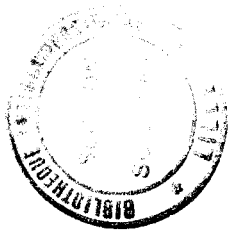


THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR

en Productique : Automatique et Informatique Industrielle

par

Mohammed MOSTEFAI

Ingénieur d'état en Electronique : Contrôle Industriel

UN MODELE D'ARCHITECTURE ORIENTE OBJET POUR LA CONCEPTION DE PLATES-FORMES DE SIMULATION INTERACTIVE.

Soutenue le 1 février 1994 devant le jury :

MM.

P. Vidal	Président	*Professeur à l'USTL*
J.-M. Toulotte	Directeur de recherche	*Professeur à l'USTL*
R. Ikni	Co-Directeur	*Maître de conférences*
J.Defrenne	Rapporteur	*Professeur à l'Université de Valenciennes*
J.P.Frachet	Rapporteur	*Professeur à l'I.S.M.C.M*
Y.David	Invité	*Directeur du C.R.E.S.T.A*

AVANT PROPOS

Le travail présenté dans ce mémoire a été effectué au centre d'Automatique de l'Université des Sciences et Technologies de Lille1.

Je tiens à exprimer ma gratitude à Monsieur le Professeur Pierre VIDAL pour l'accueil qu'il m'a réservé au sein de son laboratoire. Je le remercie vivement d'avoir accepté la présidence de mon jury de thèse.

Toute ma reconnaissance et mes sentiments respectueux vont à Monsieur le Professeur Jean Marc TOULOTTE pour la confiance et le soutien qu'il m'a témoigné tout au long de mes travaux.

Je remercie Rachid IKNI pour l'intérêt qu'il porte à ce travail. Son esprit critique et ses conseils furent toujours accueillis avec joie.

Je remercie également Messieurs : Jean DEFRENNE, Professeur à l'Université de Valenciennes et du Hainaut Cambresis, et Jean Paul FRACHET, Professeur à L'I.S.M.C.M. pour l'honneur qu'ils me font en acceptant de juger ce travail et d'en être rapporteurs.

Mes remerciements vont également à Monsieur Yves DAVID, Directeur du C.R.E.S.T.A. pour l'intérêt qu'il me témoigne en acceptant de participer à mon jury de thèse.

Enfin, je suis très reconnaissant à tous les membres du laboratoire d'Automatique pour l'aide qu'ils m'ont apportée et la sympathie qu'ils m'ont témoignée.

Sommaire

Chapitre I :

Modèle, Simulation et Interactivité

Chapitre II :

Modélisation des Systèmes Industriels

Chapitre III :

Conception des Interfaces Utilisateurs

Chapitre IV :

Le modèle P.Os.T

Chapitre V :

Applications des modèles P.Os.T

Introduction générale

Introduction générale

L'automatisation des systèmes de production rendue nécessaire pour améliorer la productivité des installations industrielles est actuellement affinée par des recherches et développements visant à augmenter leur fiabilité et leur disponibilité.

En effet, étant donné les investissements considérables mis en jeu, il devient de plus en plus nécessaire d'évaluer et d'améliorer, dès la phase de conception, les performances de tels systèmes. La simulation offre la possibilité d'accélérer, dans un environnement non critique, les différentes étapes d'un processus de conception (spécification, configuration, validation, ...).

D'autre part, une fois le système conçu, sa conduite peut s'avérer très délicate et faire appel à des pilotes chevronnés, en mesure d'avoir un comportement adéquat face à toute anomalie ou défaut de fonctionnement. D'ailleurs, dans les systèmes automatisés les opérateurs humains assurent essentiellement des tâches de surveillance et de maintenance curative pour détecter les dysfonctionnements, effectuer un diagnostic et élaborer des procédures de reprise. Compte tenu de l'irrégularité des performances humaines dans ces tâches hautement cognitives, la communication homme-machine joue un rôle primordial pour adapter les interfaces graphiques aux capacités humaines d'acquisition et de traitement des informations et de décision.

La communication homme-machine était, dans le passé et dans le cadre de la simulation, limitée à la visualisation graphique d'un certain nombre de variables de sorties sous forme de courbes ou de représentations alphanumériques.

Le rôle de l'interface homme-machine est avant tout de donner une image réelle du système piloté, donc toutes les informations nécessaires pour une bonne conduite du processus. De nombreux problèmes se posent lors de la conception des vues

graphiques de contrôle et de surveillance proposées à l'opérateur. En effet, les informations affichées doivent dépendre des besoins informationnels de l'opérateur sur les différents contextes du procédé. Ces contextes peuvent par exemple être relatifs à la conduite du processus, à la surveillance et à la détection des dysfonctionnements, au diagnostic et à des procédures de correction ou de reprise de défauts.

Par ailleurs, l'augmentation du nombre et de la puissance des calculateurs permet la hiérarchisation des systèmes de commande des machines de production et facilite leur coordination pour optimiser les procédés de fabrication. Les calculateurs de conduite assurent aux outils de production une autonomie très importante en situation de fonctionnement normal. Mais leur manque d'adaptivité dans les situations imprévues rend indispensable la présence de l'homme pour assurer les tâches de diagnostic et de reprise des dysfonctionnements, qui sont actuellement très difficile à automatiser.

En ce sens, l'utilisation du paradigme objet a permis l'apparition d'une nouvelle génération d'interfaces graphiques homme-machine appelées interfaces événementielles. En effet, à l'aide de cette nouvelle approche de programmation, on est passé des interfaces classiques contrôlées par la machine à des interfaces contrôlées par les actions de l'utilisateur traduites sous forme d'événements. Ce qui permet d'assigner une fonction sécuritaire à l'opérateur pour augmenter la fiabilité et la disponibilité des procédés.

Le travail présenté dans ce mémoire a été réalisé au centre d'Automatique de l'Université des Sciences et Technologies de Lille, sous la direction du Professeur Jean-Marc Toulotte et de Monsieur Rachid Ikni. Il s'inscrit dans la continuité des travaux entrepris depuis plus de six ans au sein de notre équipe ayant permis la définition d'outil de modélisation des processus par lots en vue de la simulation de leur comportement.

Notre contribution apparaît dans la définition d'un modèle d'architecture global permettant la conception des interfaces graphiques homme-machine et la création des plates-formes de simulation interactive. Elle sera présentée dans ce mémoire, structurée en cinq parties :

Le premier chapitre fait un bilan, d'une part des simulateurs et de leurs

caractéristiques et d'autre part des connaissances actuelles sur les techniques de modélisation, de simulation et des styles d'interactivité. Un simulateur ou encore un logiciel de simulation peut être décomposé en trois parties essentielles : un modèle de simulation, un moteur de simulation et une interface graphique.

Lors de sa conception, il convient de choisir pour chacune des trois parties une technique parmi d'autres. En ce sens, une analyse est faite en vue d'aboutir à des simulateurs plus ou moins conviviaux.

Le deuxième chapitre est consacré à la phase de modélisation des systèmes industriels. Cette modélisation est basée sur un travail existant baptisé O.S.L.O (Outil de Simulation en Langage à Objet) qui autorise une décomposition d'une installation industrielle en trois catégories d'objets appelés respectivement : objet technologique, objet opératif et objet mission. Pour cela nous avons proposé un modèle d'objet de simulation unique et générique permettant de représenter les trois types d'objets. A cela s'ajoute la définition des objets liens qui assurent la communication entre les objets de simulation en vue d'avoir un comportement global du système simulé.

Avec le troisième et le quatrième chapitre, on passe à la deuxième partie de l'outil de simulation réservée aux problèmes de conception des interfaces graphiques homme-machine en vue de la simulation interactive. Le troisième chapitre présente les règles de conception des interfaces et la nécessité d'avoir un modèle architectural. Il fait ensuite un bilan général sur les modèles existants en insistant sur l'apport de l'approche objet qui a donné naissance aux interfaces événementielles.

Le quatrième chapitre est dédié à la définition d'un modèle d'architecture baptisé P.Os.T (Présentation, Objet de simulation, Traducteur). Il présente tous les avantages de ce modèle par rapport à ceux décrits dans le chapitre précédent, entre autres, la création rapide des synoptiques et la génération automatique des animations.

Enfin, au cinquième chapitre, la description de trois applications de types différents réalisées selon ce modèle en démontre l'intérêt.

La conclusion dresse un bilan de notre contribution dans le cadre de la simulation interactive et identifie les perspectives nécessaires portant particulièrement sur l'enrichissement de la structure du modèle P.Os.T.

Chapitre I

Modèle, Simulation et

Interactivité

Modèle, Simulation et Interactivité

I.1/ INTRODUCTION

La simulation des systèmes est une technique qui se développe depuis plusieurs années. Les objectifs fixés à cette activité consistent à utiliser des modèles de ces systèmes afin de tester certaines hypothèses sur leur fonctionnement quand il est impossible, indésirable, peu pratique ou antiéconomique de le réaliser en vraie grandeur dans des usines. L'intégration de la simulation dans le cycle de la conception ne peut être efficace que si les conditions suivantes sont remplies :

- la simulation doit être facile à mettre en oeuvre pour pouvoir tester rapidement de nombreuses variantes. Il faut pour cela simplifier au maximum la phase de modélisation qui doit être ramenée à une simple description naturelle;
- il faut pouvoir décrire différents systèmes par analogie avec des systèmes existants dans lesquels seuls quelques éléments diffèrent;
- il faut avoir une bibliothèque de description de sous-systèmes correspondants à des sous-ensembles du commerce;
- il faut pouvoir raffiner progressivement la description au fur et à mesure de l'avancement du projet.

Il n'existe pas actuellement d'outil de simulation présentant ces caractéristiques : les outils actuels sont, soit très conviviaux (simulateurs dédiés ou la modélisation se fait de manière paramétrique avec un modèle d'atelier fixé), soit très flexibles (langage

de simulation demandant un apprentissage assez long et une modélisation nouvelle pour chaque cas), mais rarement les deux. Il était aussi indispensable, pour l'usage de l'ingénierie productique, de rechercher un nouveau concept d'outil de simulation, répondant aux préoccupations évoquées ci-dessus. La conception d'un outil de simulation est un projet très délicat qui présente trois niveaux de difficultés :

- Choix d'un modèle de simulation pour décrire le comportement du système à simuler. Pour cela, il faut une approche de modélisation (méthodologie de description) suivie d'une technique d'implémentation sur machine (méthodologie de programmation).
- Choix d'une technique pour la mise en oeuvre de la simulation (moteur de simulation) permettant de faire évoluer le modèle de simulation.
- Adoption d'un style d'interactivité pour la communication homme-machine.

I.2/ MODELES DE SIMULATION

Un modèle de simulation est un modèle expérimental dont le comportement est proche de celui du système que l'on veut simuler. Afin de tester sa validité, le modèle de simulation est soumis à des excitations. Les résultats obtenus par l'expérimentateur l'autorisent à apporter des jugements et des corrections éventuelles permettant d'aboutir à un modèle valide.

Un modèle peut résoudre un problème d'un système réel qu'il représente et ne pas pouvoir le faire dans le cas d'un autre problème. C'est pourquoi, le processus de construction du modèle correspondant à un système quelconque doit toujours être guidé par les objectifs de l'application d'une part et avoir le niveau d'abstraction et de simplification voulu afin de répondre aux problèmes du système réel représenté.

Dans ce sens, plusieurs types de modèles de simulation sont utilisés et peuvent être regroupés en trois catégories :

- les modèles de simulation physiques,
- les modèles de simulation analogiques,
- les modèles de simulation numériques.

I.2.1/ Les modèles de simulation physiques

Il s'agit de modèles construits par une simple ré-implémentation physique du système réel, souvent sous forme de prototype réduit. De nombreux exemples existent pour ce type de simulation, citons le cas de simulation du comportement aérodynamique des voitures ou des avions par des prototypes miniaturisés.

Ces modèles ont la propriété de reproduire de manière fidèle le comportement du système simulé mais, hélas, au détriment d'un coût élevé et d'un temps de préparation assez long. En effet, dans le cas de simulation des accidents de trains par exemple, l'expérimentateur n'a le droit qu'à un seul essai. C'est pourquoi, il procède généralement par une longue phase de préparation qui risque de durer des années.

I.2.2/ Les modèles de simulation analogiques

Dans plusieurs cas, un système peut être représenté mathématiquement sous forme d'équations différentielles. La simulation analogique est basée sur l'utilisation des amplificateurs opérationnels bouclés agissant en sommateurs ou intégrateurs pour représenter le système d'équations. Les variables du système sont représentées par des tensions analogiques.

Un des avantages de ce type de simulation est que les résultats peuvent être visualisés en temps réel sur l'oscilloscope ou sur tout autre moyen d'enregistrement. De ce fait le temps de réponse est instantané.

Par ailleurs, il faut tout de même signaler que la simulation analogique est un outil performant dans le cas des systèmes linéaires ce qui n'est pas vrai dans le cas des

systèmes non linéaires. La raison principale de cette limite est due à la saturation des amplificateurs réels. Des montages spéciaux sont utilisées dans ce cas [BAN 86].

I.2.3/ Les modèles de simulation numériques

Dans la pratique des simulations numériques, un calculateur est utilisé pour intégrer pas à pas, c'est à dire en fonction du temps, les équations représentant le procédé; dans la mesure ou ces équations traduisent *correctement* le comportement du procédé, les réponses obtenues en simulation sont *proches* des réponses que l'on pourrait relever sur le procédé. Pour ce qui suit dans cette thèse nous parlerons désormais de simulation pour désigner la simulation numérique.

I.2.4/ Les modèles de simulation hybrides

Nous avons vu dans les paragraphes précédents les différents types de modèles, chacun avec ses avantages et ses inconvénients. Il existe des modèles hybrides basés sur la combinaison de plusieurs types de modèles permettant ainsi de cumuler les avantages de l'un ou l'autre de ces modèles (fidélité pour le modèle physique, rapidité pour l'analogique et flexibilité pour le numérique). C'est le cas des applications pédagogiques qui regroupent, par exemple, une partie physique (moteur, chariot, ...etc) et une partie analogique (intégrateurs, dérivateurs, additionneurs, ...etc).

I.2.5/ Conception d'un modèle de simulation

La phase de conception d'un modèle de simulation passe par deux phases distinctes: une phase de description du système simulé et une phase d'implémentation.

La phase de description se veut conceptuelle et fait abstraction de tout ce qui est matériel. Son but est d'aboutir à un ***modèle de description*** abstrait du système réel en se basant sur une méthodologie de description (I.3).

La phase d'implémentation quant à elle, décrit la manière dont ce même ***modèle de description*** sera programmé sur machine. Son but est de transformer le modèle de description (fruit de la réflexion) en un ***modèle de simulation*** prêt à une expérimentation. Cette opération est basée sur une technique d'implémentation (I.4).

I.3/ LES METHODOLOGIES DE DESCRIPTION

Plusieurs techniques de description ont été adoptées pour représenter des systèmes industriels de production. La différence entre ces techniques influe directement sur la conception des logiciels de simulation et sur leur niveau de convivialité. Cette différence est due, principalement, à la façon dont est représentée la logique de changement d'état. De plus, une autre ligne de partage entre ces différentes techniques, est l'existence ou non de résultats mathématiques et d'algorithmes permettant de calculer directement des indices de performance sans dérouler l'historique des changements d'état. Ceci nous conduit à organiser les techniques de description en deux catégories: les méthodes analytiques et les méthodes orientées vers la simulation.

I.3.1/ Les méthodes analytiques

Ces méthodes permettent de trouver la solution à un problème (la commande optimale par rapport à un certain critère) par résolution mathématique. Elles ne font pas appel à une interprétation des résultats. L'outil le plus utilisé pour la résolution est la théorie des files d'attente. Bel et Dubois [BEL 85] présentent une revue critique détaillée de modèles analytiques (stochastiques) ou orientés vers la simulation. Ils sont arrivés à la conclusion suivante :

" En dépit de l'intérêt que peuvent présenter les modèles analytiques de systèmes de production en vue de l'évaluation grossière de leurs performances dynamiques et de l'analyse de leur logique de fonctionnement, ces modèles sont en général trop agrégés,

et deviennent vite insuffisants à un stade plus avancé de leur conception. Dans ces cas, il est nécessaire d'avoir recours à la simulation ".

I.3.2/ Les méthodes orientées vers la simulation

Ces méthodes consistent à reproduire pas à pas, événement par événement, l'évolution de l'état du système dans le temps, dictée par la logique de changement d'état. Cette logique peut être décrite par des algorithmes ou procédures aussi complexes qu'il sera nécessaire pour bien décrire le fonctionnement réel. Cette souplesse dans la description permet de modéliser le système avec le degré de détail que rend nécessaire le problème à résoudre. Il y a plusieurs façons de décrire la logique de changement d'état. Suivant que l'on décrit le système à simuler, par les événements qui se produiront, les activités qui seront entreprises ou les processus rencontrés, on aboutit à trois approches différentes : l'approche par événement, par activité et par processus.

