À ce moment, on regarde la position de l'aiguillage. Si la palette n'est pas aiguillée, le point de décision laisse passer la palette (grâce à la commande SCL *resume_travel*) ; si, au contraire, la palette est aiguillée, le point de décision extraira la palette et la mettra sur le convoyeur C (grâce à la commande SCL *route cpart*).

Avec ce type de modélisation, il se pose un problème lorsque la palette réelle passe avant la palette simulée au point de divergence. Dans ce cas, l'information de routage est stockée dans une pile dans Quest jusqu'à ce qu'une palette arrive.

IV.4.1.4 Modélisation des convergences

Au niveau d'une convergence dans le système réel, la seule décision prise est l'ordre dans lequel les palettes doivent passer. En effet, si deux palettes passent simultanément, il y a risque d'arcbouter la palette dans l'aiguillage en la poussant avec celle qui va tout droit. Un jeu de capteurs et de butées est donc installé à chaque entrée de la convergence pour filtrer les palettes, et un capteur donne l'information de sortie de la convergence à la commande. Il est crucial de bien modéliser cette convergence pour être certain de conserver le même ordre sur l'observateur que sur le système réel.

La solution que nous avons appliquée est très simple. Un point de décision est disposé à chaque entrée de la convergence. La séquence de descente des butées sur le système réel, correspondant à la provenance des palettes successives dans la convergence, est mémorisée dans une pile, et rejouée dans l'observateur à mesure que les palettes se présentent aux points de décision.

IV.4.1.5 Le temps-réel sous Quest

Comme la plupart des outils de simulation, QUEST est d'abord un outil permettant la prévision de comportement de systèmes de production. Pour cela, on cherche à faire évoluer le temps le plus rapidement possible. Ainsi, lors de l'exécution d'une simulation sous Quest, le processeur de l'ordinateur sur lequel la simulation tourne est utilisé à 100% de sa capacité de calcul. La seule façon de réduire la vitesse de simulation est d'augmenter la résolution de l'affichage et la fréquence de rafraichissement de l'affichage. Or, nous cherchons ici à tourner en temps réel, c'est-à-dire avec une seconde simulée en une seconde exactement.

Cette possibilité n'est pas offerte par défaut sous Quest. Nous l'avons donc reconstruite [Chové, 2007] en utilisant une logic personnalisée assignée à une machine extérieure au système étudié, que nous appelons *serveur de temps*. Cette machine n'a, comme fonction, que de comparer l'avancée de la simulation sur la dernière seconde simulée et l'avancée de l'horloge de l'ordinateur, considérée comme étant le temps de référence. Si la seconde simulée est plus courte que la seconde réelle, le simulateur est alors ralenti, et *vice versa*. Nous asservissons donc l'évolution du temps dans le simulateur à l'horloge de l'ordinateur.

Cette solution a l'avantage de s'adapter automatiquement en cas de modulation de la charge du simulateur, c'est-à-dire d'une augmentation du nombre d'éléments à animer. Il convient toutefois de préciser que le modèle utilisé comme observateur doit être toujours capable de simuler plus vite que le temps réel, pour pouvoir assurer le temps-réel (voir paragraphe précédent).

IV.4.2 Sous ARENA

IV.4.2.1 Primitive de transport

Le langage SIMAN nous permet d'utiliser deux moyens de transport, dénommés *Transporters* et *Conveyors* (transporteurs et convoyeurs). Les convoyeurs permettent de remplir toutes les conditions, sauf la quatrième qui est complexe à mettre en place (recalage). En effet, il est difficile de respecter cette condition sans définir un nombre très important de stations uniformément réparties sur le circuit et permettant de gérer très finement le comportement de ces convoyeurs, ce qui revient à adapter la solution que nous avons développée sous Quest.

De la même manière, les transporteurs permettent une mise en place relativement facile de tous les points sauf le quatrième. Il n'y a donc pas de caractère discriminant entre les deux primitives à ce niveau. Nous proposons donc, pour diversifier les approches, d'utiliser les transporteurs sous Arena.

Ces transporteurs se déplacent sur un *circuit*, regroupement d'un ensemble de *liens*, chacun reliant deux *intersections*. Les liens sont de plus décomposés en *zones*, que nous avons imposées de la longueur d'une palette dans notre modélisation de la topologie du système, chaque palette mesurant 20 cm de long. Chaque transporteur peut avoir une vitesse différente de celle d'un autre. Toutefois, en fonctionnement normal, la vitesse de tous les transporteurs est d'une zone par seconde, soit égale à la vitesse théorique annoncée par le constructeur de 20 cm/s.

La programmation Quest se fait entièrement par programmation textuelle : ce sont les logics des éléments ou du modèle qui vont nous permettre de gérer les évènements extérieurs. Sous Arena, nous avons choisi d'utiliser au maximum les primitives SIMAN. Ce choix nous a tout de même conduit à donner une place importante au VBA pour ce qui concerne notamment la gestion de la communication. Nous avons construits une bibliothèque de modules permettant de gérer les points difficiles de la programmation de ce système. Nous proposons de détailler ces modules.

IV.4.2.2 Le recalage

Les liens sont monodirectionnels, et ne permettent pas aux transporteurs de se doubler. Nous avons donc choisi d'accélérer infiniment les palettes pour les recaler à l'endroit où elles sont sensées être. Ce choix implique nécessairement l'hypothèse selon laquelle les convergences ont correctement rempli leur rôle et que l'ordre des palettes est donc scrupuleusement respecté sur les convoyeurs. Ainsi, le recalage que nous avons proposé en utilisant ARENA ne permet pas de prendre en compte une inversion manuelle des palettes.

La Figure 50 détaille la programmation que nous avons réalisée pour recaler l'évolution d'une palette aux abords d'un lecteur quelconque, noté *i*. Une entité, avec en attribut le numéro de la palette concernée, est envoyée par le VBA aux blocs Trace lorsque l'évènement correspondant survient sur le système réel. Lorsque cet évènement correspond à l'arrivée d'une palette au lecteur, la palette correspondante est accélérée à l'infini pour rattraper le retard éventuel qu'elle a pu prendre. Si tel n'était pas le cas, c'est-à-dire si l'entité représentant la palette simulée était précédemment arrivée à la station correspondant à la position du lecteur i sur le circuit et était stockée dans la file d'attente suivante, alors cette accélération n'aurait aucun effet, puisque le transporteur est à l'arrêt.

Dès qu'une entité est présente dans chaque file avec le même numéro en attribut (en pratique, chaque file ne contient jamais plus d'une entité), l'entité représentant la palette réelle est détruite, alors que l'entité en charge du transporteur simulé continue et entre dans une file d'attente. Sa présence dans cette file dénote du fait qu'elle est prête à partir dès que l'évènement correspondant survient sur le système réel. Lorsque c'est le cas, une entité arrive dans la file d'attente « Départ palette réelle ». Il y a alors une correspondance possible entre une entité de chaque file d'attente, et les entités subissent le même traitement que précédemment : l'entité représentant la palette réelle est détruite, alors que l'entité en charge du transporteur simulé peut continuer l'évolution du simulateur. La première opération réalisée est alors de redonner au transporteur sa vitesse normale.

Ce modèle distingue particulièrement les flux d'informations provenant de la simulation et du système réel pour pouvoir facilement les synchroniser. Il gère de plus la correspondance entre la simulation et l'extérieur à deux niveaux : à l'arrivée et au départ de la palette. Cela correspond généralement à l'apparition puis à la disparition d'un évènement sur le système réel. Cette disposition est particulièrement importante pour la modélisation des zones de travail des postes de la chaine. En effet, l'opération sur le produit transporté par la palette peut être manuelle. Dans ce cas, sa durée n'est pas connue. Il est donc important que ce soit la sortie de la palette de la zone de travail dans le système réel qui provoque l'évolution de la palette dans l'observateur.

À l'application de ce module dans notre modèle d'observateur, les palettes en retard n'étaient jamais accélérées. En effet, toutes les primitives de déplacement de transporteurs proposées dans SIMAN n'autorisent pas la modification de la vitesse du transporteur en cours de déplacement. Si un ordre de modification de la vitesse est envoyé durant un trajet, c'est-à-dire pendant un déplacement vers une destination précise, matérialisée par une zone ou une intersection, la modification ne sera appliquée qu'à l'arrivée à destination.

Une solution aurait été de définir un grand nombre d'intersections de manière à avoir un retard minimum. Toutefois, cela complexifiait inutilement le modèle. Nous avons alors opté pour la solution présentée Figure 51.

Lorsqu'un transporteur doit se déplacer entre deux intersections, il emprunte un lien, découpé en zones de longueur égale. Cette solution propose d'utiliser le découpage existant par défaut par l'intermédiaire des zones au lieu d'en recréer un artificiel en multipliant les intersections. Nous incrémentons un compteur à chaque arrivée en bout de zone, et demandons au transporteur de se rendre à la zone dont le numéro est égal au compteur, c'est-à-dire la prochaine.

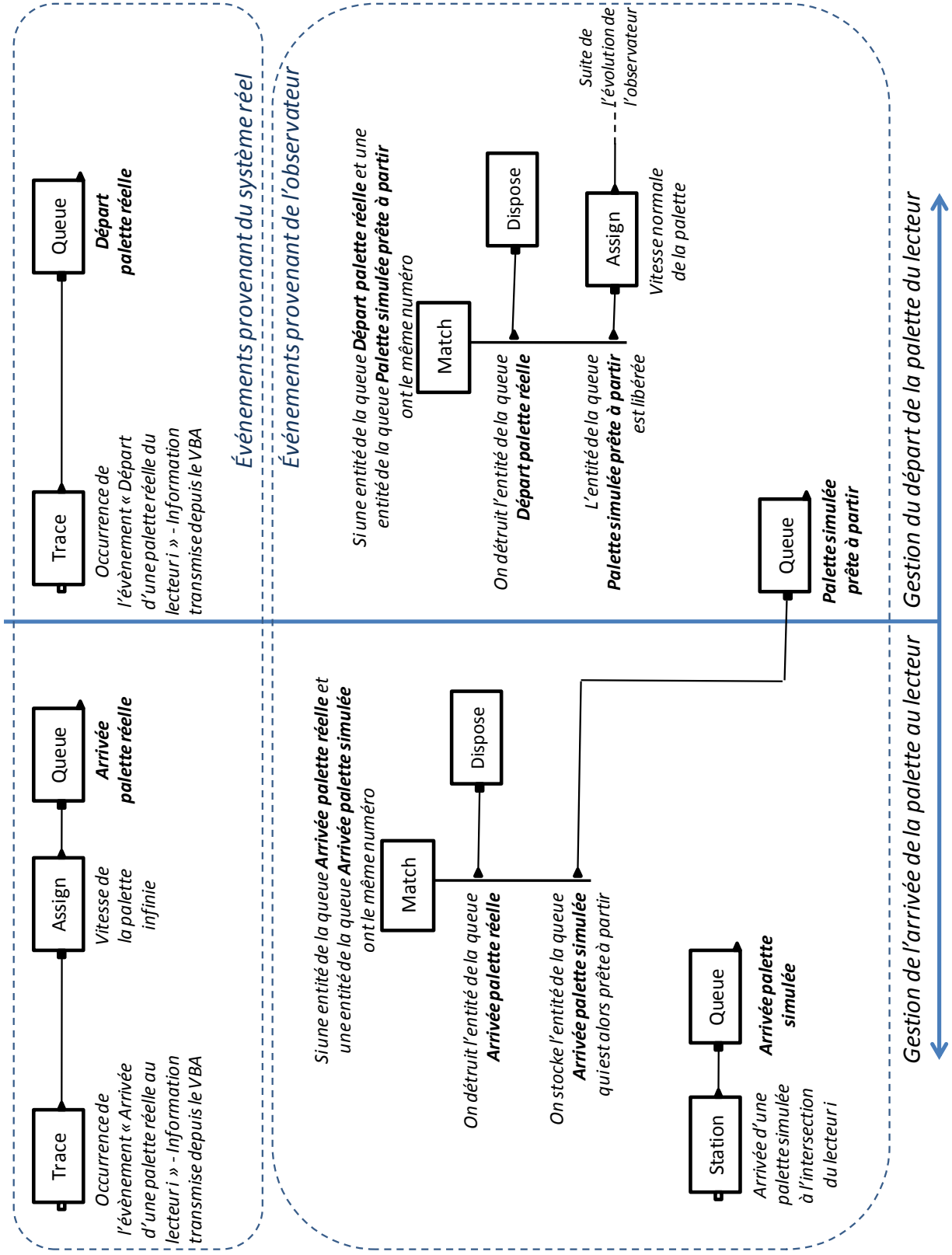

Figure 50 Le recalage à un lecteur en SIMAN

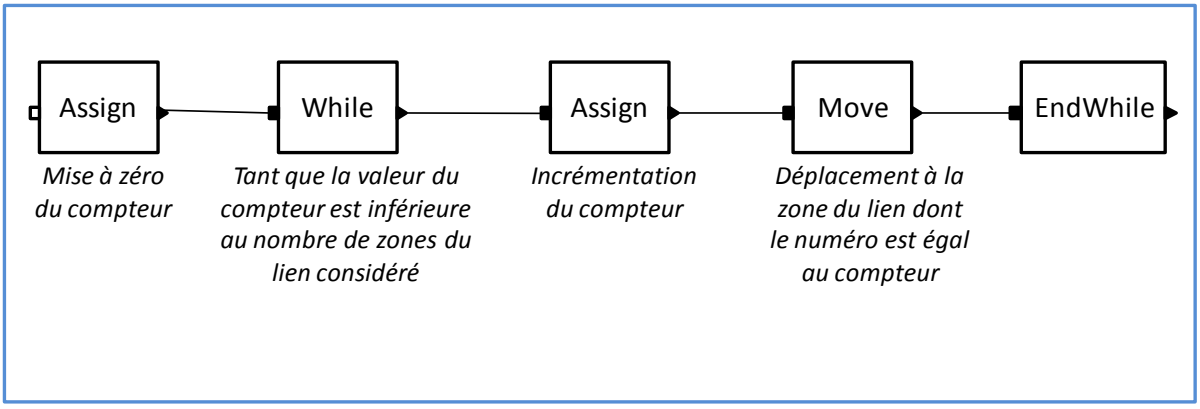

Figure 51 Le déplacement d'un transporteur dans l'observateur sous SIMAN

Ainsi, le transporteur arrive très souvent en fin de trajet, et il est alors possible de modifier sa vitesse plus souvent. Étant donné la longueur d'une zone (20 cm) et la vitesse moyenne théorique des transporteurs (20 cm/s), nous pouvons estimer le retard maximum dans le recalage du à ce problème à une durée inférieure à la seconde, ce qui est satisfaisant sur ce type de systèmes. Le modèle présenté Figure 51 est utilisé pour définir un module de transport dans la bibliothèque présentée précédemment. Il remplace dans l'observateur les blocs classiques *Route*, *Transport* ou *Move* permettant de faire bouger un transporteur.

IV.4.2.3 Modélisation des divergences

Comme nous l'avons vu au paragraphe IV.4.1.3, une divergence est un ensemble constitué d'un lecteur, d'une butée et d'un aiguillage. Comme nous l'avons vu sous Quest, nous avons choisi d'effectuer un recalage classique sur le lecteur, puis à la descente de la butée, nous regardons la position de l'aiguillage pour savoir vers où orienter la palette. Un simple module *Branch* à la sortie d'une primitive de recalage suffit donc à modéliser une divergence. Si la palette simulée est en avance sur le réel, elle est bloquée à l'entrée de l'aiguillage en attente de l'évènement réel. Si la palette simulée est en retard, les différentes positions de l'aiguillage au cours des évènements antérieurs sont enregistrées et rejouées à mesure que les palettes simulées arrivent à l'aiguillage.

IV.4.2.4 Modélisation des convergences

La solution appliquée sous Arena est, sur le principe, la même que celle que nous avons déjà évoquée pour Quest, c'est-à-dire l'enregistrement des évènements se déroulant sur le système rejouée ensuite dans l'observateur. Toutefois, nous avons choisi ici d'utiliser le langage SIMAN pour construire cet enregistrement, et non un langage textuel (Figure 52).

Arena est fortement basé sur le principe des files d'attentes. L'enregistrement que nous souhaitons reconstruire est tout à fait comparable à une telle file. Nous proposons d'utiliser cinq files d'attente pour cette convergence. Deux d'entre elles sont alimentées par les entités arrivant sur les convoyeurs d'entrée de la convergence : ceci correspond à l'arrivée de la palette simulée sur les entrées notées 1 et 2 de la convergence. Deux autres sont alimentées par le VBA envoyant des entités correspondant à des évènements du système réel : ceci correspond à l'arrivée de la palette réelle aux entrées.

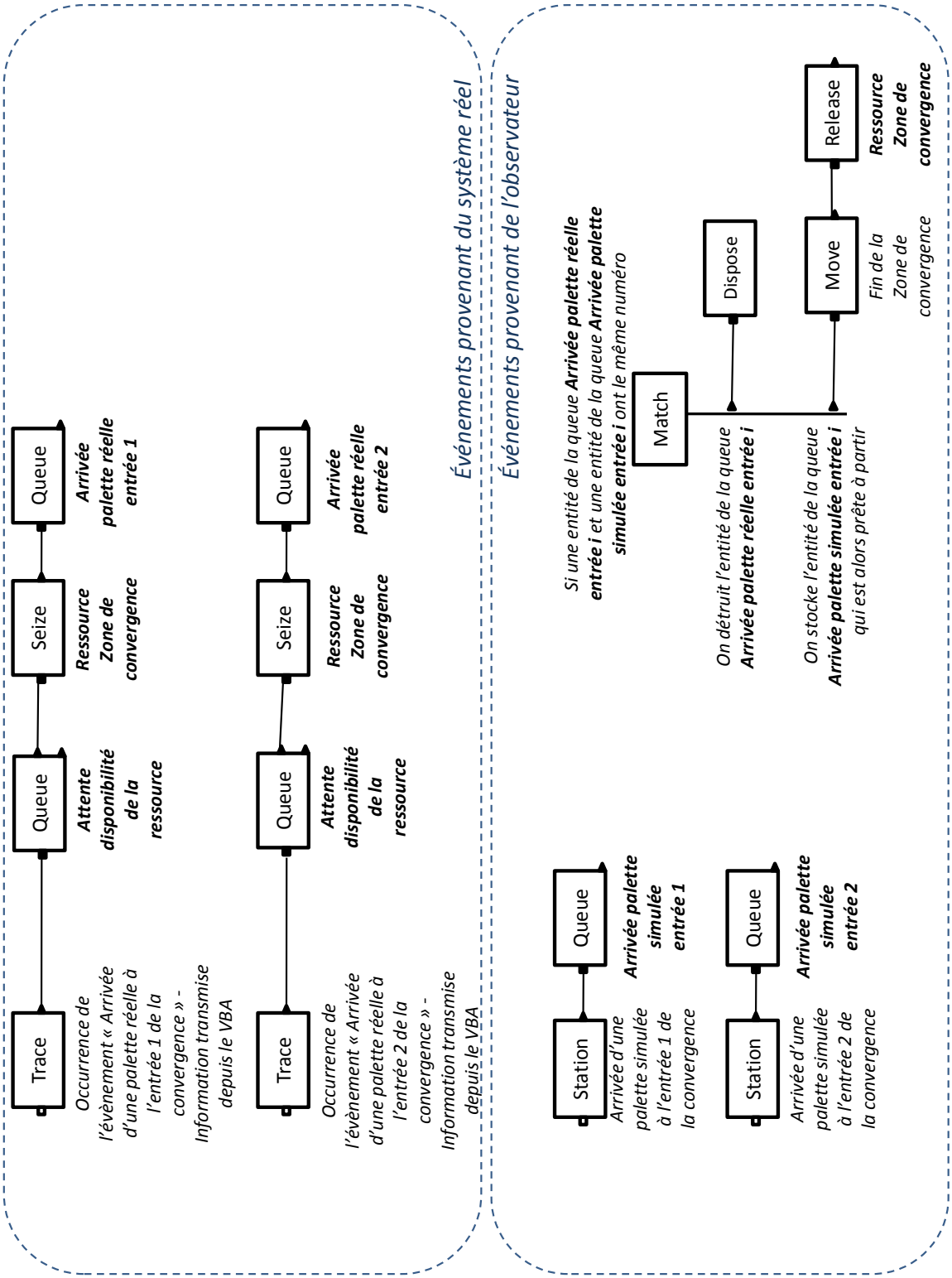

Figure 52 Modèle d'une convergence sous Arena

La dernière file d'attente correspond à l'ordre dans lequel les palettes sont passées dans le système réel, et doivent passer dans l'observateur. Lorsqu'une palette réelle arrive à une entrée, l'entité correspondant à l'évènement réserve une ressource que l'on a définie comme correspondant à la zone de la convergence (module *Seize*). Elle est ensuite stockée

dans la file correspondante. Lorsque la palette simulée arrive à son tour, l'observateur peut poursuivre son évolution puisque les deux files contiennent des entités correspondantes.