I.3.2.1/ Approche par événement

C'est l'approche la plus générale. Elle consiste à rassembler tous les événements qui peuvent se produire et à décrire la logique de changement d'état. Dans cette logique, on peut distinguer :

- des règles liées au mode opératoire sur le procédé de fabrication (exemple, fin d'usinage d'une pièce),
- des règles liées à la conduite/gestion (exemple, si le produit A et le produit B sont prêts, alors démarrer la fabrication du produit C).

Cette approche utilise un principe identique à celui qu'on retrouve dans les réseaux de Pétri.

L'application la plus adaptée à ce genre de formalisme est la simulation de fabrication de pièces quelconques par des machines. L'exemple de la figure I.2 montre le cas de fabrication d'un seul type de pièces par deux machines M0 et M1.

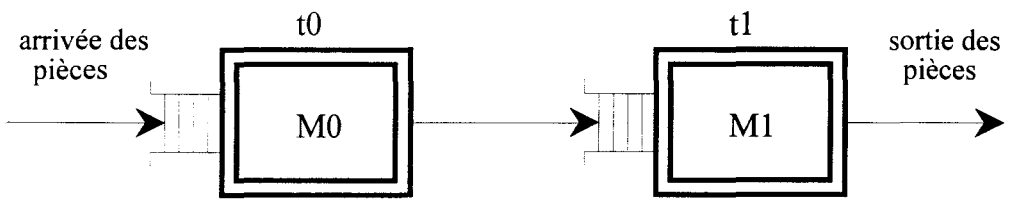


Figure I.2 : exemple de fabrication de pièces

Des pièces brutes arrivent de manière aléatoire et sont stockées à l'entrée de la machine M0 (zone de stockage). Lorsque cette machine est disponible, une seule pièce est usinée pendant un temps t0. Cette dernière est ensuite transportée vers la zone de stockage de la machine M1, en attente d'un deuxième usinage de durée t1.

Le graphique correspondant au modèle de description de cet exemple est décrit par le réseau de Pétri de la figure I.3.

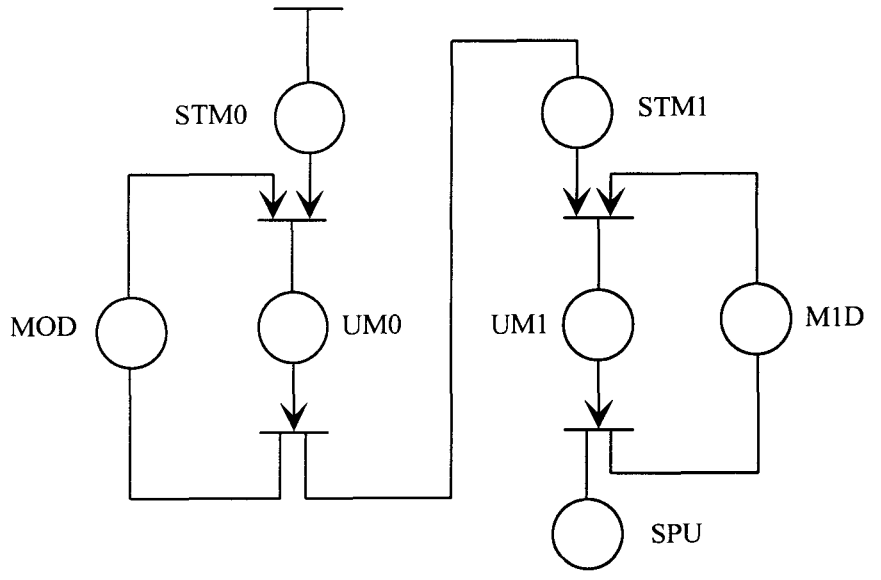


Figure I.3 : description par réseau de Pétri

Les places correspondant aux différents états des machines M0, M1 et des zones de stockage, sont les suivantes :

STM0 : stockage de la machine M0,

STM1 : stockage de la machine M1,

UM0 : usinage par la machine M0,

UM1 : usinage par la machine M1,

M0D : machine M0 disponible,

M1D : machine M1 disponible,

SPU : sortie des pièces usinées.

La description par réseau de Pétri aboutit souvent à des représentations complexes et touffues. Afin de gagner en concision, on utilise les réseaux de Pétri colorés. Ainsi, des réseaux de Pétri avec un très grand nombre de places et de transitions seront décrits plus simplement.

I.3.2.2/ Approche par activité

C'est une approche qui s'appuie sur un raisonnement naturel; un procédé est décrit comme un ensemble de ressources matérielles ou immatérielles. Il s'agit par exemple de machines, robots, opérateurs, ... etc. Chacune des ressources décelées dans le procédé, fait l'objet d'un enchaînement d'activités et d'attentes décrivant son comportement.

Ainsi, le procédé peut être vu à travers les activités en indiquant les conditions nécessaires à leurs débuts et à leurs fins.

Les attentes débutent à la fin de chaque activité. Elles se terminent lorsque les conditions nécessaires à l'activité suivante sont réunies. Pour chaque ressource, on obtient des cycles d'activités. Les différents cycles ainsi formés sont reliés aux niveaux des activités communes.

La réflexion débouche sur un graphique composé de deux symboles (figure I.4).

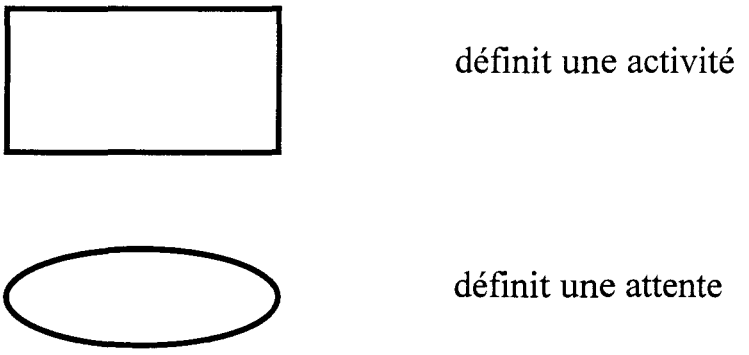


Figure I.4 : approche par activité

Lorsqu'une ressource est active, elle est engagée dans une activité. Lorsqu'elle est inactive, elle est en attente ou disponible dans une file d'attente.

Prenons l'exemple du chargement manuel d'une machine-outil par un opérateur. Les ressources identifiées sont l'opérateur et la machine.

Le cycle de la ressource opérateur comprend l'activité *chargement* de la machine. Une fois que cette activité est terminée, l'opérateur est *disponible* (figure I.5).

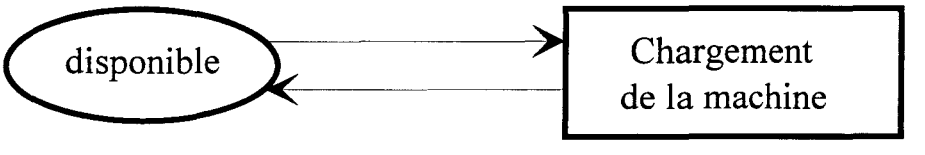


Figure I.5 : cycle opérateur

Le cycle de la ressource machine est décrit à partir des activités d'usinage et de chargement (figure I.6).

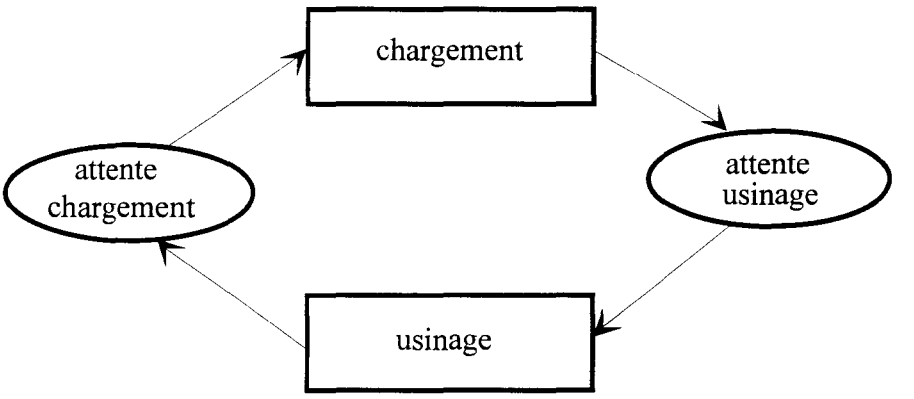


Figure I.6 : cycle machine

Lorsque l'on réunit les deux cycles obtenus au niveau de l'activité commune de chargement, le graphique résultant est le suivant (figure I.7).

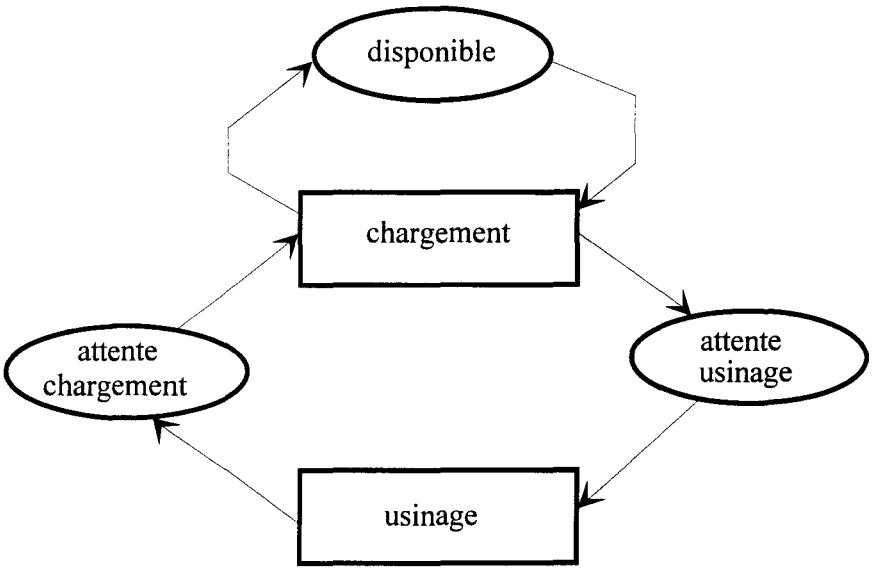


Figure I.7 : cycle d'activités

I.3.2.3/ Approche par processus

On parle de cette approche lorsque la modélisation consiste à assembler des processus. Les processus sont formés de séquences d'événements et d'activités. Ces processus peuvent être interdépendants ou non, ce qui nécessite leur gestion et leur

ordonnancement dans le temps. Plusieurs problèmes ont donné naissance à des techniques de gestion de processus. On trouve ainsi, les techniques de synchronisation, d'exclusion mutuelle, etc.

La puissance des outils utilisant cette approche est liée aux processus mis à la disposition de l'utilisateur (manutention par robots, par convoyeur, etc.). Ce dernier doit pouvoir en plus définir lui même des processus, notamment au niveau des règles de gestion.

Un exemple de modélisation est appliqué à la simulation d'un garage de lavage de voitures. La figure I.8 donne un aperçu général des éléments constituant l'application.

Chaque élément définit sa tâche comme une séquence d'actions. Dans le cas d'un véhicule appartenant à un client par exemple, la séquence d'actions consiste à :

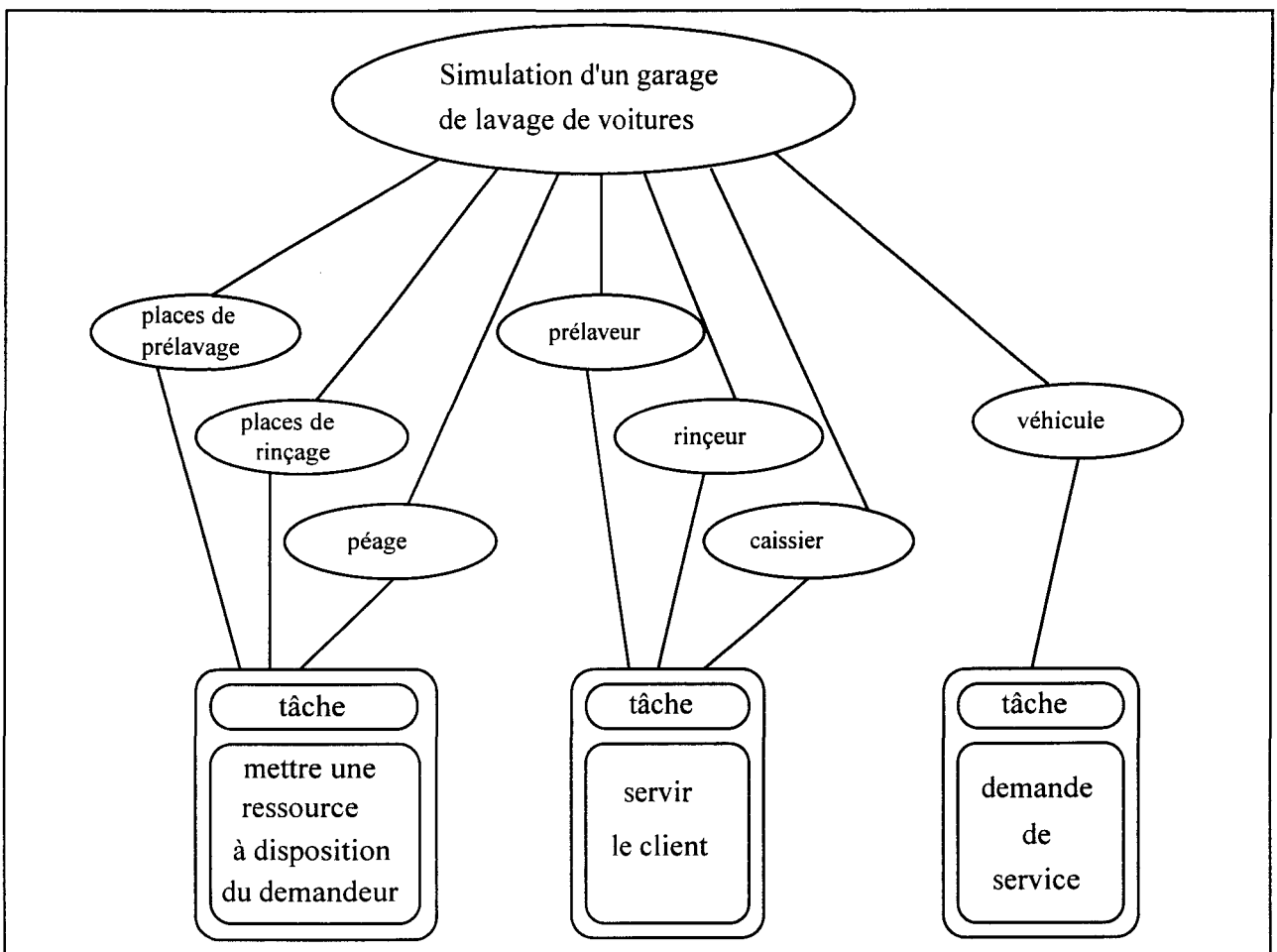


Figure I.8 : simulation d'un garage de lavage de voiture

- demander un service à un prélaveur,
- attendre le temps de prélavage,
- demander un service à un rinceur,
- attendre le temps de rinçage,
- payer le prix du service,
- quitter le garage.

I.4/ IMPLEMENTATION DES MODELES SUR MACHINE

La phase d'implémentation consiste à traduire le modèle établi lors de la phase de description, en termes de langage de programmation. Cette traduction dépend d'une part de l'approche de modélisation adoptée et d'autre part du langage utilisé.

Dans le cas de l'approche par événement, l'écriture du modèle consiste à programmer la logique de changement d'état. Supposons que la méthodologie utilisée soit celle du réseau de Pétri, l'implémentation consisterait à programmer l'ensemble des places, des transitions et des arcs décrivants le modèle en question.

La modélisation par activité consiste à programmer les conditions de déclenchement et de fin des activités.

L'implémentation d'un modèle de description basé sur l'approche par processus se résume par la création de processus en se servant, éventuellement, des processus mis à la disposition de l'utilisateur sous forme de bibliothèque.

I.5/ APPROCHE PAR OBJET

L'approche par objet est une méthodologie de description naturelle qui induit une technique d'implémentation sur machine de même nature. Ainsi, il est nécessaire de distinguer la description orienté objet de l'implémentation par objet.

Une longue pratique de la structuration de simulateurs dédiés a vite conduit les concepteurs à modulariser leur description en rassemblant les logiques de changement d'état correspondant à un type d'objet physique donné (machine, transport, etc.) ainsi que les paramètres de fonctionnement des différents objets de ce type [BEL 90].

Un modèle de description orienté objet est ainsi composé d'objets ou entités (pièces à fabriquer, chariots, etc.) et de relations entre ces objets.

L'implémentation par objet consiste à créer les différentes classes, déjà structurées lors de la phase de description du modèle, puis à les *instancier* (figure I.9). L'implémentation par objet peut être aussi utilisée pour des modèles de description à événements à activités ou à processus.

La création de classes est directement liée au langage de programmation utilisé.

La phase d'instanciation se décompose en deux parties essentielles :

- *instanciation des modèles élémentaires :* En termes de langages à objet, chaque composant élémentaire est une " *instance* " de la classe à laquelle il appartient. L'instanciation d'un modèle d'objet élémentaire consiste, après avoir créé une instance de la classe considérée, à attribuer ses variables en leur affectant des valeurs correspondant à l'état initial de l'objet. L'instanciation d'un système complexe se traduit par l'instanciation des différents composants élémentaires.

- *Instanciation du modèle global :* cette opération vient en dernier lieu et consiste à regrouper les modèles élémentaires instanciés de l'application, grâce à un mécanisme de communication afin qu'ils puissent s'échanger les informations nécessaires au bon fonctionnement de cette application.

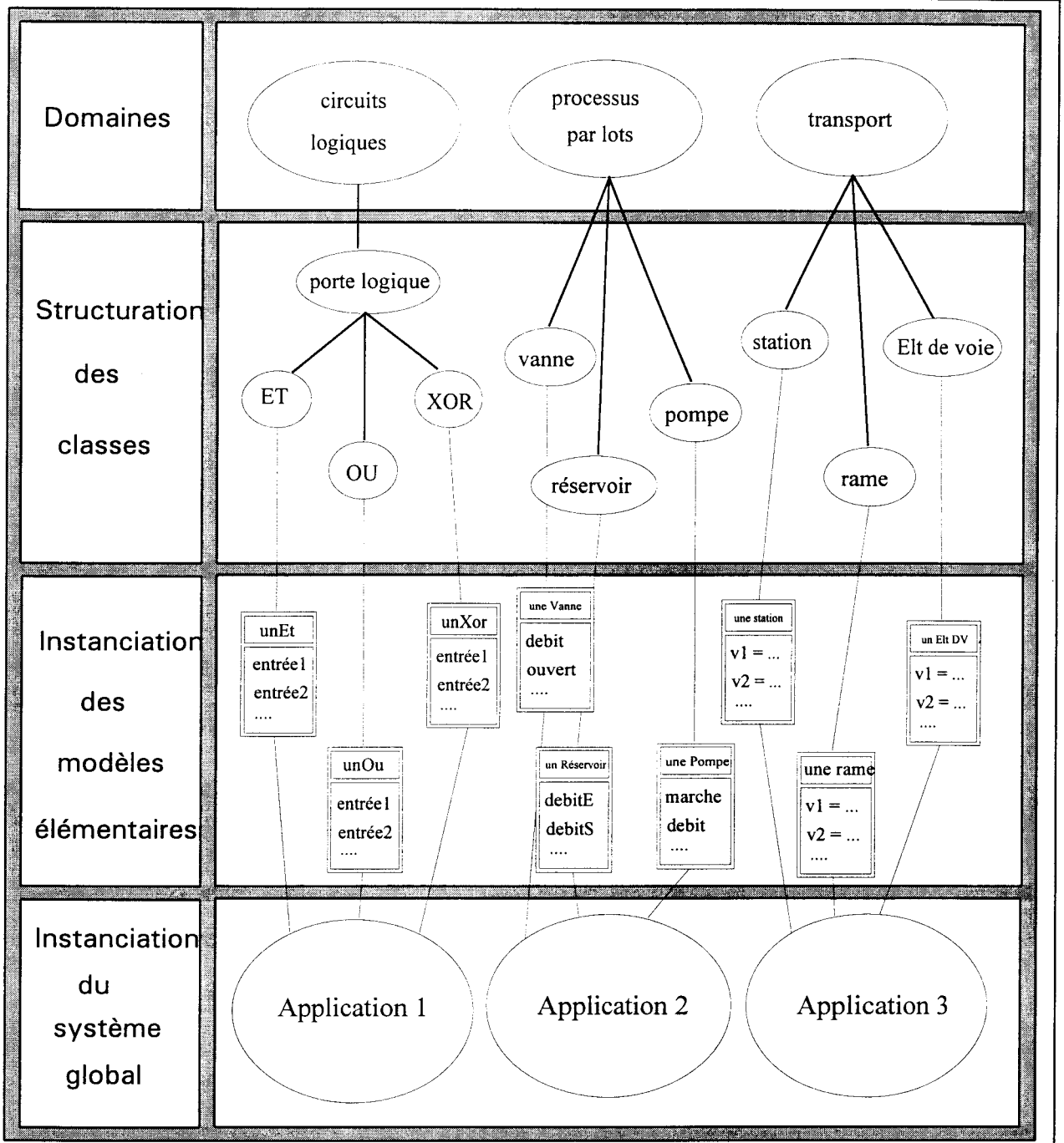


Figure I.9 : les phases d'instanciation

I.6/ LA SIMULATION

Lors de la résolution d'un problème, la simulation utilise un modèle (*modèle de simulation*) pour construire une plate-forme sur laquelle on peut faire diverses

expérimentations. Sachant que cette plate-forme possède le même comportement dynamique que celui du système réel, les résultats des expérimentations nous permettent d'induire des jugements sur le système et des solutions aux problèmes. On parle aussi de logiciels de simulation ou de simulateurs pour désigner l'outil informatique qui nous permet de réaliser des simulations sur des modèles de systèmes existants ou non.

I.7/ CARACTERISTIQUES GENERALES DES SIMULATEURS [FLA 89]

L'appellation simulateur est très générale et recouvre en fait, en terme de dimensions du programme et des fonctions réalisables, un très large éventail de logiciels. Il apparaît donc nécessaire, dans un premier temps, d'établir un certain nombre de critères qui permettent de différencier les logiciels de simulation.

I.7.1/ Simulateurs statiques et dynamiques

Un simulateur statique est un logiciel capable de prédire, en tout point d'un système en marche continue équilibrée, les débits et caractéristiques de la matière d'oeuvre. Une marche continue équilibrée sous-entend un point de fonctionnement stable.

Un simulateur dynamique est un logiciel qui propose, lui aussi, une simulation statique avec en plus, la possibilité de calculs dynamiques entrepris à partir de la perturbation des conditions de stabilité.

En effet, les simulateurs dynamiques permettent soit de calculer les régimes transitoires, c'est-à-dire les trajectoires suivies par les variables de procédé entre deux régimes statiques (simulation déterministe), soit de calculer les variations aléatoires autour d'un régime stationnaire (simulation stochastique).

I.7.2/ Flexibilité du modèle

La flexibilité du modèle apporte un deuxième critère de différenciation des logiciels de simulation qu'ils soient statiques ou dynamiques. Ainsi on trouve :

- Des logiciels dédiés à la simulation d'un modèle unique, par exemple, pour des simulateurs utilisés sur un site de production.

- Des logiciels permettant d'étudier un nombre limité de modèles qui correspondent aux situations habituellement rencontrées dans un domaine spécifique comme l'industrie pétrochimique, par exemple.

- Ou enfin, des logiciels permettant une flexibilité complète dans l'organisation des équipements dans une usine. Dans ce cas, les logiciels présentent généralement une limite dans le dimensionnement du schéma qui peut être simulé (nombre d'équipements, nombre de flux de matière), limite qui est plus ou moins contraignante selon la complexité du domaine dans lequel on travaille.

I.7.3/ Degré de spécificité des procédés

Le degré d'universalité, et inversement le degré de spécificité, sont deux aspects importants qui permettent de différencier les simulateurs. A une extrémité du spectre, on trouve les grands simulateurs qui ont été développés principalement pour l'industrie chimique comme **Aspen** [GAL 81] ou **Flowtran** [ROS 77]. Ces logiciels caractérisent la matière d'oeuvre circulant dans le procédé par ses propriétés chimiques, ses propriétés physiques, ses caractéristiques thermodynamiques. Ils offrent une gamme très importante d'opérations unitaires qui peuvent être simulées (mélangeurs, cristalliseurs, colonne à distiller, filtres, réacteurs chimiques, etc). Ces logiciels, d'un coût élevé , requièrent un support informatique de forte puissance et demandent un apprentissage assez considérable. A un degré de spécificité plus haut, on trouve les simulateurs complètement spécialisés qui sont en général beaucoup plus performants

car ils décrivent les opérations en utilisant au maximum la connaissance réelle des mécanismes qui entrent en jeu dans chaque opération unitaire.

I.7.4/ L'aspect temps réel

Avant de procéder à une classification des simulateurs basée sur le critère temps réel, voyons d'abord ce qu'est un système temps réel d'une manière générale.

Un système temps réel détecte et/ou contrôle des *événements extérieurs* au système tout en étant soumis à des contraintes de *temps*. Les systèmes temps réel ajoutent une nouvelle dimension aux systèmes, *le temps*, ce qui les rend encore plus difficiles à développer.

La problématique du temps réel nécessite la considération du temps non pas comme un facteur de performances, mais comme une contrainte logique de base.

Ainsi, nous distinguons deux catégories de simulateurs. D'une part les simulateurs non temps réel où la notion de temps n'a d'intérêt qu'en termes de rapidité de réponse, d'autre part, ce sont les simulateurs temps réel où l'on fait intervenir les mécanismes fondamentaux qui sont principalement la gestion de processus, la synchronisation et la communication inter-processus.

Une simulation industrielle temps réel, peut consister à suivre l'état du procédé et à réagir, à toute évolution significative du procédé, dans un laps de temps (temps de réaction) qui garantisse la maitrise du contrôle du procédé.

I.8/ LES APPLICATIONS DE LA SIMULATION

Les principales classes d'application de la simulation concernent les cinq domaines suivants (figure I.10) :

(a) la conception des systèmes industriels,

(b) la reconfiguration et les modifications des systèmes,

(c) l'aide à la décision,

(d) l'analyse dynamique des procédés,

(e) les applications pédagogiques de la simulation.

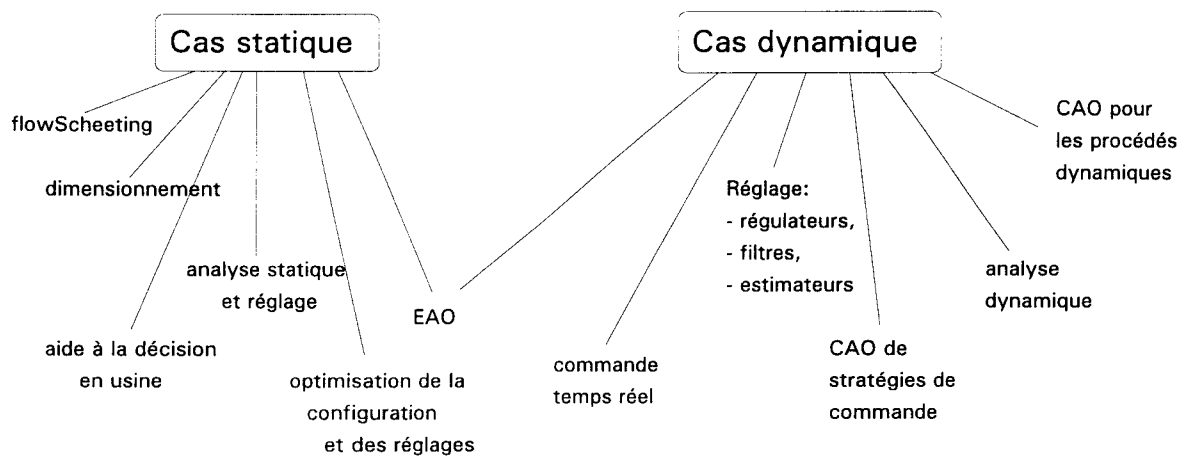


Figure I.10 : applications de la simulation

L'informatique appliquée intervient, d'une façon plus ou moins élaborée, à l'intérieur de chacune de ces applications. Au premier niveau, elle intervient comme outil d'automatisation des techniques conventionnelles qui étaient effectuées manuellement.

Dans ce cas l'application informatique est modulaire et correspond à l'utilisation de petits programmes séparés ayant des tâches spécifiques : bilan de matière, résolution d'une équation de dimensionnement, dessin assisté par ordinateur, simulation d'équipements unitaires, etc. A des niveaux d'intégration supérieurs, on

utilise des simulateurs plus complexes, permettant d'effectuer plusieurs traitements en parallèle.

I.8.1/ Conception des systèmes industriels

Le simulateur peut apporter une aide considérable dans la conception : dimensionnement des composants élémentaires du modèle à simuler et optimisation de la configuration du modèle.

I.8.2/ Reconfiguration et modification des systèmes

Certains réglages et reconfigurations sont parfois indispensables afin d'obtenir des produits finis, avec des caractéristiques particulières d'une part, et d'optimiser les flux d'entrées / sorties du procédé, d'autre part. Ceci n'est réalisable qu'avec des simulateurs dynamiques.

I.8.3/ Aide à la décision

Un simulateur, qu'il soit statique ou dynamique, peut fournir une assistance très utile aux ingénieurs ou opérateurs qui doivent prendre des décisions rapides sur le fonctionnement d'un procédé. Cela suppose que le simulateur soit implanté directement dans l'usine et accessible d'une façon conviviale, dans la salle de contrôle par exemple. Il s'agit dans ce cas de simulateurs dédiés aux installations particulières d'une usine. Les informations provenant des capteurs, comme les débits, peuvent être utilisées pour actualiser en permanence les conditions opératoires du simulateur qui est, en quelque sorte, toujours prêt à une sollicitation des opérateurs.

I.8.4/ Analyse dynamique des procédés

Grâce à un simulateur dynamique, il est possible d'étudier les variations temporelles des variables du procédé, induites par les perturbations qui affectent le fonctionnement d'une unité de l'installation. L'analyse dynamique par simulation renseigne aussi la possibilité d'évaluer à chaque instant l'état de fonctionnement d'un procédé (propriété d'observabilité). Enfin, le simulateur dynamique est un outil de sélection des variables manipulables permettant un pilotage performant du procédé (étude de contrôlabilité).

I.8.5/ Applications pédagogiques de la simulation [BOD 89]

L'utilisation de la simulation à des fins pédagogiques est devenue indispensable, car elle constitue un pont direct entre deux mondes qui étaient séparés : l'école et l'usine. En effet, au niveau formation, le simulateur est une usine à l'école qui permet aux étudiants d'avoir une perception plus concrète de la complexité des interactions entre procédés et de maîtriser les nouvelles techniques de conception et d'optimisation par ordinateur. D'un autre côté, la présence de simulateurs en usine (l'école à l'usine) est une aide à l'entraînement des opérateurs et à la prise de décision pour ces derniers. Le simulateur permet à l'opérateur de tester l'effet d'un changement opératoire qu'il désire effectuer, donc de décider s'il permet effectivement d'atteindre l'objectif fixé.

I.9/ LES TECHNIQUES DE SIMULATION

Pour un modèle donné, le concepteur peut choisir plusieurs types de simulation. Ce choix est lié aux objectifs de la simulation. Il dépend, d'une part, de l'approche de modélisation et, d'autre part, de la façon dont est géré le paramètre temps. Parfois ces deux points sont dépendants.

Dans le cas de l'approche par événement par exemple, le logiciel de simulation doit être en mesure de stocker la liste des événements crées. Le déroulement de la simulation consiste à rechercher, dans la liste, le prochain événement prévu. Alors, le nouveau temps courant à considérer est celui de la date d'arrivée de l'événement en question. L'échéancier est le module qui gère l'avance du temps au fur et à mesure que les événements apparaissent.

Dans le cas de l'approche par activité par contre, le logiciel doit faire la mise à jour des activités. A chaque incrémentation du temps, on examine si les conditions de début ou de fin de chaque activité sont réalisées et on met à jour les attentes effectuées. On peut éviter la scrutation systématique des activités en utilisant un échéancier comme pour l'approche par événement.

Dans tous les cas, dans une simulation, la notion de temps est elle même simulée ou discrétisée. Cette discrétisation peut se faire de deux manières différentes. Ils en découlent deux types de simulation :

- *simulation par tranches horaires :* lorsque le temps est divisé en petits pas réguliers appelés *pas de simulation*,
- *simulation événementielle :* lorsque le temps est divisé de façon irrégulière, cette division est basée sur le principe d'occurrence des événements dans la simulation.

Prenons l'exemple de la simulation de deux chariots qui se déplacent parallèlement, d'un point initial 0, suivant les trajectoires indiquées par la figure I.11.

Chaque chariot a pour tâche de se déplacer à droite pour atteindre la position 100, puis de reprendre le sens inverse (gauche) pour revenir à son état initial. Les deux cas de simulation peuvent être adoptés :

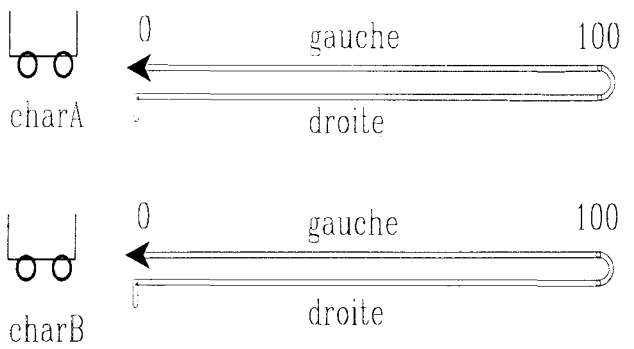


Figure I.11 : exemple des deux chariots

I.8.1/ la simulation événementielle

L'évolution des tâches, associées aux processus de simulation (cas d'une modélisation par processus), dépend du temps simulé (figure I.12). Elle applique la loi du tout ou rien et ne s'intéresse pas aux étapes intermédiaires.