La raison de la déclaration de la ressource vient lorsque l'observateur a du retard sur un convoyeur. Prenons un exemple. Une palette arrive à chaque entrée de la divergence. Le système réel décide de les faire passer dans l'ordre : entrée 1 puis entrée 2. Si l'évènement simulé correspondant à l'arrivée d'une palette à l'entrée 2 arrive avant celui de l'entrée 1, il faut alors bloquer son passage. La réservation de la ressource permet cela. Cette ressource a une capacité unitaire constante. Si la palette de l'entrée 1 est passée en premier, elle est alors la seule à pouvoir réserver la ressource et donc arriver dans la file *Arrivée palette réelle entrée 1*. Elle est donc la seule à pouvoir faire évoluer l'observateur. Lorsque les autres palettes vont arriver, les entités correspondantes, ne pouvant pas réserver la ressource, vont être classées dans leur ordre d'arrivée dans la file d'attente précédent le module *Seize*. On a ainsi une trace de l'ordre d'arrivée contenue dans la file d'attente.

IV.4.2.5 Le temps-réel sous Arena

Au contraire de Quest, Arena propose un module temps-réel dénommé Arena RT permettant de caler automatiquement l'horloge de simulation sur l'horloge du système d'exploitation de l'ordinateur. Ce module est très efficace puisqu'il permet en outre de rattraper le retard qu'il a éventuellement pris à cause d'une charge trop élevée les instants précédents. Même s'il est fortement déconseillé d'en arriver à cette situation, cette fonctionnalité peut s'avérer par exemple très utile dans des objectifs de robustesse de l'outil.

Nous avons bien évidemment utilisé cet outil dans nos travaux.

IV.4.2.6 La sauvegarde de l'état

L'objectif de l'observateur, et donc une de ses fonctionnalités principales, est de pouvoir sauvegarder son état sous la forme de fichiers textuels. Cette sauvegarde est déclenchée par le simulateur et est exécutée par une routine VBA. La Figure 53 présente l'algorithme utilisé pour sauvegarder l'état de notre observateur. Cet algorithme ne concerne pas les variables dont la sauvegarde est triviale. En sortie de cet algorithme, trois fichiers sont disponibles :

- Un fichier « Queues », listant l'identité et le rang de chaque entité contenues ;
- Un fichier « Transporteurs », contenant la position, l'état et l'identité de l'entité contrôlant chaque transporteur ;
- Un fichier « Ressources » contenant le nombre d'unités réservées de chaque ressource ;
- Un fichier « Entités », listant l'identité de chaque entité importante dans la définition de l'état et la valeur de ses attributs.

IV.5 Modélisation du simulateur

Après avoir construit un observateur, nous avons développé une application d'aide à la décision permettant de valider notre approche. Nous avons choisi de traiter la première décision évoquée dans ce chapitre (paragraphe IV.2.1) sous Arena. Le principe est d'aider le

pilote de la production dans le paramétrage d'un ordre de fabrication. Notre application permettra, à la saisie d'un ordre, de calculer une date de fin estimée de tous les ordres en cours et en préparation (saisis mais pas encore lancés). Ainsi, l'opérateur pourra tester l'influence du nombre de palettes impliquées dans cet ordre sur la date de fin de l'ensemble des ordres.

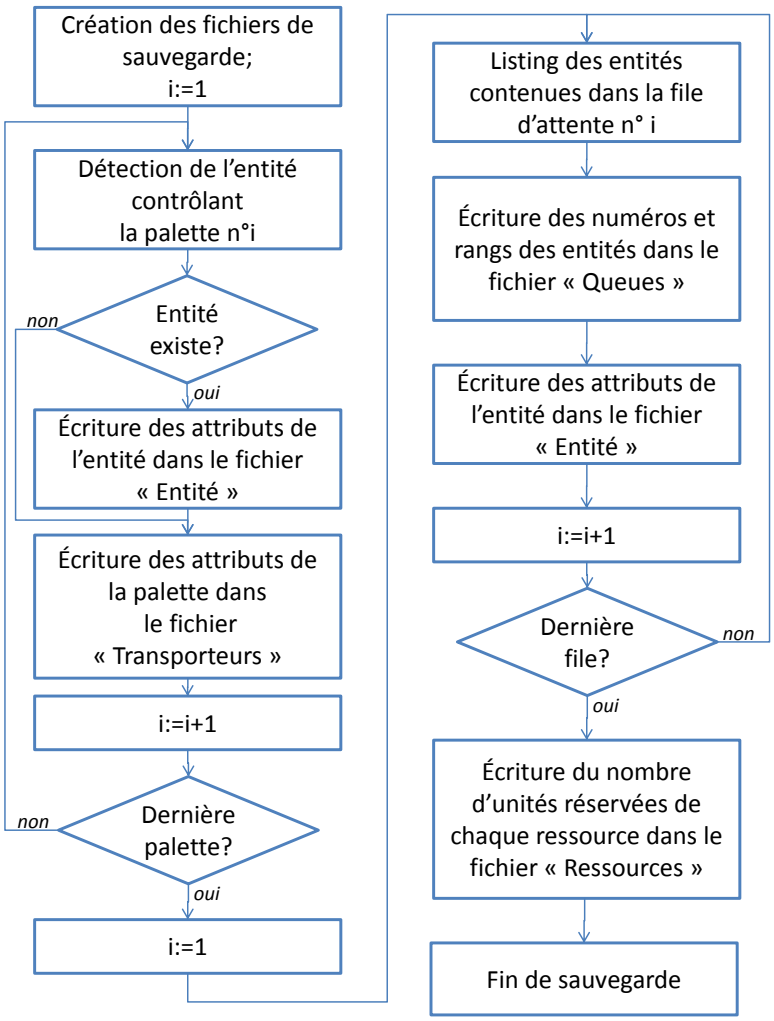

Figure 53 Algorithme de sauvegarde de l'état de l'observateur

Nous avons construit un modèle dédié à cette application, et qui a donc besoin d'un nombre minimal de paramétrage extérieur pour fonctionner. De ce fait, dès qu'il reçoit une notification sous forme de socket de la part du MES (Figure 41), le modèle va directement lire au niveau de la base de donnée du MES par requêtes SQL la liste des ordres de fabrication en cours ou en préparation et toutes les données de production nécessaires à la simulation (répartition des opérations sur les postes, temps opératoires, temps de réglages, etc.). L'observateur met également à sa disposition une copie de son état sous forme de fichiers textuels, tels que définis au Chapitre II.

La première opération que fait Arena lorsqu'on lance un modèle « classique » est de le compiler. À partir de ce moment, aucun changement important ne pourra intervenir dans la structure du modèle. Or, la position initiale des transporteurs sur le circuit doit être lue

depuis le fichier d'état de l'observateur, ce qui impose de modifier le module de définition des transporteurs après le début de l'exécution du modèle. Le VBA intégré à Arena nous offre la possibilité d'exécuter une routine entre le lancement du modèle et le début de la compilation grâce à l'évènement *RunBegin*. C'est donc ici que la position des transporteurs sera prise en compte. Cela impose que toutes les simulations commençant avec une position différente des transporteurs devront recommencer une compilation du modèle, ce qui augmente les temps de calcul. Toutefois, généralement, les réplications multiples pouvant survenir lors de la prise en compte d'évènements stochastiques, ou lors du test de plusieurs hypothèses partent avec un paramétrage différent mais avec le même état initial, et donc la même position initiale des transporteurs. Ce handicap n'intervient donc pas dans nos travaux.

Une seconde contrainte survient de la forme des fichiers d'état de l'observateur. En effet, l'état de l'observateur est connu sous forme chiffrée, n'ayant de sens que par rapport à la modélisation de l'observateur. Par exemple, on peut connaitre la position d'un transporteur sous la forme : Lien 5 – zone 3. Or, le lien 5 d'un modèle ne correspond pas obligatoirement au lien 5 d'un autre modèle. Il est donc préférable de conserver un circuit identique à la construction du simulateur à celui utilisé dans l'observateur. Ce constat est généralisable à toutes les déclarations de variables, ressources, files d'attente, etc. Les déclarations doivent être strictement identiques dans les deux simulateurs pour être capable d'identifier l'élément de l'observateur avec l'état duquel on cherche à initialiser l'élément du simulateur correspondant.

L'étape suivante est d'initialiser toutes les variables du modèle, ainsi que de régler la date à celle de l'envoi de la requête par l'observateur, date à laquelle celui-ci à enregistré son état. Cette mise à jour doit se faire après la compilation, mais être le premier évènement exécuté. Le VBA d'Arena nous permet d'exécuter une routine lors de l'évènement *RunBeginSimulation* ayant ces caractéristiques.

À partir de ce moment, il est possible de travailler sur les entités du modèle. Le but est de les replacer au même endroit et dans le même état où elles étaient dans l'observateur au moment de l'enregistrement. Pour les files d'attente, il n'y a pas de problèmes. Toujours dans le RunBeginSimulation, les entités sont créées, on leur assigne les valeurs des attributs correspondants et on les envoie dans les files d'attente correspondantes. Pour les ressources, on créé une entité à qui l'on demande de réserver toutes les unités de ressources réservées dans l'observateur. On détruit ensuite cette entité.