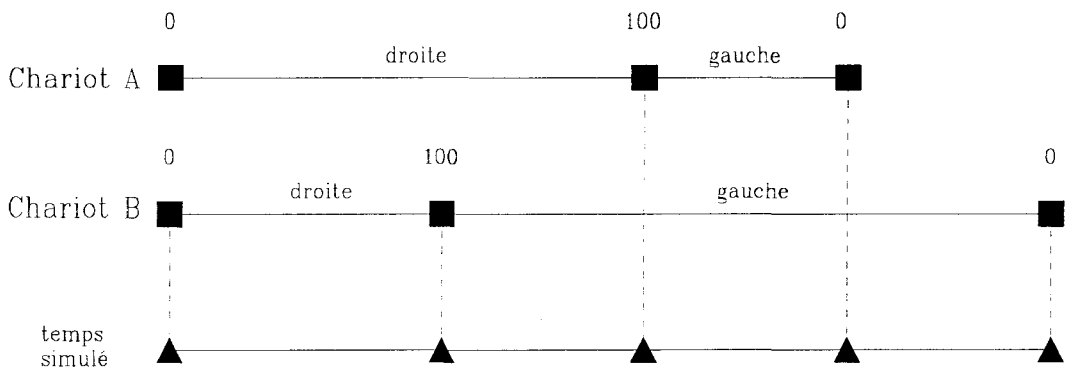


Figure I.12 : simulation événementielle

En effet, connaissant la vitesse du chariot et la distance à parcourir dans telle direction, on lance une temporisation (i.e. : suspendre le processus); cette temporisation serait égale aux temps de simulation nécessaire à cette évolution (distance / vitesse), après quoi le processus est réveillé et peut par exemple, afficher sa nouvelle position, changer éventuellement ses paramètres de fonctionnement (sens) et lancer une autre action.

I.8.2/ La simulation par tranches horaires

Dans la simulation par tranches horaires, on s'intéresse à l'évolution des acteurs, pour une vue microscopique des phénomènes, à chaque *pas de simulation* (figure 13).

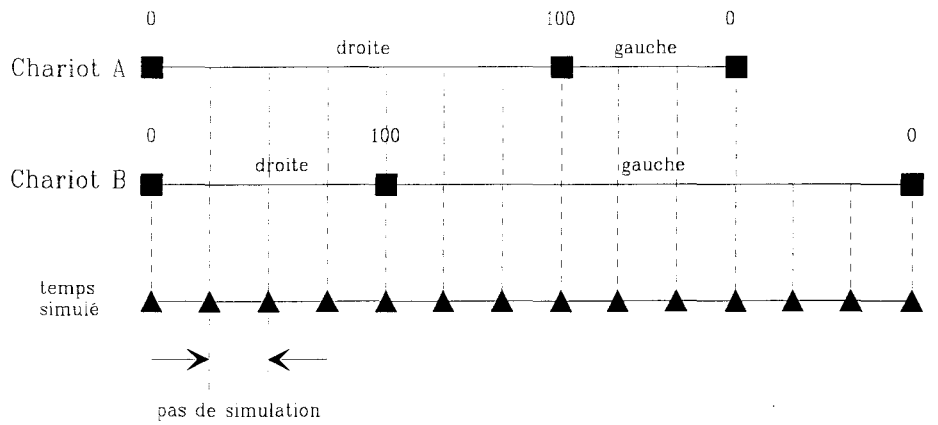


Figure I.13 : simulation par tranches horaires

La mise en oeuvre de la simulation se déroule comme le montre la figure I.14.

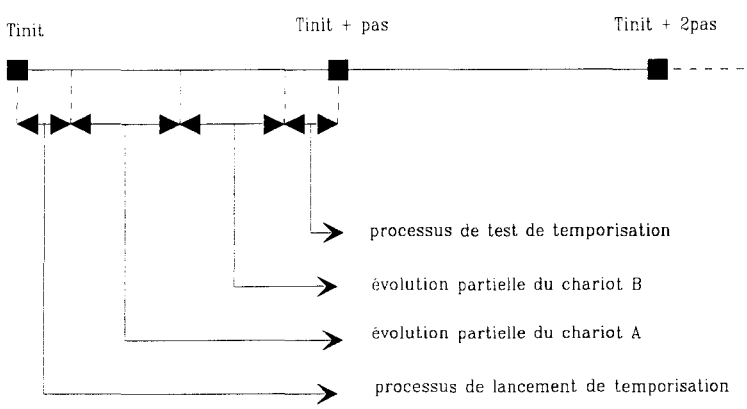
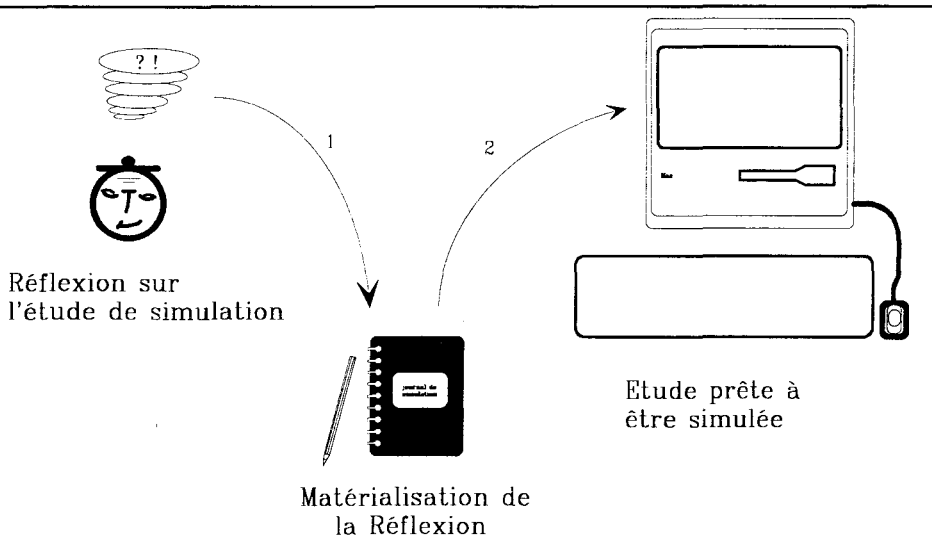


Figure I.14 : gestion du pas de simulation

En début de pas, une temporisation est initialisée puis lancée. Quand toutes les actions à effectuer lors du pas sont achevées, cette temporisation est comparée à la durée du pas. Le passage au pas suivant s'effectue lorsque la temporisation est au moins égale à cette durée.

I.10/ L'INTERACTIVITE

Le niveau de convivialité d'un logiciel de simulation est déterminé par le style d'interactivité utilisé [COD 90]. Son intérêt intervient lors du dialogue homme-machine pendant les phases de modélisation, de simulation et d'interprétation des résultats. Ceci nous conduit à distinguer deux types d'interactivité : *interactivité de conception* concernant la phase de modélisation et *interactivité d'utilisation* concernant les phases de simulation et d'interprétation des résultats.



1 - Aide à la matérialisation du problème de simulation posé, en un produit issu de la réflexion, qui tient compte du formalisme du logiciel utilisé.

2 - Aide à la transformation du produit de la réflexion en des informations prêtes à être utilisées par le calculateur pour lancer la simulation.

Figure I.15 : les phases de modélisation

I.10.1/ L'interactivité de conception

Pour son travail de modélisation, l'utilisateur doit poser le problème dans le formalisme du logiciel utilisé. Il doit le faire mentalement puis matérialiser sa réflexion par écrit ou à l'écran. Il dispose pour ceci, selon le logiciel, de différents styles d'interactivité.

Les deux composantes d'un style d'interactivité sont, d'une part, l'aide à la matérialisation du problème de simulation posé, en un produit issu de la réflexion qui tient compte du formalisme du logiciel étudié, d'autre part, la transformation du produit de la réflexion en des informations prêtes à être utilisées par le calculateur pour lancer la simulation (figure I.15).

Les cinq types d'interactivité sont présentés dans le tableau I.1.

Le souci d'être présent sur le marché impose aux concepteurs de faire évoluer leur produit vers une meilleure convivialité. A ce stade de la réflexion, on se doute bien qu'il est certainement plus facile de proposer une modélisation conviviale si le type de problèmes abordés représente un domaine restreint, c'est-à-dire si le logiciel est dédié à une catégorie d'applications. Au contraire, il sera plus difficile pour les logiciels qui offrent la possibilité d'aborder de larges types de problèmes, d'être conviviaux. L'ambition des concepteurs est que, quelle que soit la capacité à traiter des cas variés, la convivialité soit la plus grande possible.

I.10.2/ L'interactivité d'utilisation

Lors de la phase de simulation, l'utilisateur intervient en tant que pilote dont le rôle est d'initialiser le modèle (état initial), de mettre en marche la simulation, de surveiller l'évolution dynamique de l'état du procédé représenté et de réagir d'une manière ou d'une autre, face à tout changement d'état survenant au sein du même procédé. Ensuite, vient la phase d'interprétation des résultats de la simulation.

Ainsi, contrairement à la phase de modélisation, les deux composantes du style d'interactivité, lors de la phase d'utilisation, sont d'une part, l'aide à la mise en oeuvre de la simulation et le suivi de l'état du procédé, et d'autre part, l'aide à l'interprétation des résultats. Dans ces cas, le niveau de convivialité du logiciel de simulation est lié, en grande partie, à l'ergonomie de son interface graphique.

Dénomination de l'interactivité	1 - Forme de l'aide à la matérialisation du problème posé	2 - Forme de l'aide à la transformation du produit de la réflexion en des informations prêtes à une utilisation par calculateur
1 - Langage	Méthodes d'élaboration de programmes informatiques (algorithmes, programmation structurée, essais ...), Graphisme.	Programmation, Editeur, Compilateur, Documentation du langage.
2 - Langage aidé	Graphisme adapté à la simulation correspondant à un raisonnement proche du fonctionnement du procédé.	Programmation, Primitives spécialisées, Structuration du programme et assistance à la programmation (facilité par le travail graphique précédent).
3 - Programmation simplifiée	Une limitation du domaine d'application conduit à proposer une réflexion simplifiée dans un cadre restreint	Programmation simplifiée consistant à paramètrer un programme plus ou moins standard.
4- Sans programmation et formalisme	La réflexion s'appuie sur un formalisme solide. Il utilise un formalisme adapté à la description du fonctionnement et à la simulation	Pas de programmation. Le travail consiste à renseigner des questions en suivant : - un menu (les réponses sont tirées du travail précédent), - en utilisant souris et fenêtrage.
5 - Sans programmation	Transparence du formalisme. La réflexion peut être naturelle, proche du procédé et très ouverte aux domaines différents; facilité par une interactivité entre un graphique fonctionnel que l'on construit au fur et à mesure et les données que l'on indique.	Pas de programmation. Le travail consiste à renseigner des questions en suivant : - un menu (les réponses sont tirées du travail précédent), - en utilisant souris et fenêtrage.

Tableau I.1 : les cinq styles d'interactivité

Il détermine ainsi :

- la facilité et l'efficacité d'utilisation,
- la facilité d'apprentissage et en conséquence,
- l'efficacité et la productivité de l'utilisateur au travail.

Par ailleurs, l'utilisateur qui intervient pendant la phase d'utilisation est généralement différent de celui intervenant pendant la phase de conception. Ainsi, nous considérons désormais deux modèles :

- le modèle de conception ou modèle conceptuel créé par le concepteur à partir des modèles qu'il se fait de l'utilisateur au travail,
- le modèle mental que se crée peu à peu l'utilisateur en utilisant le système et qui lui permet de déduire des comportements du système, donc son propre comportement opératoire.

Un des objectifs majeurs, lors de la conception de l'interface utilisateur, est de faire en sorte que le modèle mental soit le plus proche possible du modèle conceptuel.

De plus, on distingue trois types d'utilisateur :

1- L'utilisateur naïf : il lui faut une interface intuitive et totalement auto-descriptive. On notera qu'il ne restera pas naïf éternellement, et que des possibilités d'utilisation plus efficaces devront être fournies en option.

2- L'utilisateur intensif (professionnel) : c'est le cas de l'employé qui effectue, à longueur de journée, des tâches répétitives. Il faut donc optimiser l'efficacité d'utilisation lorsqu'il est expérimenté.

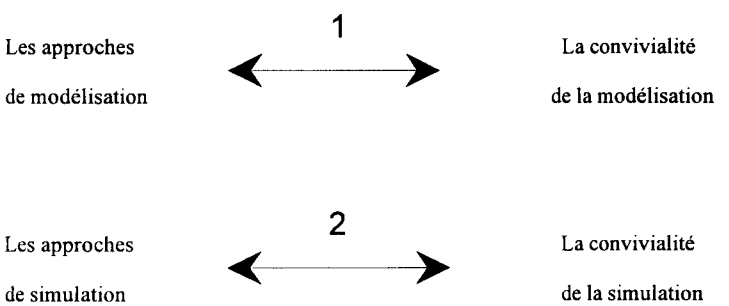
3- L'utilisateur occasionnel : c'est typiquement le cas d'un cadre qui, contrairement à l'employé, ne se spécialise pas sur une tâche. On privilégiera alors la facilité d'apprentissage. L'interface doit être intuitive, cohérente et intégratrice (facilité de passer d'une application à une autre).

I.11/ CONVIVIALITE DES LOGICIELS DE SIMULATION

L'utilisateur perçoit la simulation à travers trois phases: la phase de modélisation, la phase de simulation et enfin la phase d'interprétation des résultats.

Dans chacune d'elles, il dialogue avec la machine pour réaliser la tâche en question. La phase de modélisation souhaitée est une modélisation simple, conviviale et débouchant sur une représentation fidèle du système réel à analyser. La phase de simulation est souhaitée rapide et pouvant traiter des problèmes de grandes dimensions. Enfin, la phase d'interprétation souhaitée est celle qui suppose une présentation explicite de la simulation grâce à une animation du synoptique et la mise à disposition des outils nécessaires permettant d'analyser rapidement et efficacement les résultats de la simulation.

Le niveau de convivialité dépend énormément de l'approche adoptée lors de la modélisation et de la mise en oeuvre de la simulation. De ce fait, on obtient deux types de dépendances :



I.11.1/ La relation modélisation - convivialité

Une interface est dite conviviale lorsque, lors de son interaction avec la machine, l'utilisateur se crée une image qui soit la plus proche possible de celle que veut représenter le concepteur en un temps court.

Les différentes approches de modélisation se distinguent par leurs façons de représenter le système à simuler à l'utilisateur. L'approche par événement, utilisée à l'origine, n'offre pas une bonne convivialité. Elle a été délaissée pour donner naissance à des logiciels utilisant l'approche par processus dont on s'efforce d'améliorer la convivialité (GASP, SIMSCRIPT79, SEDRIC85).

L'approche par activité se base effectivement sur un raisonnement naturel mais nécessite la prévision d'un formalisme spécialisé correspondant à la définition des activités et des cycles d'attente, afin de faciliter la matérialisation de la réflexion de l'utilisateur. Deux exemples sont à considérer : ECSL 70 , HOCUS 85.

L'approche par processus est la plus utilisée dans les logiciels car elle offre une meilleure convivialité. En effet, cette approche ne nécessite par un formalisme spécialisé, ce qui permet d'éliminer, sinon de réduire au maximum la phase d'apprentissage et d'adaptation. Les concepteurs utilisant cette approche se sont investis pour faire évoluer les logiciels et ont abouti à des versions très conviviales telles que : CADENCE 85, MICROSAINT 80, WITNESS 85, SLAM2 85 ... etc.

L'approche par objet a apporté un changement radical sur la façon de voir et de gérer les éléments de la simulation. En effet, cette approche permet à l'utilisateur de voir les objets du système à simuler de la même manière que ceux du monde réel, c'est ce que l'on appelle *métaphore du monde réel.*

En plus, cette approche permet une utilisation particulièrement conviviale grâce à quatre niveaux possibles d'intervention de l'utilisateur [BEL 90]:

- Utilisation d'instances d'objets déjà existants : on peut ainsi constituer une bibliothèque de modules déjà utilisés : machines, convoyeurs, etc. L'utilisateur n'a plus qu'à assembler les modules qui lui sont nécessaires en indiquant les noms des instances correspondantes, le travail de modélisation préalable assurant leur parfaite cohérence.

- Instanciation de classes déjà existantes : on utilise des classes déjà définies avec des données particulières, on peut alors mettre à la disposition de l'utilisateur une bibliothèque de classes pré-définies assurant la majorité des domaines courants.

- Enrichissement de classes déjà existantes : lorsque l'on vient, lors de la phase de conception, préciser le système de simulation, on peut être amené à détailler certaines classes si ces nouvelles structures ne sont pas en bibliothèque. En effet, on peut, au lieu de définir intégralement la nouvelle classe, partir de classes déjà existantes, grâce aux mécanismes d'héritage associés à un langage orienté objet.