Lorsque les transporteurs sont en simple transport, il suffit de créer une entité, de lui assigner les attributs de l'entité qui pilotait le transporteur dans l'observateur, de réserver le transporteur, puis d'envoyer l'ordre de se déplacer à la destination prévue du transporteur. Le concept de stations d'Arena permet d'identifier la présence d'un transporteur à un endroit donné avec les actions que l'entité qui le pilote doit effectuer. Si le transporteur est à une station, alors l'entité qui le pilote est en train d'effectuer une opération. Le fonctionnement d'Arena permet de simplifier le problème. Ceci a déjà été évoqué au paragraphe II.2.2.2. Si l'entité est dans une file d'attente, on utilise la technique présentée ci-dessus en ajoutant la réservation du transporteur par l'entité. Si l'entité est dans un module d'attente (*Delay*), il est nécessaire de connaître sa position et la date à laquelle elle est censée en sortir. Cette information doit être obtenue depuis l'observateur. Nous proposons de la stocker dans un attribut dédié, qui sera nul lorsque l'entité n'est pas dans

un module d'attente. L'entité réserve tout de suite le transporteur et est réintroduite dans le module suivant le module Delay à la date de sortie prévue.

Le paragraphe II.2 détaille les mécanismes mis en jeu pour l'initialisation du simulateur. Cette initialisation requiert une modélisation proche de celle de l'observateur, ce qui poussera l'utilisateur final à concevoir son modèle de simulation à partir du modèle d'observation. L'initialisation est principalement conduite en VBA pour ce qui concerne l'intégration des informations au simulateur grâce aux fonctions avancées qu'il propose, et à son adaptation au traitement de données. Le VBA a ensuite pour objectif d'introduire à bon escient les entités permettant de faire vivre le modèle.

IV.6 L'architecture implantée

L'architecture de notre système ne comprend pas d'élément du niveau de la planification (pas d'ERP, APS, etc. – voir Chapitre I). Elle se restreint donc, outre les équipements de terrain (niveau de l'opération) et sa commande bas-niveau (niveau de la commande), à un serveur OPC, un observateur et un MES – constitué d'une base de données, d'un utilitaire de supervision et d'un module d'aide à la décision (niveau de l'exécution).

La Figure 54 présente en détail l'ensemble des éléments de cette architecture où les moyens de communication sont détaillés. La communication se fait majoritairement sur Ethernet. Seuls les modules de lecture/écriture des étiquettes électroniques sont connectés aux automates par liaison FipIO et les pupitres par liaison série.

La Figure 55 présente l'implantation physique des éléments sur le système. L'observateur et la simulation en ligne se font nécessairement sur des ordinateurs distincts pour libérer de la ressource processeur et ainsi accélérer les simulations, et parce que des logiciels tels qu'Arena n'autorisent qu'une seule exécution à la fois sur chaque processeur, ce qui n'est pas le cas de Quest par exemple. À l'opposé, les serveurs peuvent être placés sur la même station que l'observateur pour diminuer les latences dues au réseau lors des échanges d'informations et parce que l'observateur ne nécessite généralement pas beaucoup de ressources processeur. Nous avons opté pour une solution différente avec les serveurs sur la même station que le MES dans un souci de modularité de l'architecture.

Notre architecture est principalement construite autour d'un réseau Ethernet Industriel. Ces réseaux sont apparemment identiques aux réseaux de bureautique utilisés dans le reste de l'entreprise, mais il est largement préférable de séparer ces deux réseaux. En effet, les critères de Qualité de Service, généralement notés QoS, ont besoin d'être souvent largement meilleurs pour un réseau dédié à la commande que pour un réseau dédié à la bureautique.

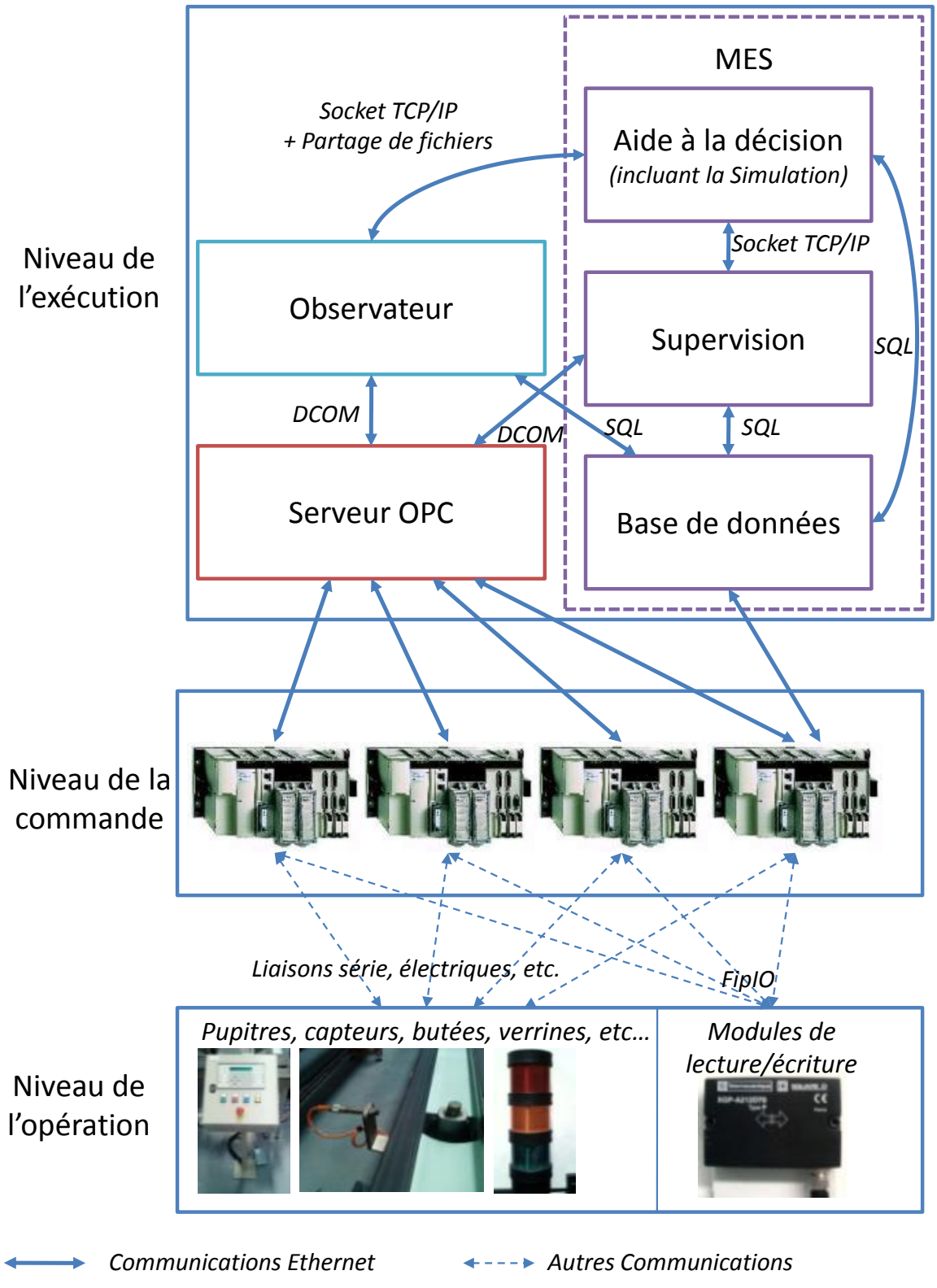

Figure 54 L'architecture complète

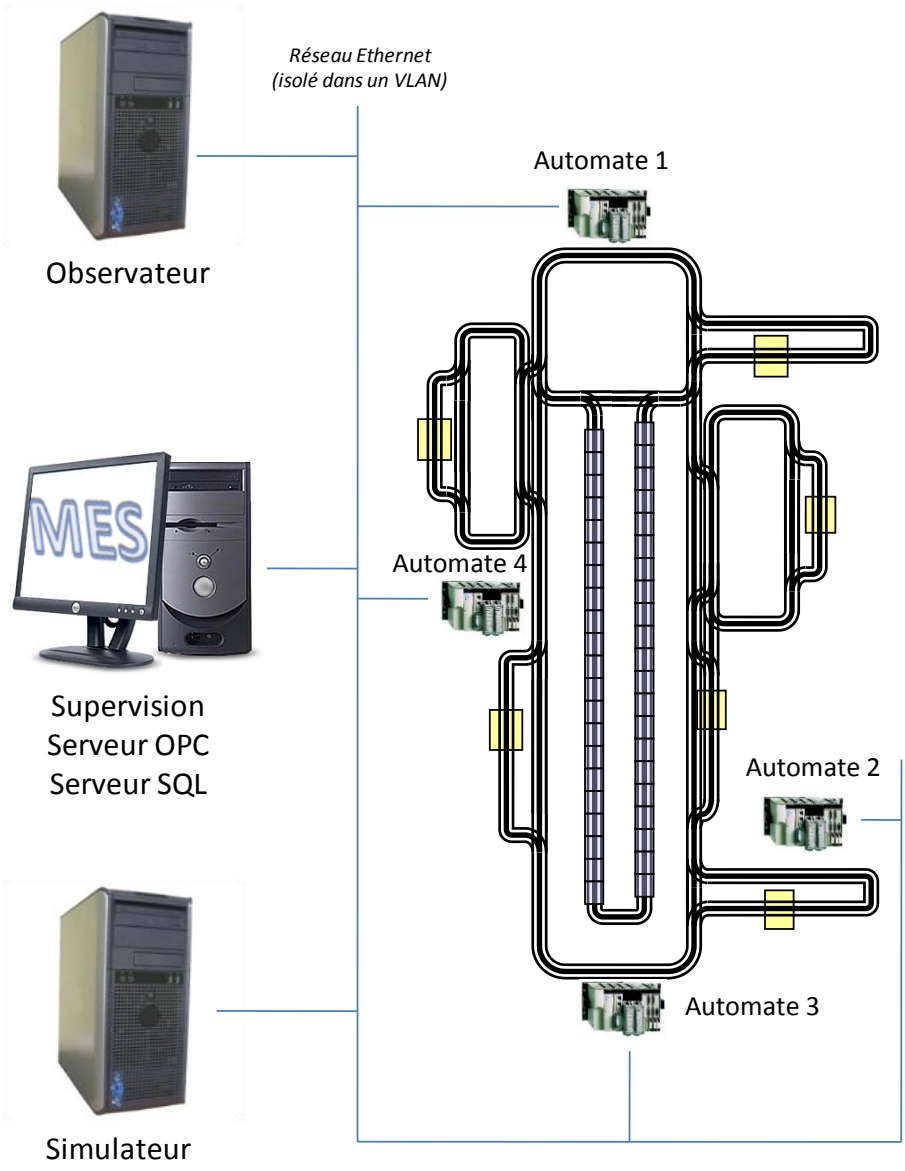

Figure 55 Implantation matérielle de la solution

Les principaux critères permettant d'apprécier la qualité de service sont les suivants :

- Débit (en anglais *bandwidth*), parfois appelé *bande passante* par abus de langage : il définit le volume maximal d'information (en bits) transmis par unité de temps ;
- Gigue (en anglais *jitter*) : elle représente la fluctuation du signal numérique, dans le temps ou en phase ;
- Latence, délai ou temps de réponse (en anglais *delay*) : elle caractérise le retard entre l'émission et la réception d'un paquet ;
- Perte de paquet (en anglais *packet loss*) : elle correspond à la non-délivrance d'un paquet de données, la plupart du temps due à un encombrement du réseau ;
- Déséquencement (en anglais *desequencing*) : il s'agit d'une modification de l'ordre d'arrivée des paquets.

Le critère de latence nous intéresse particulièrement dans la comparaison des deux utilisations d'Ethernet [Koubâa et Song, 2002]. En effet, il est souvent primordial d'assurer une latence de l'ordre de la milliseconde entre deux automates dont les temps de cycle sont de cet ordre, ou entre deux commandes de robots se déplaçant à grande vitesse. Or, le trafic bureautique, si les deux réseaux sont communs, a tendance à charger les routeurs de manière très irrégulière. En effet, sur un réseau Ethernet, même commuté, il est nécessaire d'avoir des trames de diffusion (que ce soit en *multicast* ou en *broadcast*) qui créent un volume de trafic très important. De ce fait, les chemins de données évoluent sans cesse, ce qui ne permet pas de garantir de latence maximale compatible avec l'utilisation pour la commande de systèmes automatisés.

D'un autre côté, utiliser Ethernet rend possible l'ouverture des systèmes de production sur le monde de l'internet. Isoler physiquement le réseau de commande interdirait cette ouverture, ce qui est dommageable. Nous avons choisi d'isoler logiciellement notre application à l'intérieur d'un VLAN. Un VLAN (*Virtual Local Area Network* ou *Virtual LAN*, et en français *Réseau Local Virtuel*) permet la réalisation de sous-réseaux indépendants, isolés les uns des autres de manière logique (logicielle) et non physique, sans contrainte géographique. Cela permet une plus grande segmentation du réseau, basée sur des critères de ports, d'adresse MAC, etc. Les VLAN sont définis par les normes IEEE 802.1D, 802.1p, 802.1Q et 802.10.

Grâce à l'utilisation d'un VLAN englobant notre système de production, nous avons un trafic maitrisé entre les automates et les quelques ordinateurs servant à la commande du système. Toutefois, nous avons également la possibilité d'ouvrir notre système aux applications internet, telles que la commande ou la surveillance à distance de nos équipements. De plus, cette technologie permet l'ajout d'une communication spécifique entre les différents réseaux d'entreprise, par exemple entre les logiciels dédiés à l'horizon stratégique et le MES.

IV.7 Le module d'aide à la décision

Le déroulement d'une simulation en ligne dans notre application est décomposable en 10 étapes :

1. L'opérateur entre ou modifie le paramétrage d'un nouvel ordre de fabrication, dont fait partie le paramétrage de la durée de décision ;
2. À la sauvegarde de cet ordre dans la base de donnée, le MES détecte une modification dans la table des ordres ;
3. Le MES vérifie que la durée de prise de décision est compatible avec les durées de simulation. Dans le cas contraire, celui-ci bloque le processus et affiche un message d'erreur en attente d'une modification de l'opérateur ;
4. Le MES bloque la modification des ordres de fabrication de la table puis ordonne au simulateur une simulation démarrant à la date actuelle avec un horizon courant jusqu'à complétion de tous les ordres présents dans la table ;
5. Le simulateur demande l'état actuel de l'atelier à l'observateur ;
6. L'observateur stocke son état dans des fichiers de données et renvoie un acquittement au simulateur ;
7. Le simulateur s'initialise à partir des données des fichiers et démarre la simulation ;

8. Pendant la simulation, la date de fin de chaque ordre de fabrication est enregistrée dans une table de la base de données ;
9. À la fin de la simulation, la simulation envoie un acquittement au MES ;
10. Le MES affiche les données de la base de données dans la fenêtre de paramétrage des ordres et autorise de nouveau la modification de la table des ordres.

Les étapes 2 à 10 sont totalement automatisées et ne sont que les phases de calcul du scénario entré par l'opérateur. De ce fait, elles sont totalement invisibles pour l'utilisateur final, qui n'a que l'utilitaire de supervision du MES à manipuler. Cela permet de mettre la simulation en ligne à disposition de tous les acteurs de la production, spécialistes ou non de la simulation.

De plus, les essais que nous avons réalisés sur la chaine flexible permettent de réaliser ces étapes en environ 5 secondes, décomposables en :

1. Communication MES – Simulateur : durée négligeable ;
2. Récupération des données de l'état de l'observateur et des données de production sur la base de données : 1 seconde environ ;
3. Initialisation et simulation : 4 secondes environ ;
4. Communication des résultats au MES : durée négligeable.

Ces résultats nous permettent d'envisager une utilisation suffisamment ergonomique pour être d'ores et déjà utilisables. Pour une utilisation industrielle, le modèle de simulation peut de plus être optimisé, ce qui réduirait encore les temps de calcul.

IV.8 Conclusion

Ce chapitre nous a permis de présenter l'application de simulation en ligne que nous avons développée. Après avoir présenté la chaine flexible d'assemblage qui nous a servi de support, nous avons pu détailler les modélisations successives de chacun des modèles utilisés, puis leur intégration dans l'architecture de commande du système.

L'application que nous avons développée concerne l'aide au gestionnaire de production dans le lancement d'un nouvel ordre de production sur le système. Le module d'aide à la décision permet de calculer les dates de fin estimées de chaque ordre déjà en production et de celui que le gestionnaire veut tester. Ainsi, celui-ci peut étudier l'influence du paramétrage de cet ordre sur le comportement global du système.

Cette application nous a permis de valider l'applicabilité des concepts que nous avons développés dans les chapitres précédents, autant sur l'aspect technique que sur l'aspect d'ergonomie d'utilisation au vu des durées mises en jeu.

Conclusion et perspectives

Les travaux présentés dans cette thèse ont permis de proposer une architecture permettant l'utilisation de la simulation pour l'aide à la décision concernant le pilotage d'un système de production complexe. Cette utilisation était depuis longtemps envisagée au vu des possibilités offertes, mais les différents auteurs ayant essayé de développer des applications complètes d'aide à la décision incluant la simulation se sont heurtés à de nombreux problèmes empêchant généralement l'implantation de leur solution.

La seconde partie de ce document traite de la résolution des deux problèmes qui nous ont semblés majeurs au vu de la littérature : la place relative de l'homme et de la machine dans une prise de décision assistée par la simulation et la liaison existant entre la simulation et le système réel. Cette résolution s'appuie sur le cas général des systèmes de production complexe, et un effort particulier a été fait pour que les solutions proposées soient applicables à la majorité de ces systèmes. Ainsi, nous avons appliqué à cette classe de systèmes les théories d'ingénierie et de psychologie cognitives permettant de dissocier les tâches réalisées par l'humain et par la machine dans une démarche de simulation en ligne. Cette dissociation effectuée, nous avons ensuite proposé différentes répartitions des tâches nous semblant convenir à l'application souhaitée, ainsi que leurs conditions d'application.

Nous avons ensuite proposé une expérimentation simple permettant de montrer la nécessité d'un lien entre la simulation et le système de production. Ce lien est nécessaire au niveau de l'initialisation de la simulation, qu'il est préférable de réaliser avec l'état actuel du système pour garantir une précision suffisante sur les résultats de simulation. La question est alors posée quant à la réalisation de ce lien, posant de nombreux problèmes tant au niveau technique que théorique.

Nous avons à cette fin proposé trois alternatives ayant chacune leur domaine d'application :

- une initialisation directe avec les données disponibles par les capteurs et détecteurs utilisés pour la commande du système. Celle solution est réservée aux applications où ceux-ci sont suffisamment nombreux et performants. Nous avons mis en évidence une limitation importante de cette solution : les informations nécessaires à l'initialisation de notre simulation ne sont pas toujours présentes au niveau du système de commande.
- une initialisation à partir d'un simulateur temps-réel tournant en parallèle du système, principalement orientée vers les systèmes à cycles régénératifs. En effet, on observe un écart divergent à long terme entre un simulateur temps réel fonctionnant

en « boucle ouverte » et le système de production. Le fait d'avoir un système à cycles régénératifs permet de repartir régulièrement d'un état identique entre le système et le simulateur. Notons que, pour être efficace, le système doit être très bien modélisé et la fréquence d'occurrence d'aléas doit être relativement faible ;

- Au vu des inconvénients des deux solutions précédentes, nous avons proposé une solution novatrice : une initialisation à partir d'un observateur construit sur la base d'un simulateur temps réel, en communication constante avec le système de production. Cette communication permet de « fermer la boucle » en asservissant notre simulateur au système réel.

Cette dernière solution est plus difficile à mettre en place, mais permet de traiter l'ensemble des systèmes ne rentrant pas dans les cas d'application cités précédemment.

Toutes ces propositions ont été appliquées à un système flexible de production. Le module d'aide à la décision intégrant la simulation en ligne était destiné à aider le gestionnaire de production dans le paramétrage des ordres de fabrication avant leur lancement. Celui-ci permet de calculer une date de fin estimée des différents ordres en attente de lancement ou déjà lancés sur le système de production. À partir de cet indicateur, le gestionnaire de production humain peut modifier le nombre de transporteurs alloués à l'ordre pour respecter au mieux les dates de livraison promises.

La conclusion que nous pouvons tirer de ces travaux est qu'il est désormais possible d'utiliser la simulation en ligne pour des systèmes en production. Si son installation n'est pas triviale, les bénéfices entrevus nous semblent suffisamment conséquents pour que certaines industries puissent d'ores et déjà s'y intéresser.

Deux limitations sont toutefois à signaler. Tout d'abord, l'installation d'un tel module d'aide à la décision est relativement coûteuse, que ce soit en temps ou en argent. Son application est donc principalement destinée à des systèmes de taille importante et dont le pilotage revêt une importance et une complexité particulière.