- Création de nouvelles classes : quand la base de modèles est insuffisante, on est obligé d'utiliser le formalisme du logiciel, qui est plus ou moins convivial, pour créer de nouvelles classes. A ce niveau d'interaction, la convivialité dépend énormément de l'ergonomie de l'interface homme - machine utilisée à cet effet.

I.11.2/ La relation simulation - convivialité

Lors de la mise en oeuvre de la simulation, le dialogue homme - machine peut être divisé en deux parties correspondant aux deux sens de communication :

- La première partie concerne les actions de l'utilisateur sur la machine grâce aux dispositifs d'entrées (clavier, souris, etc). Le rôle de l'interface graphique consiste à prendre en compte et à gérer ces actions qui font partie intégrante de la simulation. On parle ainsi d'interfaces événementielles. Nous verrons dans le troisième chapitre comment, avec une interface graphique, on est passé d'un ancien type de dialogue contrôlé par l'ordinateur qui présentait à l'utilisateur des écrans successifs, à un type de dialogue contrôlé par les actions de l'utilisateur, traduites sous forme d'événements auxquels l'ordinateur doit répondre.

- La deuxième partie concerne la présentation graphique des résultats statiques (taux d'utilisation des machines, historique d'évolution de certains paramètres, etc.) et dynamiques (animation des synoptiques).

Quel que soit le sens de la communication, le choix de l'approche de simulation est un facteur important pour une bonne convivialité.

En effet, l'approche événementielle qui s'intéresse à l'évolution macroscopique des éléments de la simulation, n'offre pas la possibilité de présenter les résultats de manière continue du fait de la logique tout ou rien. Le chariot de l'exemple précédant ne peut se trouver que dans l'une des positions suivantes : 0 ou 100, ce qui au niveau animation graphique ne présente pas beaucoup d'intérêts.

Cette limite est supprimée, dans le cas de l'approche par tranches horaires, dans laquelle la trajectoire du chariot est décomposée en petits déplacements réglables. Pendant chaque pas de simulation, le chariot peut afficher sa nouvelle position. L'utilisateur, étant limité par son acuité visuelle, ne verra pas de différence entre cette évolution (même si elle est toujours discrétisée) et une évolution réelle (continue). D'un autre coté, grâce à cette deuxième approche, il est possible à l'utilisateur d'intervenir dans la simulation en tant qu'acteur. Son action sera prise en compte à la fin de chaque pas de simulation par exemple.

En contre partie, l'approche par tranches horaires nécessite des calculs plus importants. Ces calculs sont souvent répétitifs entre un pas de simulation et un autre. L'inconvénient apparaît dans le cas ou les objets (acteurs) évoluent en même temps. En effet, le calcul de l'évolution de tous les objets risque de dépasser les capacités de la machine et de ralentir ainsi la simulation puisque le pas de simulation devient grand.

Afin de pallier cet inconvénient, on peut envisager deux solutions complémentaires :

- la première nécessite une machine puissante permettant des calculs très rapides,
- la deuxième est l'utilisation d'un moteur de simulation puissant et intelligent. Dans le chapitre II, nous verrons comment une modélisation à base de contraintes peut contribuer à l'optimisation des calculs. En effet, seuls les objets dont la contrainte est modifiée sont traités par le moteur de simulation.

I.12/ CONCLUSION

Dans ce chapitre, nous avons commencé par donner une définition de la simulation et de la notion d'interactivité. Après avoir introduit une classification des simulateurs, nous avons défini les trois facteurs importants qui influent directement sur la convivialité des logiciels de simulation interactive. Il s'agit des approches de modélisation, de simulation et du style d'interactivité utilisés. Il en découle deux niveaux de problèmes auxquels on a essayé d'apporter des solutions : le problème de la modélisation des systèmes à simuler et celui de la conception de l'interface homme-machine.

I.12.1/ problème de modélisation des systèmes à simuler

Ce problème est directement lié à la phase de modélisation, c'est-à-dire de l'approche adoptée pour représenter le système à simuler. En effet, quatre approches ont été étudiées : approche par événement, par activité, par processus ou par objets.

L'approche par objets permet une utilisation particulièrement conviviale grâce à quatre niveaux possibles d'intervention de l'utilisateur : utilisation d'instances d'objets déjà existants, instanciation de classes déjà existantes, enrichissement de classes déjà existantes, création de nouvelles classes. Nous proposerons à cet effet un modèle de simulation permettant de représenter des systèmes industriels appartenant à des domaines variés.

I.12.2/ problème de conception de l'interface homme-machine

L'interactivité des logiciels de simulation dépend en grande partie de l'ergonomie de l'interface graphique homme-machine. La conception de cette dernière constitue une phase très importante. Cette partie sera étudiée en détail au chapitre III dans lequel nous montrerons la nécessité d'associer un modèle à l'interface. Nous verrons que ce modèle sera constitué de trois parties : l'abstraction, représentée par le modèle de simulation du système à simuler, la représentation graphique modélisant l'interface et le traducteur reliant les deux premières.

Chapitre II

Modélisation des Systèmes

Industriels

Modélisation des Systèmes Industriels

II.1/ INTRODUCTION

Avant de pouvoir simuler un système quelconque, il est nécessaire de le comprendre. Il faut donc l'analyser puis le modéliser et ensuite il pourra être simulé. Si l'analyse paraît quelquefois assez simple pour comprendre le fonctionnement du système, les problèmes surviennent souvent durant la phase de modélisation.

Cette phase a été définie dans le chapitre précédent et décomposée en deux étapes : description du système et implémentation sur machine. Notre objectif est d'approcher un formalisme de description le plus naturel possible. Il est donc logique que nous ayons choisi l'approche par objet pour chacune des étapes de la modélisation. Le point de départ des différents travaux de notre équipe de recherche sur la simulation est le système O.S.L.O (Outil de Simulation en Langage à Objet) [CAN 88]. Ce système est basé sur une décomposition d'un système industriel en trois familles d'objets : les objets opératifs, les objets technologiques et les objets missions.

Dans un premier temps, nous abordons les systèmes industriels de production en apportant un certain nombre de définitions. Ensuite, nous expliquons brièvement la modélisation adoptée dans le cas du système O.S.L.O. Nous en ressortons les avantages et les inconvénients pour arriver à une modélisation plus générale.

Nous présentons notre approche de modélisation d'un système par l'introduction de la notion d'objet de simulation et son implémentation dans le cas du système Malaxeur. Enfin, nous terminons sur une présentation du fonctionnement du moteur de simulation M.A.R.C. (Moteur A Résolution des Contraintes).

II.2/ DECOMPOSITION D'UN SYSTEME INDUSTRIEL

Un *système*, en particulier un ***système de production*** ou ***système industriel***, est un ensemble d'éléments matériels ou immatériels réunis pour atteindre un objectif ou rendre un service déterminé; les éléments de cet ensemble sont en interaction, c'est-à-dire qu'ils exercent une influence les uns sur les autres. Le système est rarement isolé; il entretient des relations avec l'extérieur, dont il reçoit des flux (physiques, financiers, informatifs) et auxquels, à son tour, il renvoie des flux. Sa raison d'être réside dans certains flux privilégiés qu'il émet. Le but du système est donc défini par le choix que l'on a effectué parmi les flux sortants. Un système de production est constitué d'un ***processus de commande*** et d'un ***processus de fabrication*** (figure I.1).

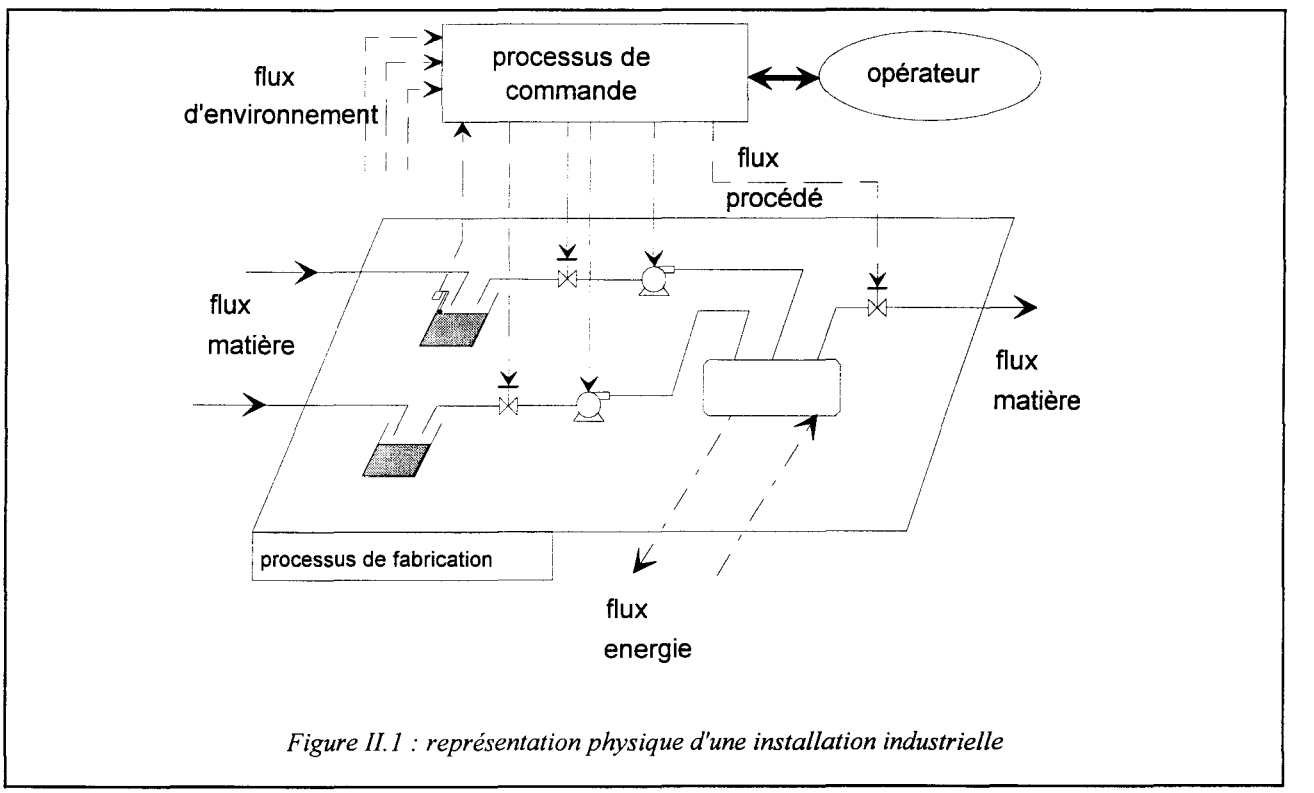


Figure II.1 : représentation physique d'une installation industrielle

Un ***processus de fabrication*** est un ensemble de ressources physiques (pompes, vannes manuelles, vannes régulées, capteurs de niveau, débitmètre, agitateur, réservoirs, conduites, ...etc) connectées de manière adéquate pour leur permettre de travailler ensemble. C'est le "hardware" du système de production.

Un ***processus de commande*** est l'ensemble des mécanismes animant le système pour l'exécution d'une tâche donnée. Il est donc défini pour que le système industriel fournisse le produit désiré, dans les meilleures conditions. Il reçoit des informations de l'extérieur et connaît l'état du système à l'aide de capteurs ; il commande le système par des actionneurs qui agissent sur les paramètres de son fonctionnement. Certaines de ses actions peuvent être exécutées sans interventions humaines, par exemple, la régulation d'un paramètre. Mais la conduite d'ensemble nécessite l'intervention d'un opérateur qui donne les instructions nécessaires à son bon fonctionnement.

II.3/ REPRESENTATION DES OBJETS D'UN SYSTEME INDUSTRIEL

Les différentes entités présentes, soit au niveau du processus de commande soit au niveau du processus de fabrication, dialoguent entre elles en échangeant un certain nombre de grandeurs physiques. Ce dialogue est caractérisé par des flux d'entrées et de sorties pouvant être continus ou discrets. Nous distinguons les types de flux suivants : flux de matière, flux d'énergie, flux d'environnement, flux d'informations.

Le **flux de matière** porte toutes les informations caractéristiques des matières d'oeuvre absorbées en amont ou évacuées en aval avec les volumes de consommation et de production. La définition de ces flux de matière est liée aux délimitations du système étudié et permet de modéliser une installation en tant qu'élément d'un système de production plus vaste.

Le **flux d'énergie** qui peut apparaître à divers niveaux, tout d'abord sous la forme du flux d'énergie de fabrication dont l'importance est considérable sur le plan de l'optimisation de la production. Le flux entrant correspond à l'énergie nécessaire et le flux sortant donne l'énergie produite ou perdue. Un des critères essentiels est actuellement l'économie d'énergie. Le flux d'énergie peut également apparaître au niveau des actionneurs. Son influence est surtout examinée lors de son absence et doit être prise en compte dans les causes d'anomalies. Il en va de même pour l'énergie d'alimentation des processus de commande proprement dits et de certains systèmes de mesure (capteurs).

Le **flux d'environnement** comprend des grandeurs telles que température, humidité, poussière, bruits. Il permet de replacer l'installation modélisée dans son contexte physique. Il intervient, lors de la conception, comme contrainte dans le choix des objets. Il permet également d'introduire des conditions d'alarme ou d'anomalies relativement à une modélisation de l'environnement. Le flux sortant peut correspondre aux déchets et pollutions diverses.

Le **flux des informations procédé** véhicule les données relatives au processus de fabrication. On y trouve les recettes, les paramètres de fabrication, les bases de données nécessaires à certains postes de travail.

Le **flux des informations opérateur et de contrôle logiciel** peut se situer à divers niveaux (opératif, commande, conduite, organisation, décision).

Les objets du système peuvent ainsi être représentés par une boîte noire à laquelle est associée un certain nombre de flux d'entrées et de sorties (figure II.2).

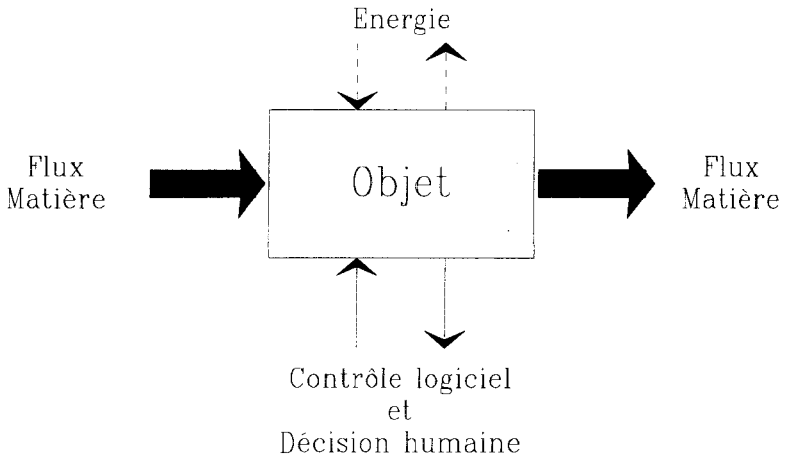


Figure II.2 : flux entrant et sortant d'un objet

II.4/ ANALOGIE AVEC LE MODELE S.A.D.T

SADT est une méthode de spécification et de conception. Elle est constituée d'un outil graphique et d'une méthodologie d'analyse et de description des besoins généraux d'un système. Cette technique utilise un ensemble structuré et hiérarchisé de

diagrammes dont la construction suit des règles garantissant une approche systématique des spécifications.

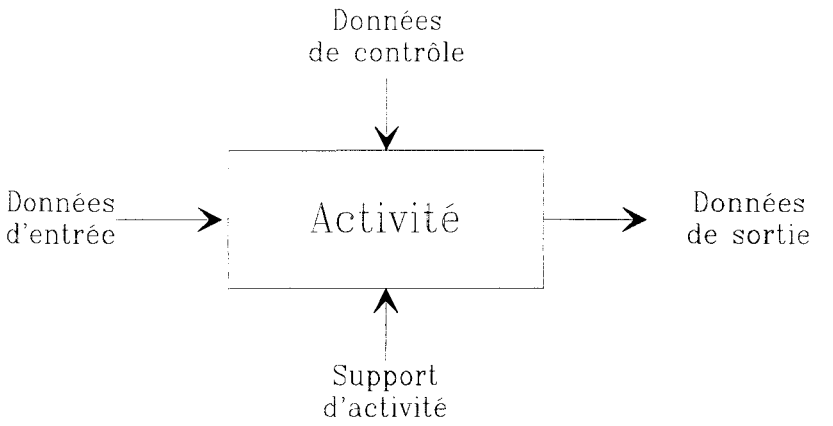


Figure II.3 : actigramme

SADT différencie le modèle fonctionnel, présentant les objectifs du système, du modèle de conception présentant les moyens à mettre en oeuvre. Ces deux modèles sont représentés respectivement par des diagrammes d'activité ou *actigrammes* et des diagrammes de données ou *datagrammes* (figures II.3 et II.4). La représentation SADT est une vision globale du système décomposée progressivement, jusqu'à obtenir les détails jugés suffisants.