De plus, du fait des temps de communication entre le système réel et notre observateur, nous avons pu voir que les solutions techniques que nous avons proposées ne permettent pas, pour l'instant, de mettre en place un observateur dans le cas général, notamment dans le cas des systèmes à très haute cadence.

Au terme de cette thèse, nous pouvons envisager une suite de ces travaux dans plusieurs directions importantes.

L'application proposée lors du Chapitre IV est destinée à instaurer une collaboration entre l'humain et la machine pour la prise de décision. Nous avons montré lors du Chapitre II qu'il existait une possibilité d'exclure, sous certaines conditions, l'humain de la boucle de décision. Une des directions futures de nos travaux sera tout d'abord d'implémenter une simulation totalement automatisée, de la détection du besoin de simulation à son application effective, sur le système. Cette problématique pose des questions différentes, principalement au niveau des durées de calcul et de décision mises en jeu. En effet, le système de production est généralement localement bloqué dans son évolution avec une telle approche. De ce fait, il convient de diminuer le plus possible la durée nécessaire aux simulations (communications, calcul et décisions) au risque de perturber fortement le

fonctionnement du système. Une piste serait d'utiliser des moyens de calculs parallèles pour accélérer les simulations.

Dans l'approche couplant l'humain et la machine, nous avons utilisé uniquement la simulation pour aider le pilote dans sa prise de décision. Lorsque la décision devient complexe, il devient nécessaire d'associer des outils d'optimisation à la simulation en ligne pour aider le pilote dans sa recherche de solutions. À cette fin, nous nous proposons de mener une recherche pour définir les classes d'outils d'optimisations utilisables avec la simulation en ligne. Le critère de choix principal que nous retenons est le problème du temps limité pour la prise de décision, les outils d'optimisation couplés à la simulation étant souvent très gourmands en temps de calcul. Nous allons notamment étudier le comportement de la solution que fournit l'outil d'optimisation lorsque le critère d'arrêt est un temps maximum de calcul, et non un critère plus classique de convergence sur une fonction objectif par exemple.

Ensuite, nous nous sommes focalisés dans ces travaux sur le cas des systèmes à évènements discrets. Pour élargir encore le domaine d'application de la simulation en ligne, nous pensons poursuivre ces travaux en étendant les concepts que nous avons proposé au domaine du continu. En effet, il est fréquent d'avoir des systèmes de production hybrides, au sens continu/discret, et il nous semble important de pouvoir les prendre en compte. L'exemple d'application que nous avons présenté dans cette thèse pourra continuer à servir de base expérimentale pour cette extension aux systèmes hybrides. En effet, si nous étendons le cadre de notre étude à l'ensemble de l'installation présentée Figure 34, nous avons rapidement un système devant être modélisé par un modèle combinant discret et continu. Si nous prenons en compte par exemple le robot cartésien servant d'interface entre le magasin dynamique et le poste 1, la position de ce robot, modélisée par des variables continues, devra être prise en compte dans la simulation en ligne, et donc dans l'observateur.

La suite de cette étude sera probablement une application de la simulation en ligne en milieu industriel. En effet, l'application que nous avons développée dans ces travaux est à caractère expérimental, ce qui est susceptible de masquer certains problèmes que nous n'avons donc pas pu encore rencontrés. Il nous semble également intéressant non seulement de voir comment la simulation peut s'insérer dans le cadre d'une production établie, mais également d'étudier l'efficacité de la coopération humain/simulation lors de décisions à prendre en une durée limitée. Il en découlera probablement des indications sur l'acceptation de cette nouvelle approche par le milieu industriel.

La notion d'observateur que nous avons proposé dans cette thèse est très riche, et présente sans doute d'autres applications que celles que nous avons proposées dans ce document. En effet, tel que nous l'avons défini jusqu'à présent, l'observateur ne sert qu'à fournir son état sur demande au simulateur. Or, une particularité de cet observateur est d'avoir la possibilité de comparer en temps-réel le comportement réel du système de production et son comportement prévu par la modélisation contenue dans l'observateur. À partir de cette constatation, nous pensons qu'il serait intéressant de faire un suivi de l'écart entre le comportement réel et le comportement simulé. Ce suivi peut par exemple entrer dans les méthodes de maintenance préventive, grâce à la détection de déviance comportementale, ou dans la détection de dysfonctionnement dans le cadre de la maintenance réactive.

Un autre vaste domaine dans lequel s'inscrivent nos perspectives concerne la construction des modèles d'observation et de simulation en ligne. Dans nos travaux, nous avons entièrement reconstruit ces deux modèles de simulation alors que nous possédions déjà un modèle, réalisé lors de la conception de la ligne. Notre réflexion aujourd'hui serait de voir comment un modèle destiné à la conception initiale d'un système pourrait, avec le moins de modifications possible, devenir un modèle d'observation ou un modèle de simulation en ligne pour l'aide à la décision. L'objectif ultime serait d'imaginer un module qui, ajouté à n'importe quel modèle, permettrait à la fois la synchronisation avec un système réel et la sauvegarde de l'état du simulateur, transformant celui-ci en observateur. De même, un autre module permettrait de démarrer notre simulation à partir d'un état non vide, transformant notre simulateur de conception en simulateur en ligne pour la prise de décision. Remarquons que cette perspective demandera de rentrer encore plus dans le fonctionnement interne de l'outil de simulation, entrainant une collaboration indispensable avec les concepteurs d'outils, ou remettant en cause l'utilisation de COTS.

Bibliographie

[Banks, 1996] Banks, Jerry. «Software for simulation.» *Winter Simulation Conference.* Piscataway, 1996. pp. 31-38.

[Banks *et al.*, 1996] Banks, Jerry, John S. Carson II, et Barry L. Nelson. *Discrete-event system simulation.* Second Edition. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1996.

[Barkmeyer *et al.*, 1999] Barkmeyer, Edward, Peter Denno, Shaw Feng, Evan Wallace, et Albert Jones. «NIST Response to MES Request for Information.» NISTIR 6397, National Institute of Standards and Technology, Gaithersburg, MD, 1999.

[Berchet, 2000] Berchet, Claire. «Modélisation pour la simulation d'un système d'aide au pilotage industriel.» Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, 2000.

[Blanc, 2006] Blanc, Pascal. «Pilotage par approche holonique d'un système de production de vitres de sécurité feuilletées.» Thèse de doctorat, Ecole Centrale de Nantes et Université de Nantes, 2006.

[Boer et Verbraeck, 2003] Boer, Csaba Attila, et Alexander Verbraeck. «Distributed simulation and manufacturing: distributed simulation with cots simulation packages.» *Proceedings of the 35th conference on Winter simulation: driving innovation.* New Orleans, Louisiana: Winter Simulation Conference, 2003. pp. 829 - 837.

[Cardin et Castagna, 2005] Cardin, Olivier, et Pierre Castagna. «Defining a command architecture enabling proactive simulation on a complex manufacturing system.» *Conceptual Modeling and Simulation Conference 2005.* Marseille, France, 2005. pp. 205-210.

[Cardin et Castagna, 2006a] Cardin, Olivier, et Pierre Castagna. «Handling uncertainty in production activity control.» *12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006.* Saint-Etienne, France, 2006.

[Cardin et Castagna, 2006b] Cardin, Olivier, et Pierre Castagna. «Utilisation de la simulation proactive: une aide au pilotage des systèmes de production.» *6e Conférence Francophone de MOdélisation et SIMulation - MOSIM'06 .* Rabat, Maroc, 2006.

[Cardin et Castagna, 2008] Cardin, Olivier, et Pierre Castagna. «Proactive production activity control by online simulation.» *International Journal of Simulation and Process Modeling* [Article à paraître], 2008.

[Castagna *et al.*, 2001] Castagna, Pierre, Nasser Mebarki, et Roland Gauduel. «La simulation, un outil d'aide au pilotage des systèmes de production. Exemples d'application.» *MOSIM'01.* Troyes, France, 2001.

[Cauvin, 2005] Cauvin, Aline. «Analyse, modélisation et amélioration de la réactivité des systèmes de décision dans les organisations industrielles: vers une aide à la conduite des processus d'entreprise dans un contexte perturbé.» Habilitation à diriger les recherches, Université Paul Cézanne, Aix-Marseille III, 2005.

[Cegarra et Hoc, 2006] Cegarra, Julien, et Jean-Michel Hoc. «Cognitive styles as an explanation of expert's individual differences: A case study in computer-assisted troubleshooting diagnosis.» *International Journal of Human-Computer Studies* Vol. 64 [2006]: pp. 123-136.

[Chové, 2007] Chové, Etienne. «Observateur par simulation sous Quest.» Thèse de Master, École Centrale de Nantes, 2007.

[Davis, 1998] Davis, Wayne J. «Online simulation: Need and evolving research requirements.» Dans *Handbook of Simulation*, de Jerry Banks, 465-516. John Wiley and Sons Inc., 1998.

[Davis, 1999] Davis, Wayne J. «Simulation: Technologies in the new millenium.» *31st conference on Winter simulation: Simulation - a bridge to the future.* Phoenix: ACM Press, 1999. pp. 141-147.

[De Vin et Jägstam, 2001] De Vin, Leo J., et Mats Jägstam. «Why we need to offer a modeling and simulation engineering curriculum.» *33rd conference on Winter simulation.* Arlington, Virginia: IEEE Computer Society, 2001. pp. 1599 - 1604.

[Doumeingts *et al.*, 1995] Doumeingts, Guy, Bruno Vallespir, et David Chen. «Methodologies for designing CIM systems: A survey.» *Computers in Industry* Vol. 25 [1995]: pp. 263-280.

[Draghici *et al.*, 1998] Draghici, George, Nicolae Brinzei, et Ioana Filipas. «La modélisation et la simulation en vue de la conduite des systèmes de production.» *Les cahiers des enseignements francophones en Roumanie*, 1998.

[Drake et Smith, 1996] Drake, Glenn R., et Jeffrey S. Smith. «Simulation system for real-time planning, scheduling, and control.» *28th conference on Winter simulation.* Coronado: ACM Press, 1996. pp. 1083 - 1090.

[Endsley et Kaber, 1999] Endsley, Mica R., et David B. Kaber. «Level of automation effects on performance, situation awareness and workload in dynamic control task.» *Ergonomics* Vol. 42, n° 3 [1999]: pp. 462-492.