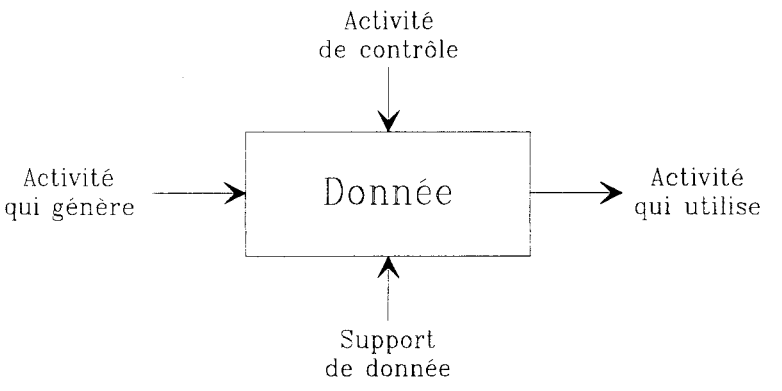


Figure II.4 : datagramme

Les boîtes ainsi obtenues d'un diagramme sont interconnectées par des flèches, dont l'une représente les données d'entrée ou les activités d'entrée, une autre donne le flux des données de sortie ou le flux des activités de sortie, une troisième permet le contrôle sur la boîte et enfin la dernière précise si nécessaire, le support physique de l'activité ou de la donnée.

Ainsi l'application de la représentation SADT à la modélisation des systèmes industriels conduit à l'affectation des actigrammes aux éléments du processus de commande et les datagrammes à ceux du système de fabrication.

II.5/ MODELISATION SELON O.S.L.O [CAN 87] [BAU 87]

Le modèle utilisé dans O.S.L.O a été développé sur la base de la décomposition vue précédemment, d'une part, et de la représentation SADT d'autre part.

Ce modèle autorise un nouveau découpage du système en trois parties. La première partie, appelée ***partie commande,*** correspond au processus de commande. Les deux autres parties sont le résultat de l'éclatement du processus de fabrication et sont appelées respectivement ***partie opérative*** et ***partie d'action et de compte-rendu*** (figure II.5).

Ce modèle a permis de définir trois familles d'objets : les ***objets opératifs*** (partie opérative), ***les objets technologiques*** (partie d'action et de compte-rendu) ***et les objets missions*** (partie commande).

II.5.1/ Objet opératif

Les objets opératifs servent à spécifier le comportement de la matière d'oeuvre en fonction des états des objets d'action de manière à générer les valeurs pour les objets de compte-rendu. La spécification du comportement doit satisfaire les lois physiques, chimiques, biologiques, etc. Dans cette famille, on trouve deux catégories d'objets : les objets de transfert et les objets passifs.

Les **objets de transfert** ont une existence physique mais ils échappent au contrôle des automatismes. Ils permettent le cheminement de la matière d'oeuvre.

Bien que ces objets n'influent pas sur l'étude de l'automatisation, le concepteur les considère comme des objets à part entière. Prenons par exemple l'étude du transfert d'un produit d'une cuve à une autre. Le concepteur étudie le type de canalisation à utiliser, le diamètre du tuyau selon le débit désiré, le matériau à employer en fonction du fluide, etc.

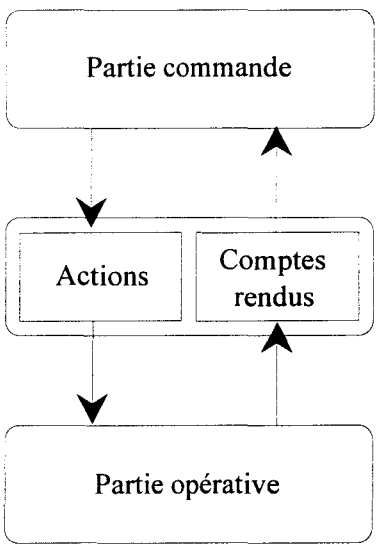


Figure II.5 : découpage général d'un système

En simulation, ces objets présentent l'intérêt de modéliser certains comportements de la matière d'oeuvre. Ils permettent d'une part, de faire apparaître des défaillances telles que l'encrassement, l'obturation, une fuite, ou la saturation, le blocage de pièce pour un convoyeur ...etc, et d'autre part, de modéliser des phénomènes physiques plus particuliers pouvant influencer l'évolution des matières d'oeuvre telles que les turbulences naissant à l'entrée d'une conduite de dérivation.

Comme les objets de transfert, **les objets passifs** ne subissent pas le contrôle direct des automatismes. Ils ne créent pas d'action sur la matière d'oeuvre mais, ils modifient des qualificatifs de ses caractéristiques. Le débit du produit entrant dans une cuve est traduit en élévation de niveau.

En simulation, seules les transformations de la matière d'oeuvre qui s'y produisent peuvent être modélisées.

II.5.2/ Objet technologique

Les objets technologiques regroupent les ***capteurs*** et les ***actionneurs***. Cette famille d'objets sert à relier la partie commande à la partie opérative. Autrement dit, la partie opérative, vue de la partie commande, est restreinte à la seule existence des actionneurs pour commander et des capteurs pour rendre compte de l'état du système.

Tout objet technologique est supposé avoir un comportement idéal (non faillible). Si, par exemple, le débit à la sortie d'une vanne ne donne pas de valeur correcte du fait d'un encrassement, cette anomalie est introduite par la variable correspondante au niveau de l'objet opératif associé.

II.5.3/ Objet mission

Un système de production est fait pour assurer une ou plusieurs missions, décomposables en sous missions, jusqu'à définir des missions élémentaires exécutant l'activation des objets technologiques d'action tels que moteurs, vannes, pompes...etc. Le contrôle de la bonne exécution de ces missions est fait via des objets technologiques de compte-rendu.

La mission a pour objet d'exécuter une tâche du système de production, par exemple, le transfert de la matière d'oeuvre d'une cuve vers une autre. Pour cela, il regroupe un certain nombre d'objets technologiques lui permettant d'accomplir sa tâche.

Une mission peut déclencher une ou plusieurs autres missions. Nous obtenons alors une organisation hiérarchique des missions correspondant à l'organisation complète du système de commande.

La décomposition en objets missions est duale de celle des objets opératifs. En effet, ces deux types d'objets regroupent les objets technologiques pour le séquencement de l'exécution de leur comportement.

II.5.4/ Exemple d'illustration (le Malaxeur)

Un malaxeur M reçoit des produits A et B pesés par une balance C et des briquettes solubles, amenées une par une par un tapis d'amenage.

L'automatisme permet de réaliser un mélange comportant les trois produits. Le cycle de fonctionnement est le suivant:

- l'action sur le bouton de départ cycle "dcy" provoque:
  - le pesage et l'amenage des produits A et B,
  - pesage du produit A jusqu'au repère a,
  - puis pesage du produit B jusqu'au repère b,
  - vidange de la bascule C dans le malaxeur,
  - amenage de N briquettes,

    contrainte: le remplissage des briquettes ne peut pas se faire avant l'amenage du mélange A + B. Il peut se faire en même temps.

- le cycle se poursuit par la rotation du malaxeur et par son pivotement, la rotation du malaxeur étant maintenue pendant la vidange. Le début de pivotement ne peut se faire qu'après un intervalle de temps t à partir du début de rotation du malaxeur TL2 <= t <= TL3,

- le cycle se termine par le retour du malaxeur en P0.

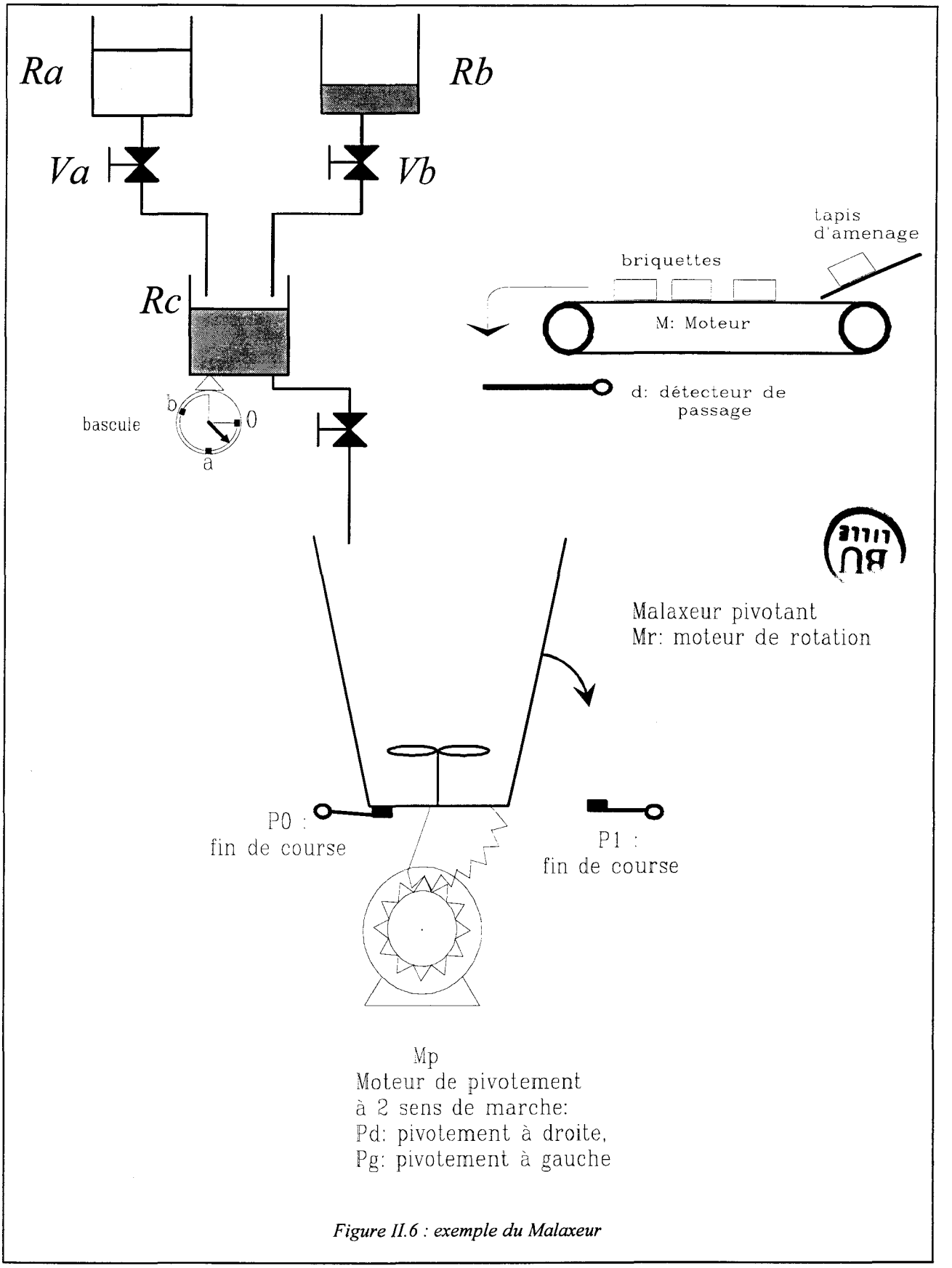


Figure II.6 : exemple du Malaxeur

II.5.4.1/ Une méthode de décomposition

Après avoir défini le cahier des charges du système à simuler, le travail consiste à lui associer une décomposition en trois familles d'objets: opératifs, technologiques et missions. Le principe consiste à décrire la partie commande du système, c'est-à-dire les différentes opérations correspondant à son fonctionnement ainsi que leur séquencement. L'utilisateur peut faire alors recours à un langage formel tel que le Grafcet. Chaque opération élémentaire ou composée est associée à une mission. La deuxième étape sert à définir tous les objets technologiques manipulés par les éléments de la commande afin d'accomplir leurs missions.

La dernière phase consiste à décrire des objets opératifs permettant de faire une liaison entre les objets technologiques d'action et les objets technologiques de compte rendu d'une part, et de faire évoluer la matière d'oeuvre d'autre part.

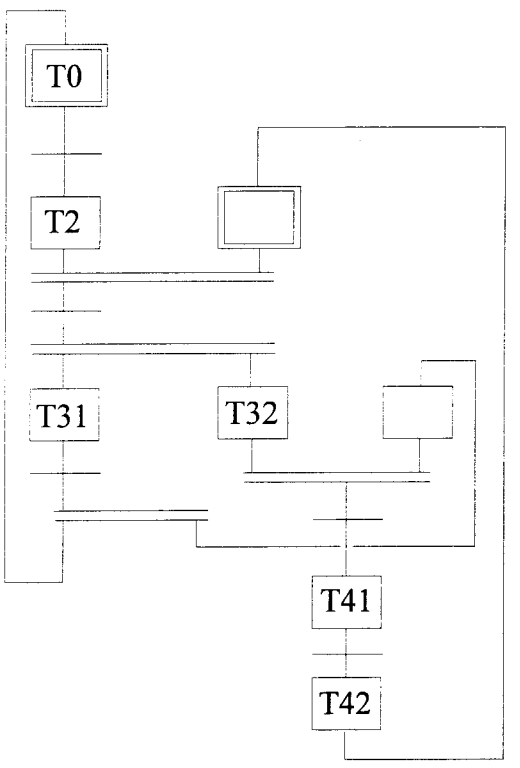


Figure II.7 : grafcet de commande

Dans le cas de l'exemple du Malaxeur, nous pouvons commencer par définir le Grafcet décrivant la partie commande (figure. II.7).

Les différentes opérations sont:

T2: Préparation des liquides, qui comprend :

- amenage et dosage de A,
- amenage et dosage de B,
- stockage de A + B.

T3: Remplissage du malaxeur, avec :

- T31: amenage de A + B,
- T32: amenage et comptage des briquettes.

T4: Malaxage décomposable en :

- T41: mélange,
- T42: vidange.

Cette représentation nous permet d'aboutir à la liste des objets missions élémentaires suivants:

- mission chargtBascule (T2),
- mission chargtAB (T31),
- mission chargtBr (T32),
- mission mélange (T41),
- mission vidange (T42).

La décomposition du système en objets est donnée en figure II.8. Cette décomposition n'est pas unique, il aurait été possible d'envisager, par exemple, une communication entre les objets "mission malaxeur" et "mission chargBascule", une simple communication entre ces objets aurait été suffisante dans notre cas.

Il existe un objet opératif "opérateur", capable de dialoguer avec les objets opératifs et technologiques, et dont le rôle est de représenter l'introduction, par un pilote de conduite, de consignes de fonctionnement ou de forçage sur les objets.

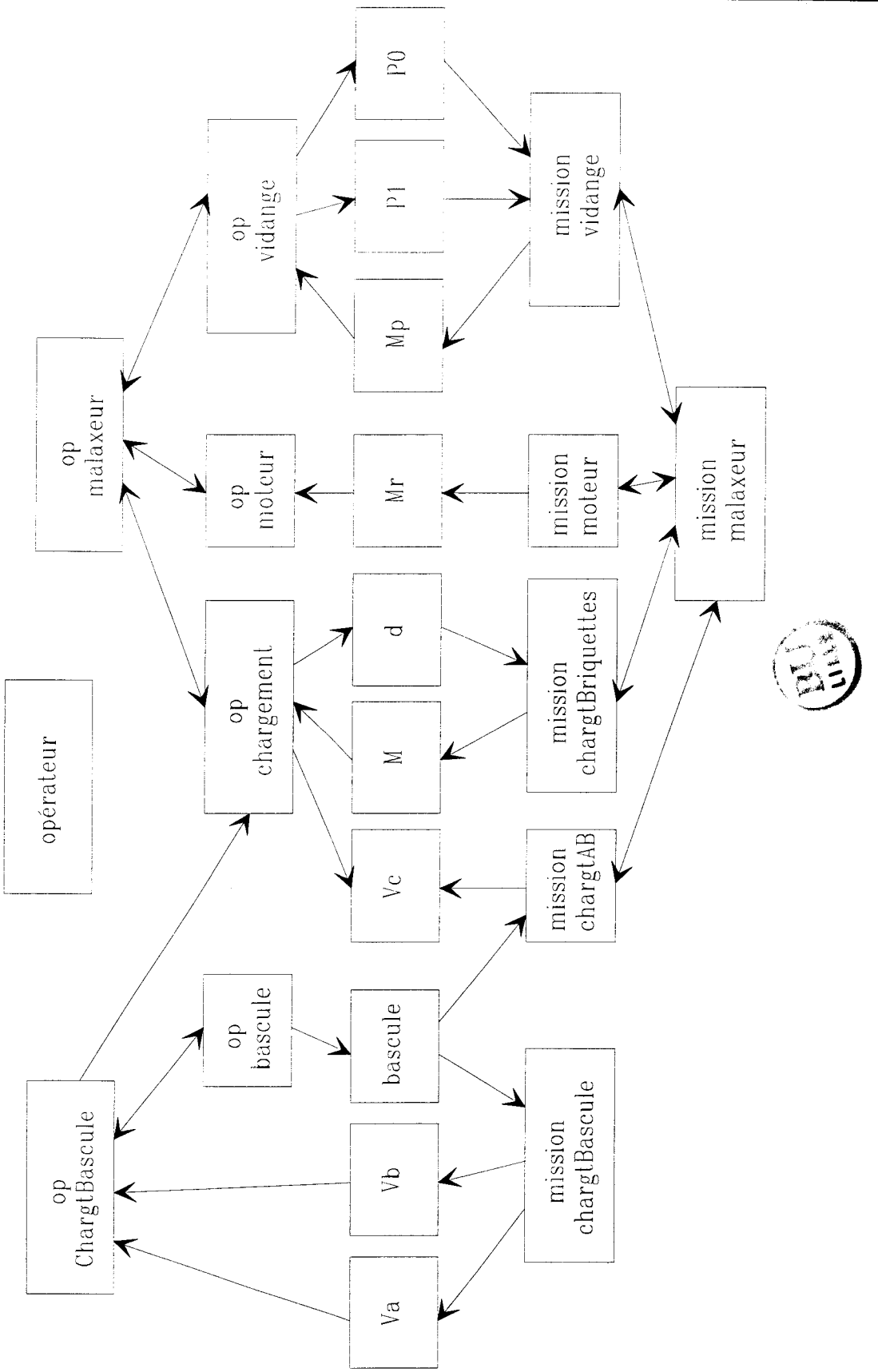


Figure II.8 : décomposition du système Malaxeur selon O.S.L.O

II.6/ INTRODUCTION DES OBJETS DE SIMULATION [MAR 91] [MOS 91]

La notion d'objet de simulation a été envisagée pour plusieurs raisons. La plus pesante est la complexité de la modélisation des objets opératifs dans le système O.S.L.O. Cette complexité apparaît à deux niveaux :

- Au niveau de la conception : en effet, Il n'est pas aisé de définir les frontières entre les objets opératifs et les objets technologiques. Nous pouvons regrouper l'influence de tous les objets technologiques dans un seul objet opératif, ou au contraire, associer à chacun d'eux un objet opératif. Ces deux cas extrêmes sont à éviter, un juste compromis doit être trouvé par le concepteur.