[Fegan *et al.*, 1991] Fegan, Joseph M., Gearoid M. Lane, et Paul J. Nolan. «Introduction to simulation using Intelligent Simulation Interface (ISI).» *Procceding of the 23rd conference on Winter simulation.* Phoenix, Arizona: IEEE Computer Society, 1991. pp. 143-147.

[Fontanili, 1999] Fontanili, Franck. «Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone.» Thèse de doctorat, Université Paris XIII, 1999.

[Fowler et Rose, 2004] Fowler, John W., et Oliver Rose. «Grand Challenges in modeling and simulation of complex manufacturing systems.» *Simulation* Vol. 80 [2004]: pp. 469-476.

[García *et al.*, 2003] García, Andrés, Duncan McFarlane, Martyn Fletcher, et Alan Thorne. «Auto-ID in materials handling.» *Auto-ID Centre White Paper*, 1 February 2003.

[Gonzalez et Davis, 1997] Gonzalez, Fernando G., et Wayne J. Davis. «A simulation-based controller for distributed discrete-event systems with application to flexible manufacturing.» *29th conference on Winter simulation.* Atlanta: ACM Press, 1997. pp. 845 - 852.

[Gonzalez et Davis, 1998] Gonzalez, Fernando G., et Wayne J. Davis. «Initializing on-line simulations from the state of a distributed system.» *30th conference on Winter simulation.* Washington, D.C.: IEEE Computer Society Press, 1998. pp. 507-513.

[Gupta et Sivakumar, 2005] Gupta, Amit Kumar, et Appa Iyer Sivakumar. «Conjunctive simulated scheduling.» *International Journal of Advanced Manufacturing Technologies* Vol. 26 [2005]: pp. 1409-1413.

[Hanisch *et al.*, 2003] Hanisch, André, Juri Tolujew, Klaus Richter, et Thomas Schulze. «Modeling people flow: online simulation of pedestrian flow in public buildings.» *35th conference on Winter simulation: driving innovation.* New Orleans: Winter Simulation Conference, 2003. pp. 1635 - 1641.

[Hanisch *et al.*, 2005] Hanisch, André, Juri Tolujew, et Thomas Schulze. «Initialization of online simulation models.» *37th conference on Winter simulation.* Orlando, Florida: Winter Simulation Conference, 2005. pp. 1795-1803.

[Hoc, 1990] Hoc, Jean-Michel. «Les activités du diagnostic.» Dans *Traité de psychologie cognitive. Tome II: Le traitement de l'information symbolique.*, de J.-F. Richard, C. Bonnet et R. Ghiglione, 158-165. Paris: Dunod, 1990.

[Hoc, 1996] Hoc, Jean-Michel. *Supervision et contrôle de processus, la cognition en situation dynamique.* Presses Universitaires de Grenoble, 1996.

[Hoc, 2001] Hoc, Jean-Michel. «Towards a cognitive approach to human-machine cooperation in dynamic situations.» *International Journal of Human-Computer Studies* Vol. 54 [2001]: pp. 509-540.

[Hollingworth et Spackman, 2007] Hollingworth, William, et D. Eldon Spackman. «Emerging methods in economic modeling of imaging costs and outcomes: a short report on Discrete Event Simulation.» *Academic Radiology* Vol. 14, n° 4 [2007]: pp. 406-410.

[Hotz *et al.*, 2006] Hotz, Ingo, André Hanisch, et Thomas Schulze. «Simulation-based early warning systems as a practical approach for the automotive industry.» *37th conference on Winter simulation.* Monterey, California: Winter Simulation Conference, 2006. pp. 1962 - 1970.

[Howard *et al.*, 1992] Howard, Randall B., Mark A. Gallagher, Kenneth W. Bauer, et Peter S. Maybeck. «Confidence intervals for univariate discrete-event simulation output using the Kalman filter.» *24th conference on Winter simulation.* Arlington: ACM Press, 1992. pp. 586 - 593.

[Huang, 2002] Huang, Chin-Yin. «Distributed manufacturing execution systems: A workflow perspective.» *Journal of Intelligent Manufacturing* Vol. 13 [2002]: pp. 485-497.

[Hugan, 1994] Hugan, Joseph C. «Quest - Queuing Event Simulation Tool.» *26th conference on Winter simulation.* Orlando, Florida, 1994. pp. 458-463.

[Iassinovski *et al.*, 2007] Iassinovski, Serguei, Abdelhakim Artiba, et Christophe Fagnart. «A generic production rules-based system for on-line simulation, decision making and discrete process control.» *International Journal of Production Economics*, 2007.

[Jaklic *et al.*, 2007] Jaklic, Anton, Franci Vode, et Tomaz Kolenko. «Online simulation model of the slab-reheating process in a pusher-type furnace.» *Applied Thermal Engineering* Vol. 27 [2007]: pp. 1105-1114.

[Kaber et Endsley, 2004] Kaber, David B., et Mica R. Endsley. «The effects of level of automation and adaptative automation on human performance, situation awareness and workload in a dynamic control task.» *Theoretical Issues in Ergonomics Science* Vol. 5, n° 2 [2004]: p. 113 – 153.

[Kelton *et al.*, 2004] Kelton, W. David, Randall P. Sadowski, et David T. Sturrock. *Simulation with Arena.* 3e édition. New York: McGraw-Hill, 2004.

[Kim et Gibson Jr, 2003] Kim, K. J., et G. Edward Gibson Jr. «Interactive simulation modeling for heavy construction operations.» *Automation in Construction* Vol. 12 [2003]: pp. 97-109.

[Klingstam et Gullander, 1999] Klingstam, Pär, et Per Gullander. «Overview of simulation tools for computer-aided production engineering.» *Computers in Industry* Vol. 38, n° 2 [1999]: pp. 173-186.

[Koubâa et Song, 2002] Koubâa, Anis, et Ye-Qiong Song. «Evaluation de performances d'Ethernet commuté pour des applications temps réel.» *Real Time Systems'2002.* Teknea, 2002.

[Kouiss et Pierreval, 1999] Kouiss, Khalid, et Henri Pierreval. «Implementing an on-line simulation in a flexible manufacturing system.» *ESS'99 conference.* Nuremberg, 1999. pp. 484-488.

[Kouiss et Najid, 2004] Kouiss, Khalid, et Najib M. Najid. «Un couplage MES-simulation pour le pilotage d'un système de production.» *5e Conférence Francophone de MOdélisation et SIMulation "Modélisation et simulation pour l'analyse et l'optimisation des systèmes industriels et logistiques" MOSIM'04.* Nantes, France, 2004.

[Kumar et Seidman, 1990] Kumar, P. R., et T. I. Seidman. «ynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems.» *IEEE Transactions on Automatic Control* Vol. 35, n° 3 [1990]: pp. 289-298.

[Law et Kelton, 1982] Law, Averill M., et W. David Kelton. *Simulation modeling and analysis.* New York: McGraw-Hill Book Company, 1982.

[Law et Kelton, 1991] Law, Averill M., et W. David Kelton. *Simulation modeling and analysis.* New York: McGraw-Hill Book Company, 1991.

[Le Moigne, 1990] Le Moigne, Jean-Louis. *La modélisation des systèmes complexes.* Paris: Bordas, 1990.

[Lenclud, 1993] Lenclud, Thierry. «Contribution à la conception d'un système intégré de simulation des systèmes de production.» Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambresis, 1993.

[McFarlane *et al.*, 2002] McFarlane, Duncan, James Carr, Mark Harrison, et Andrew McDonald. «Auto-ID's three R's: Rules and Recipes for product Requirements.» *Auto-ID Centre White Paper*, 1 Novembre 2002.

[Mebarki *et al.*, 1998] Mebarki, Nasser, Alain Duchaussoy, et Henri Pierreval. «On the Comparison of Solutions in Stochastic Simulation-Optimization Problems with Several Performance Measures.» *International Transactions in Operational Research* Vol. 5, n° 2 [1998]: pp. 137-145.

[Merlaud *et al.*, 1995] Merlaud, Christian, Jacques Perrin, et Jean-Paul Trichard. *Automatique Informatique Industrielle.* Paris: Dunod, 1995.

[Mesarovic *et al.*, 1970] Mesarovic, Mihajlo D., Dušan Macko, et Yasuhiko Takaha. *Theory of Hierarchical, Multilevel, Systems.* New York: Academic Press, 1970.

[Millot, 2007] Millot, Patrick. «Toward Human-Machine Cooperation.» *4th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2007.* Angers, France, 2007. pp. 1-16.

[Morel *et al.*, 2003] Morel, Gérard, Hervé Panetto, Marek B. Zaremba, et Frédérique Mayer. «Manufacturing enterprise control and management system engineering: paradigms and open issues.» *IFAC Annual Review in Control* Vol. 27 [2003]: pp. 199-209.

[Murata, 1989] Murata, Tadao. «Petri nets: Properties, analysis and applications.» *Proceedings of the IEEE* Vol. 77, n° 4 [1989]: pp. 541-580.

[Pannequin, 2007] Pannequin, Rémi. «Proposition d'un environnement de modélisation et de test d'architectures de pilotage par le produit de systèmes de production.» Thèse de doctorat, Université Henri Poincaré de Nancy, 2007.

[Pannequin et Thomas, 2006a] Pannequin, Rémi, et André Thomas. «Proposition d'une plateforme d'expérimentation sur le contrôle par le produit des flux de production.» *MOSIM 2006: Modélisation, optimisation et simulation des systèmes; Défis et opportunités.* Rabat, Maroc: Lavoisier, 2006.

[Pannequin et Thomas, 2006b] Pannequin, Rémi, et André Thomas. «Cooperation between business and holonic manufacturing decision systems.» *12th IFAC Symposium on Information Control Problems in Manufacturing - INCOM'2006.* Saint-Étienne, France, 2006.

[Peters et Smith, 1998] Peters, Brett A., et Jeffrey S. Smith. «Real-time, simulation-based shop floor control.» *ArenaSphere'98.* Pittsburgh, 1998. pp. 188-194.

[Pinot *et al.*, 2007] Pinot, Guillaume, Olivier Cardin, et Nasser Mebarki. «A study on the group sequencing method in regards with transportation in an industrial FMS.» *IEEE International Conference on Systems, Man and Cybernetics.* Montréal, 2007.