- Au niveau de la réutilisabilité : comme pour les objets mission, les objets opératifs sont très spécifiques au système modélisé. Un simple changement dans l'installation remettrait en cause la structure des objets opératifs et nécessiterait de nouveaux objets. Le principe de réutilisabilité n'est autorisé que pour les objets technologiques.

Afin de pallier cet inconvénient, nous avons choisi une description plus générale des objets d'un système de production. Ainsi, nous avons proposé une structure unique pour les objets missions, les objets opératifs et les objets technologiques.

Comme le but de la modélisation d'un système est, dans notre cas, la simulation, nous regroupons les trois types d'objet sous le nom d'***objet de simulation***. Ainsi tout objet de simulation, pourra être défini comme l'un ou l'autre des types d'objets ou même comme une composition de ces types.

II.6.1/ Description de l'objet de simulation

II.6.1.1/ La structure de l'objet

La structure d'un objet est entièrement décrite par ses attributs (variables). Ces attributs peuvent être classés en trois catégories :

♦ Les **caractéristiques** décrivent les données statiques propres à l'objet. Ces caractéristiques peuvent être modifiées selon les besoins de l'application. Ces variables sont souvent utilisées pour représenter des seuils (niveau minimum, température maximale, ...), des valeurs de référence, ...etc.

♦ Les **entrées/sorties** regroupent les informations reçues ou fournies par l'objet. Contrairement aux caractéristiques de l'objet, ces informations sont des données dynamiques. Par exemple, les débits d'entrée et de sortie d'un objet réservoir. Ces variables sont le support de communication entre les objets, elles permettent la connexion entre les différentes parties du système ou de rendre compte de certaines caractéristiques de la matière d'oeuvre, comme le débit.

♦ Les **variables internes** décrivent les données dynamiques propres à l'objet. Elles sont fonction du comportement de l'objet face aux entrées/sorties, par exemple, le niveau ou le contenu du réservoir. Ces variables représentent le support de description de l'état de l'objet qu'elles représentent. Elles permettent en effet, elles aussi, de rendre compte de certaines caractéristiques de la matière d'oeuvre.

II.6.1.2/ Le comportement de l'objet

Le comportement de l'objet de simulation est donné par un ensemble de méthodes qui agissent directement sur sa structure. Ce comportement peut être divisé en deux parties complémentaires:

♦ Les ***contraintes statiques*** [BEK 91] ou ***règles*** assurent la cohérence de relations physiques ou de lois statiques. Par exemple, pour une résistance, la tension est égale au produit de la valeur de sa résistance par le courant qui la traverse. Pour une vanne, le débit est proportionnel à la pression et à son ouverture.

L'état d'un objet de simulation est donné par l'ensemble des valeurs affectées à ses variables. De ce fait, nous distinguons deux états possibles : stable et instable.

Un objet de simulation est dit stable si les contraintes statiques sont satisfaites. Chaque fois qu'une nouvelle valeur est affectée à l'une des variables, l'objet en question passe à l'état instable et il faut vérifier si ses contraintes sont toujours satisfaites.

♦ Les ***actions*** de l'objet représentent son comportement dépendant du temps. Un exemple d'action est l'évolution du niveau d'un produit quelconque dans un réservoir. A ces actions, on associe des conditions d'activation et de désactivation. Ces conditions sont spécifiées par des tests logiques sur les variables de l'objet. Par exemple, l'action qui correspond à l'évolution du niveau n'est déclenchée que si les débits d'entrée et de sortie du réservoir en question sont différents.

II.6.2/ Les liaisons entre les objets

Pour décrire un système quel qu'il soit, il ne suffit pas de décrire ses différentes parties. Il faut aussi définir les relations entre ces parties et un moyen de communication. Selon la description des objets proposés plus haut, il est évident que les objets communiquent par leurs variables d'entrée et de sortie. Mais la définition des entrées et des sorties n'est pas aussi simple qu'il paraît.

En effet, la connexion physique entre la sortie d'une vanne et l'entrée d'un réservoir semble évidente. Toutefois, la modélisation d'une telle relation peut poser un certain nombre de problèmes : l'évolution d'un produit dont la température est un facteur déterminant nécessite l'utilisation de connexions plus complexes qu'un produit où seul le débit est pris en compte. Les objets de simulation possèdent des variables pouvant caractériser la matière d'oeuvre, or la matière d'oeuvre évolue d'un objet à l'autre. Ceci implique que chacune des caractéristiques doit suivre cette évolution, et chaque objet doit avoir suffisamment de variables (et de connexions) pour ne pas perdre les informations utiles.

Un autre problème de connexion apparaît lorsque l'entrée ou la sortie n'est pas physiquement bien délimitée, par exemple, l'éclairement d'une lampe dans un espace ou l'échange thermique entre l'environnement et un réservoir.

II.6.2.1/ Un modèle de connexion

Pour représenter ces connexions, nous avons défini un nouvel objet appelé **lien**. Ce lien permet le couplage de variables entre n'importe quels objets de simulation.

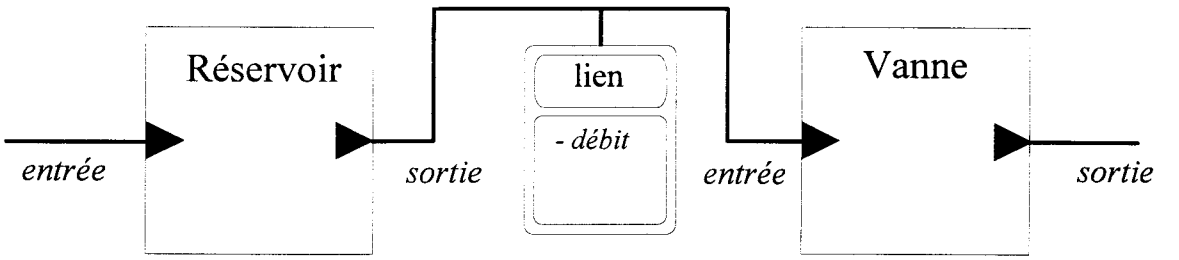


Figure II.9 : représentation des liens par des objets

Le lien est un objet qui possède une structure et un comportement. Cette propriété sera largement exploitée pour définir différents types de liens correspondant aux différentes relations qu'il peut exister dans un système. En effet, selon sa structure, il pourra transférer un nombre quelconque d'informations, de plus ces informations pourront subir un traitement spécifié par le comportement de l'objet lien.

II.6.3/ Implémentation des modèles d'objets de simulation

II.6.3.1/ Une bibliothèque hiérarchisée

Le découpage d'un système en composants élémentaires, en vue de sa simulation, nous conduit naturellement à définir des types d'objets de simulation que l'on retrouve à plusieurs endroits du système et dans différentes applications. A chacun de ces types, nous définissons un modèle qui est spécifié par une classe. Ainsi à chaque type correspond une classe, et l'ensemble des classes forme une bibliothèque de modèles. Chaque classe peut générer un nombre quelconque d'instances d'objets de simulation correspondant à son modèle.

II.6.3.2/ Composition d'objets

La composition d'objets consiste à assembler plusieurs objets de simulation pour en définir un nouveau plus complexe. Cette technique est utilisée pour deux objectifs différents. La composition permet d'une part, de structurer un gros système décomposable en sous-systèmes et, d'autre part, de dupliquer des objets complexes dans le cas où un système possède plusieurs sous-systèmes identiques.

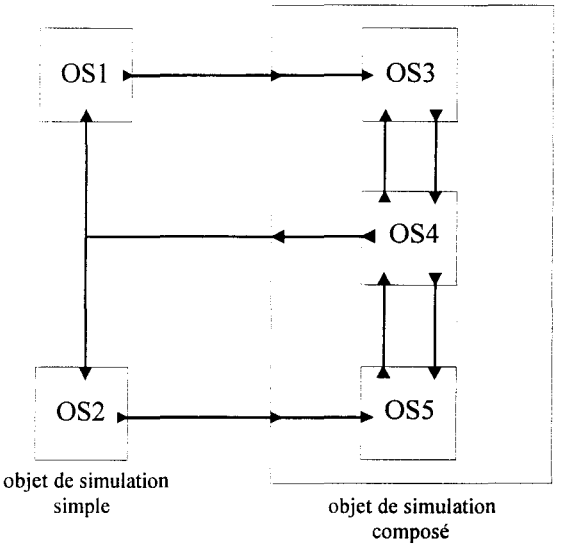


Figure II.10 : composition d'objets

La structure d'un objet composé est identique à celle d'un objet de simulation simple. Il possède des entrées et des sorties, des variables internes et un comportement. L'utilisation des liens permet de définir des objets composés. Il suffit pour cela de désigner le groupe d'objets liés entrant dans la composition et de définir les entrées et les sorties du nouvel objet composé. Chaque entrée et chaque sortie de l'objet composé sont reliées (par un lien) aux entrées et sorties des objets internes (figure II.10).

II.7/ PASSAGE DU SYSTEME O.S.L.O. AU MODELE D'OBJET DE SIMULATION

Le passage du système O.S.L.O. à la notion d'objet de simulation implique deux changements. D'abord, dans la phase de description, les objets missions, opératifs et technologiques sont considérés au même niveau. Il n'y a donc plus de notion de maître et d'esclave. Le deuxième changement apparait au niveau de l'implémentation et consiste à associer une structure informatique unique pour tous les types d'objets.

II.7.1/ Modélisation des objets technologiques

Même si sa fonctionnalité reste la même, le statut de l'objet technologique passe de l'état esclave, manipulé par les objets missions à l'état d'objet de simulation

indépendant et géré de la même manière et au même niveau que les objets missions et les objets opératifs. Un objet technologique est ainsi défini comme un objet ayant une existence physique dans le système et un comportement traduit, sur machine, sous forme de contraintes à résoudre par le moteur de simulation M.A.R.C. (paragraphe II.8).

II.7.2/ Modélisation des objets opératifs

La décomposition en objets opératifs est effectuée en fonction des tâches opératives, tout en gardant un caractère générique.

Cette décomposition est obtenue à partir de l'objet opératif principal de l'application auquel est associé différents objets opératifs accomplissant chacun une tâche opérative bien définie. Afin d'obtenir la tâche principale désirée, ces objets opératifs peuvent se servir d'autres objets opératifs et ainsi de suite. Cette logique peut aller jusqu'à associer un objet opératif à chaque objet technologique.

La difficulté réside dans le fait qu'il n'y a pas de stratégie standard, il s'agit d'une logique dont le résultat est spécifique à chaque application. Une décomposition trop fine n'apporte rien, si ce n'est un surcroît de travail et une structure encombrante. Par contre, supposer qu'un seul objet peut décrire toute la partie opérative, entraîne une très grande complexité et interdit l'aspect générique.

Sachant que les objets opératifs représentent deux catégories d'objets : les objets de transfert et les objets passifs, il convient de définir un objet opératif comme un objet de simulation ayant lui aussi une existence physique. Son comportement serait celui de l'objet de transfert (canalisation) ou de l'objet passif (réservoir).

II.7.3/ Modélisation de la matière d'oeuvre

La matière d'oeuvre est vue comme un objet de simulation à part entière ayant ses propres paramètres (exemple, température, couleur, volume, ...etc.) et son propre comportement.

De ce fait, un réservoir est désormais représenté par deux entités distinctes :

- un objet de simulation "*Conteneur*" de la matière d'oeuvre,

- un objet de simulation "*Contenu*" (la matière d'oeuvre elle même).

Afin d'assurer un fonctionnement global cohérent, ces deux objets communiquent entre eux par l'intermédiaire des objets liens. En effet, le comportement de la matière d'oeuvre dépend énormément des paramètres de son conteneur. Le schéma de la figure II.11 illustre la structure d'un réservoir vue par la notion d'objet de simulation.

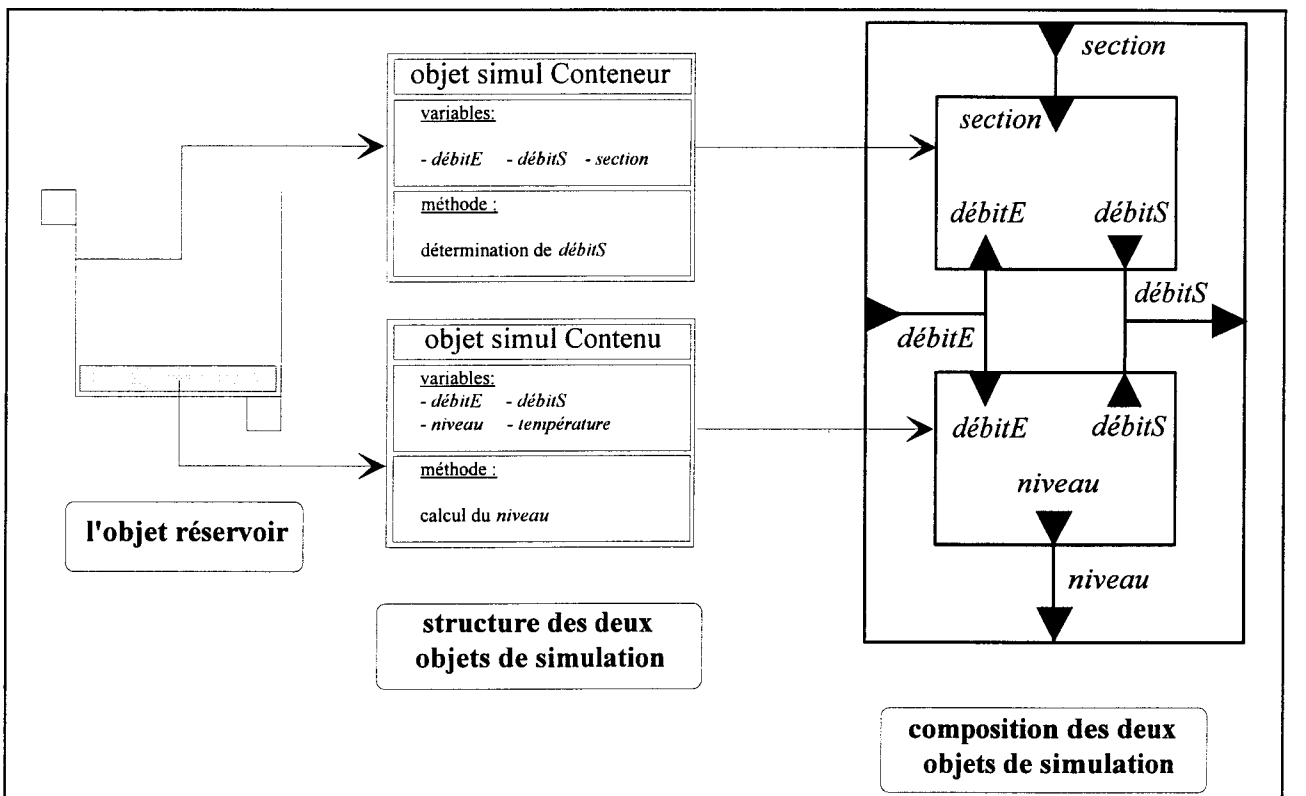


Figure II.11 : représentation de la matière d'oeuvre par objet de simulation

De plus, la propriété de composition nous offre la possibilité d'encapsuler ces deux objets liés dans un objet composé appelé *réservoir*.