[Pooch et Wall, 1993] Pooch, Udo W., et James A. Wall. *Discrete Event Simulation: a Practical Approach.* Boca Raton, Florida: CRC Press, Inc., 1993.

[Pujo *et al.*, 2004] Pujo, Patrick, Massimo Pedetti, et Fouzia Ounnar. «Pilotage proactif des lignes de production kanban par modélisation DEVS et simulation temps-réel.» *5e Conférence Francophone de MOdélisation et SIMulation "Modélisation et simulation pour l'analyse et l'optimisation des systèmes industriels et logistiques", MOSIM'04.* Nantes, France, 2004. pp. 593-600.

[Ramakrishnan *et al.*, 2002] Ramakrishnan, Sreeram, Seungyub Lee, et Richard A. Wysk. «Real-time control: implementation of a simulation-based control architecture for supply chain interactions.» *34th conference on Winter simulation: exploring new frontiers.* San Diego: Winter Simulation Conference, 2002. pp. 1667 - 1674.

[Rasmussen, 1983] Rasmussen, Jens. «Skills, rules and knowledge; Signals, signs and symbols and other distinctions in humn performance models.» *IEEE Transactions on Systems Man and Cybernetics* Vol. SMC-13, n° 3 [1983]: pp. 257-266.

[Rasmussen, 1986] Rasmussen, Jens. «Information processing and human-machine interaction; An approach to cognitive engineering.» Dans *System Science and engineering*, de P. Sage. Elsevier, 1986.

[Rogers et Flanagan, 1991] Rogers, P., et M.T. Flanagan. «On-line simulation for real-time scheduling of manufacturing systems.» *Industrial Engineering* Vol. 23 [1991]: pp. 37-40.

[Roy, 1989] Roy, Bernard. «Main sources of inaccurate determination, uncertainty and imprecision in decision models.» *Mathematical Computer Modelling* Vol. 12, n° 10/11 [1989]: pp. 1245-1254.

[Santos *et al.*, 2005] Santos, Raùl Alves, Julio E. Normey-Rico, Alejandro Merino Gomez, Luis Felipe Acebes Arconada, et César de Prada Moraga. «OPC based distributed real time simulation of complex continuous process.» *Simulation Modelling Practice and Theory* Vol. 13 [2005]: p. 525–549.

[Schriber et Brunner, 1994] Schriber, Thomas J., et Daniel T. Brunner. «Inside simulation software: how it works and why it matters.» *26th conference on Winter simulation.* Orlando, Florida: Society for Computer Simulation International, 1994. pp. 45 - 54.

[Sheridan, 1984] Sheridan, T. B. *Supervisory control of remote manipulators, vehicules and dynamic processes: experiments in command and display aiding.* Vol. 1, chez *Advances in Man-Machines systems researchs*. 1984.

[Swisher *et al.*, 2003] Swisher, James R., Sheldon H. Jacobson, et Enver Yücesan. «Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey.» *ACM Transactions on Modeling and Computer Simulation (TOMACS)* Vol. 13, n° 2 [2003]: pp. 134 - 154.

[Tautou-Guillaume, 1997] Tautou-Guillaume, Laure. «Contribution des méthodes évolutionnistes à l'optimisation via la simulation des systèmes de production.» Thèse de doctorat, Université de Clermont-Ferrand 2, 1997.

[Taylor III, 1996] Taylor III, Bernard W. *Introduction to Management Science.* 5e édition. Upper Saddle River, New Jersey: Prentice Hall, 1996.

[Trentesaux, 2002] Trentesaux, Damien. «Pilotage hétérarchique des systèmes de production.» Thèse d'HDR, Université de Valenciennes et du Hainaut-Cambrésis (UVHC), Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH), 2002.

[Tweedale *et al.*, 2007] Tweedale, Jeffrey, Nikhil Ichalkaranje, Christos Sioutis, B. Jarvis, Angela Consoli, et G. Phillips-Wren. «Innovations in multi-agent systems.» *Computer Applications* Vol. 30 [2007]: p. 1089–1115.

[Valckenaers, 2001] Valckenaers, Paul. «Special issue on holonic manufacturing systems.» *Computers in industry* Vol. 46 [2001]: pp. 233-331.

[Van Brussel *et al.*, 1998] Van Brussel, Hendrik, Jo Wyns, Paul Valckenaers, Luc Bongaerts, et Patrick Peeters. «Reference architecture for holonic manufacturing systems: PROSA.» *Computers in Industry* Vol. 37 [1998]: pp. 255-274.

[Van Volsem *et al.*, 2007] Van Volsem, Sofie, Wout Dullaert, et Hendrik Van Landeghem. «An Evolutionary Algorithm and discrete event simulation for optimizing inspection strategies for multi-stage processes.» *European Journal of Operational Research*, 2007: pp. 621-633.

[Vollmann *et al.*, 1988] Vollmann, Thomas E., William Lee Berry, et D. Clay Whybark. *Manufacturing planning and control systems.* Second edition. Homewood, Illinois: Business One Irwin, 1988.

[Wong *et al.*, 2002] Wong, Chien Yaw, Duncan McFarlane, Ahmad A. Zaharudin, et V. Agarwal. «The Intelligent Product Driven Supply Chain.» *IEEE Systems Man and Cybernetics.* Hammammet, Tunisia, 2002.

[Wu et Wysk, 1989] Wu, Szu-Yung David, et Richard A. Wysk. «An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing.» *International Journal of Production Research* Vol. 27, n° 9 [1989]: pp. 1603-1623.

[Zamarreño *et al.*, 2000] Zamarreño, Jesús M., Felipe Acebes, et Raúl Alvés. «OPC-based real time simulator: architecture and practical example.» *41st SIMS Simulation Conference.* Copenhague, Danemark, 2000.

APPORT DE LA SIMULATION EN LIGNE DANS L'AIDE À LA DÉCISION POUR LE PILOTAGE DES SYSTÈMES DE PRODUCTION – APPLICATION À UN SYSTÈME FLEXIBLE DE PRODUCTION

RÉSUMÉ

Cette thèse porte sur l'apport de la simulation en ligne dans le pilotage des systèmes de production. Au fil des ans, ces systèmes deviennent de plus en plus compliqués et requièrent une capacité d'analyse toujours plus grande de la part de son pilote. Plusieurs études ont démontré que l'aide la plus précieuse qu'une entité décisionnelle pouvait obtenir pour optimiser ses décisions était un outil lui permettant de prévoir la conséquence de ses décisions. En effet, c'est cet aspect de la prise de décision qui est rendu particulièrement hasardeux par l'augmentation de la complexité des systèmes. L'objet de cette thèse est d'étudier comment la simulation de flux, après avoir été un outil d'aide à la conception d'unité de production, peut devenir un outil d'aide à la décision dans le cadre du pilotage de cette unité.

La première partie présente le contexte de nos travaux. Après avoir resitué le pilotage des systèmes de production, nous présentons les principes et les outils de la simulation du flux. Enfin, nous présentons un état de l'art concernant la simulation en ligne, en remarquant qu'il s'agit d'un domaine encore relativement peu exploré par la communauté de la recherche sur les systèmes de production. La seconde partie de nos travaux portent sur l'intégration des outils de simulation dans l'architecture de commande. Dans un premier temps, nous étudions la place relative de l'humain et de la machine au niveau de la prise de décision. Les outils de simulation classique permettent depuis longtemps déjà de prévoir la conséquence de décisions portant sur un terme long relativement à la dynamique du système. Lorsque l'horizon se restreint, l'état initial des simulations est problématique : de par son influence grandissante sur les résultats, il est nécessaire d'assurer une équivalence de plus en plus stricte avec l'état du système de production au moment du début des simulations. C'est pourquoi, dans la deuxième partie de ce chapitre, nous montrons sur un exemple l'importance de l'initialisation dans une simulation en ligne pour la prise de décision.

La troisième partie de nos travaux porte sur la résolution de ce problème par l'introduction dans l'architecture de commande d'un observateur, également réalisé par simulation, en communication constante avec les organes de commande du système. Au moment où la simulation se lance, elle s'initialise sur l'état de l'observateur, qui est considéré comme étant une bonne approximation de l'état actuel réel du système. La dernière partie présente une validation sous la forme d'une application complète des concepts et de l'architecture proposés sur un système de production flexible, actuellement en production dans le cadre de l'Institut Universitaire de Technologie de Nantes.

CONTRIBUTION OF ONLINE SIMULATION TO PRODUCTION ACTIVITY CONTROL DECISION SUPPORT APPLICATION TO A FLEXIBLE MANUFACTURING SYSTEM

ABSTRACT

This thesis deals with the contribution of online simulation to production activity control. Along the years; manufacturing systems get more and more complex and require more and more analysis skills from the pilot. Several studies showed that the most precious help that a decisional entity could get to optimise its decisions was a tool enabling to foresee the consequences of these decisions. Indeed, the rise of systems complexity makes this aspect of decision making particularly tricky. The subject of this thesis is to study how discrete-event simulation, after being used as a support for production units design, may become a decision support tool for production activity control of these units.

First section presents the context of this work. After positioning the manufacturing systems production activity control, we present the discrete-event simulation tools and principles. Finally, we present a state of the art about online simulation, noticing it is a subject relatively poorly explored by the manufacturing systems research community. Second section of our work deals with the integration of simulation tools in the control architecture. Our first step is to study the relative position of human and machine in the decision making process. For a long time, classical simulation tools have enabled to foresee the consequences of decisions acting on a term considered long relatively to the system dynamic. When the horizon shortens, the initial state of simulations becomes a problem: because of its growing influence on the results, it is necessary to insure a more and more strict equivalence with the state of the system at the simulations starting date. This is why, in the second part of this section, we show on an example the importance of initialisation in an online simulation for decision making.

Third section of this work is about the resolution of this last problem with the insertion of a simulation-made state estimator in the control architecture, constantly communicating with the system control. At the simulation start date, it is initialised on the present state of the observer, which is considered as a good approximation of the real present state of the system. Last section presents how we validated what we proposed with a complete application of the concepts and architecture on a flexible manufacturing system, currently running at the Technology University Institute (IUT) of Nantes, France.

MOTS-CLÉS

Simulation en ligne, observateur, pilotage, systèmes de production, aide à la décision, MES, OPC.

KEYWORDS

Online simulation, state estimator, production activity control, manufacturing system, decision support, MES, OPC.

DISCIPLINE : Automatique

N° : ED 366-325