II.7.4/ Modélisation des missions

La décomposition en objets mission reste valable même si elle ne s'apparente plus à celle des objets opératifs. En effet, un objet mission a pour but d'exécuter une tâche du système de production.

Une mission peut déclencher une ou plusieurs autres missions. Nous obtenons alors une organisation hiérarchique des missions correspondant à l'organisation complète du système de commande.

Bien que les objets missions soient spécifiés de la même manière que les objets opératifs ou technologiques, leur description reste particulière du fait de leur fonctionnalité qui est de définir un séquencement d'instructions. Ce séquencement correspond au comportement de la mission. Si l'on peut réutiliser des réservoirs et des vannes dans un grand nombre d'applications, les missions quant à elles, sont très rarement réutilisables car chaque application possède des objectifs différents.

En ce sens, nous avons adopté deux solutions différentes : modélisation par des objets de simulation ou modélisation par un outil formel.

II.7.4.1/ Modélisation des missions par des objets de simulation

Dans ce cas, les missions sont représentées par des objets possédant des entrées et des sorties connectées aux autres objets de simulation par l'intermédiaire des liens. De même que pour tout autre objet de simulation, une mission peut avoir des variables internes et pourquoi pas, certaines caractéristiques (constantes de temps, seuils, ...etc). Son comportement sera décrit par ses activités, ses conditions d'activation et de désactivation et enfin ses contraintes statiques.

Dans un souci de réutilisabilité, il peut être défini des missions élémentaires, tels que temporisation, détection de seuil, boucle tant que, ...etc, puis par composition de ces objets obtenir des missions plus globales.

II.7.4.2/ Modélisation des missions par un outil formel

Cette technique consiste en fait à redéfinir un langage de spécification de commande. L'utilisation du Grafcet ou des réseaux de Pétri semble tout indiquée pour spécifier la commande d'un système. Pour cela, des travaux dans l'équipe sont dédiés à la conception d'un outil permettant de définir les différents objets (Etapes et Transitions) pour avoir à notre disposition un moyen générique de définir des missions.

II.7.5/ Exemple d'illustration (le Malaxeur)

Reprenons l'exemple du malaxeur décrit précédemment. La décomposition en objets nous conduit à définir la liste des objets de simulation suivants : 3 réservoirs (A, B et C), 2 vannes (V1 et V2), une conduite (CTE), une bascule de pesage (BP), un réservoir malaxeur, un convoyeur, trois capteurs, un tapis d'amenage et les briquettes. Il s'ensuit la liste desdifférents modèles d'objets représentés dans le tableau suivant :

Objet	Structure	Comportement
Le réservoir	Caractéristique : *niveauMax* Entrée/Sortie : *débitE, débitS* Variable interne : *niveau*	Activité : Evolution du *niveau* en fonction des débits d'entrée et de sortie.
La conduite	Variables internes : *débitE1,débitE2, débitS*	Contrainte statique : débitS = débitE1 + débitE2
La Vanne	Caractéristique : *section* Entrée/Sortie : *débit* Variable interne : *ouverture*	Contrainte statique : le *débit* est proportionnel à *l'ouverture*.
La Bascule	Caractéristique : *poidsMax* Entrée/Sortie : *quantité* Variable interne : *poids*	Contrainte statique : le *poids* est proportionnel à la *quantité*.

Objets	Structure	Comportement
Le réservoir Malaxeur	Caractéristique : *niveauMax* Entrée/Sortie: *débitE, débitS, produit* Variables internes : *niveau, angleMelange, anglePivotement.*	Activité : Evolution du *niveau* en fonction des débits d'entrée et de sortie, de l'arrivage des briquettes, et de l'angle de pivotement.
Le convoyeur	Caractéristique : *capacitéMax* Entrée/Sortie : *produitE, produitS* Variable interne : *listeProduits.*	Activités : - orientation des produits entrants ou sortants, - mise à jour de *listeProduits.*
Tapis d'amenage	Entrée/Sortie : *produit* variable interne : *généreUnProduit*	Contrainte statique : [*généreUnProduit*] Alors: [*produit* = un nouveau Produit]
Briquette	Caractéristique : *positionDest* Variable interne : *position, vitesse*	Activité : évolution de la *position* en fonction de la *vitesse* et de la *position destination*
La mission ChargtBascule	Caractéristiques : *seuilA, seuilB* Entrées/Sorties : *débutActivité, finActivité, activité, quantité, ouvertureA, ouvertureB*	Contraintes statiques : lancement du processus de la mission si *débutActivité* est à l'état *true* et que la mission est désactivée (*activité = false*). le contenu et la gestion du processus sont décrits ultérieurement.

Ensuite, il faut construire les différentes missions du cahier des charges déjà défini. On trouve ainsi :

- mission malaxeur, mission chargtBascule, mission chargtAB,

mission chargtBr, mission moteur, mission vidange.

En plus, on définit une mission d'attente pour gérer la synchronisation, autrement dit, la mission qui représente l'étape d'attente dans le grafcet.

La figure II.12 montre la structure de la partie opérative du système malaxeur. Les liens entre les variables sont représentés par des traits.

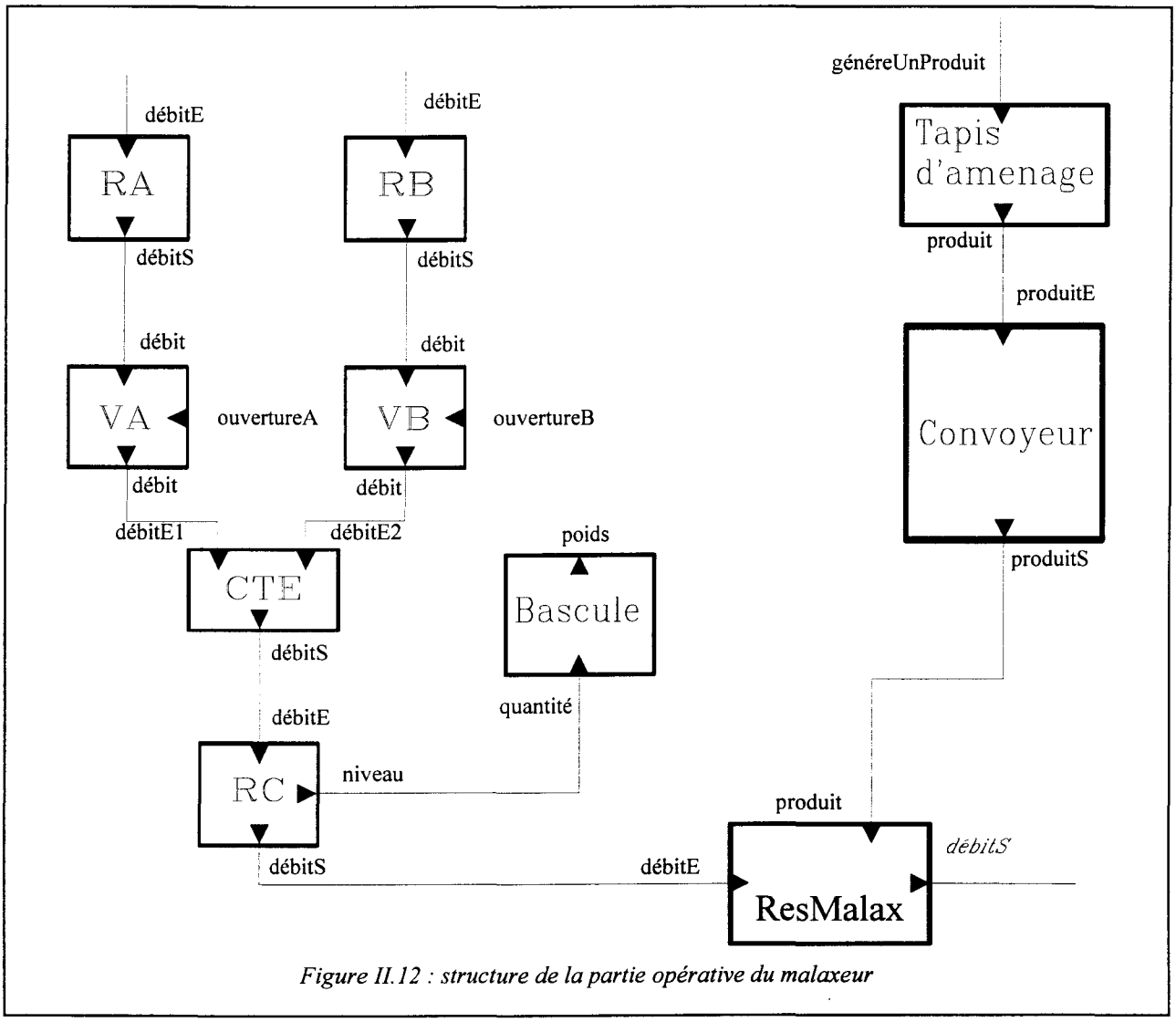


Figure II.12 : structure de la partie opérative du malaxeur

Dans la figure II.13, on trouve tous les objets de simulation modélisant les différentes missions. La mission malaxeur a pour rôle de gérer la synchronisation des missions élémentaires.

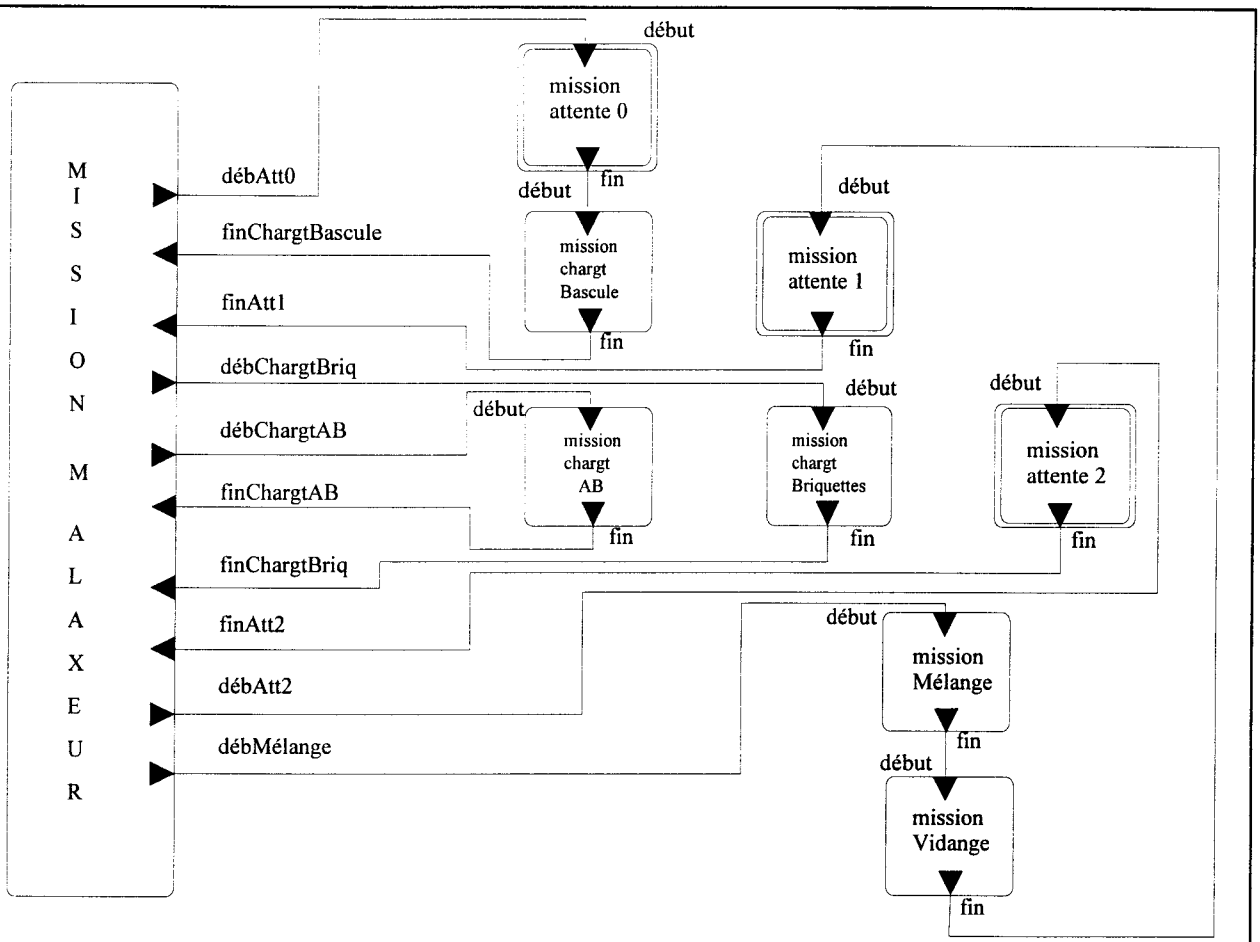


Figure II.13 : structure de la partie commande

II.8/ MISE EN OEUVRE DE LA SIMULATION [MAR 91]

La mise en oeuvre de la simulation est assurée par un moteur à résolution des contraintes. Ce moteur est un processus qui doit effectuer plusieurs tâches :

- vérifier toutes les contraintes des objets,
- lancer toutes les actions des objets pendant un intervalle de temps,
- et enfin, permettre les interventions extérieures de l'utilisateur.

II.8.1/ Notion de stabilité et de propagation

La notion de résolution de contraintes apporte une certaine instabilité passagère des variables impliquées dans la contrainte. On peut dire que le système (réseau de contraintes) est stable si toutes les contraintes sont vérifiées. Dès qu'une variable contrainte est modifiée, il faut vérifier la contrainte associée. La résolution de cette contrainte peut alors modifier une ou plusieurs autres variables qui peuvent avoir chacune une ou plusieurs contraintes.

Ainsi, il faut veiller à ce que la propagation des contraintes ne soit pas infinie.

II.8.2/ Résolution des contraintes

Pour gérer les contraintes, on construit une liste des objets de simulation instables. Ceci présente deux avantages : ne pas vérifier toutes les contraintes du système (gain de temps) et la possibilité de tester la stabilité (liste vide).

Pour construire cette liste, on utilise l'attachement procédural qui consiste à ajouter un objet de simulation dès que sa structure est modifiée. C'est à dire, dès qu'une variable est affectée par une nouvelle valeur. Si l'affectation concerne une variable contenant un lien, tous les objets en relations sont alors ajoutés à la liste.

La résolution des contraintes consiste donc à sortir un à un les objets de la liste en vérifiant ses contraintes, ceci jusqu'à la stabilité. Cette technique s'inspire des travaux de D.Waltz [WAL 75].

II.8.3/ Les actions

Parmi les contraintes, certaines sont utilisées pour déclencher des actions (II.6.1.2). Dans le cas de la cuve par exemple, une contrainte [1] consiste à vérifier si les valeurs des débits d'entrée et de sortie sont différentes pour lancer ou non l'action correspondant à l'évolution temporelle du niveau [2].

[1] ***contraintes***

(débitE - débitS = 0) ifTrue: [self action] ifFalse: [self inaction]

[2] ***action: pas***

*niveau := niveau + ((débitE - débitS) * pas)*

Ce principe est aussi bien appliqué pour les objets de simulation de la partie opérative que pour ceux de la partie commande.

II.8.4/ Gestion des missions

Vis-à-vis du moteur de simulation, la gestion des objets missions est identique à celle des autres objets de simulation. En effet, chaque mission définit trois variables d'entrée / sortie : *débutActivité, finActivité et activité*. Une contrainte statique identique à toutes les missions consiste, lors de sa résolution, à vérifier l'état de la variable *débutActivité (débutActivité = true)* et celui de la mission *(activité = false)*.

Dans ce cas, un processus dont le contenu est spécifique à chaque mission, est lancé en parallèle au moteur de simulation. Ce processus contient une séquence de règles et/ou d'actions. Le processus correspondant à la mission ChargtBascule par exemple, est écrit de la manière suivante :

tâcheMission

```
ouvertureA := true.                 " ouverture de la vanneA "
quantité < seuilA whileFalse: [Processor yield].
ouvertureA := false.                " fermeture de la vanneA "
ouvertureB := true.                 " ouverture de la vanneB "
quantité < seuilB whileFalse: [Processus yield]
ouvertureB := false                 " fermeture de la vanneB "
```

Processor yield est une instruction qui appartient à Smalltalk-80 [GOL 83] et permet de remettre le contrôle au seul processeur de la machine afin d'exécuter les autres processus suspendus.

La figure II.14 illustre les organigrammes représentant le processus du moteur de simulation et celui de la mission ChargtBascule. La gestion des processus est mise en oeuvre selon la structure FIFO. Lorsqu'un processus prend le contrôle, il reconstitue son contexte et continue à exécuter sa tâche à partir du point ou il à été suspendu. Dès qu'une instruction de remise de contrôle est rencontrée dans le corps de la mission, le processus est de nouveau suspendu. Une fois que la tâche est entièrement exécutée, la mission est désactivée (*finActivité = true*). Ceci permet d'informer, par l'intermédiaire des objets liens, les autres objets de simulation de son état et d'activer ainsi une autre mission par exemple.

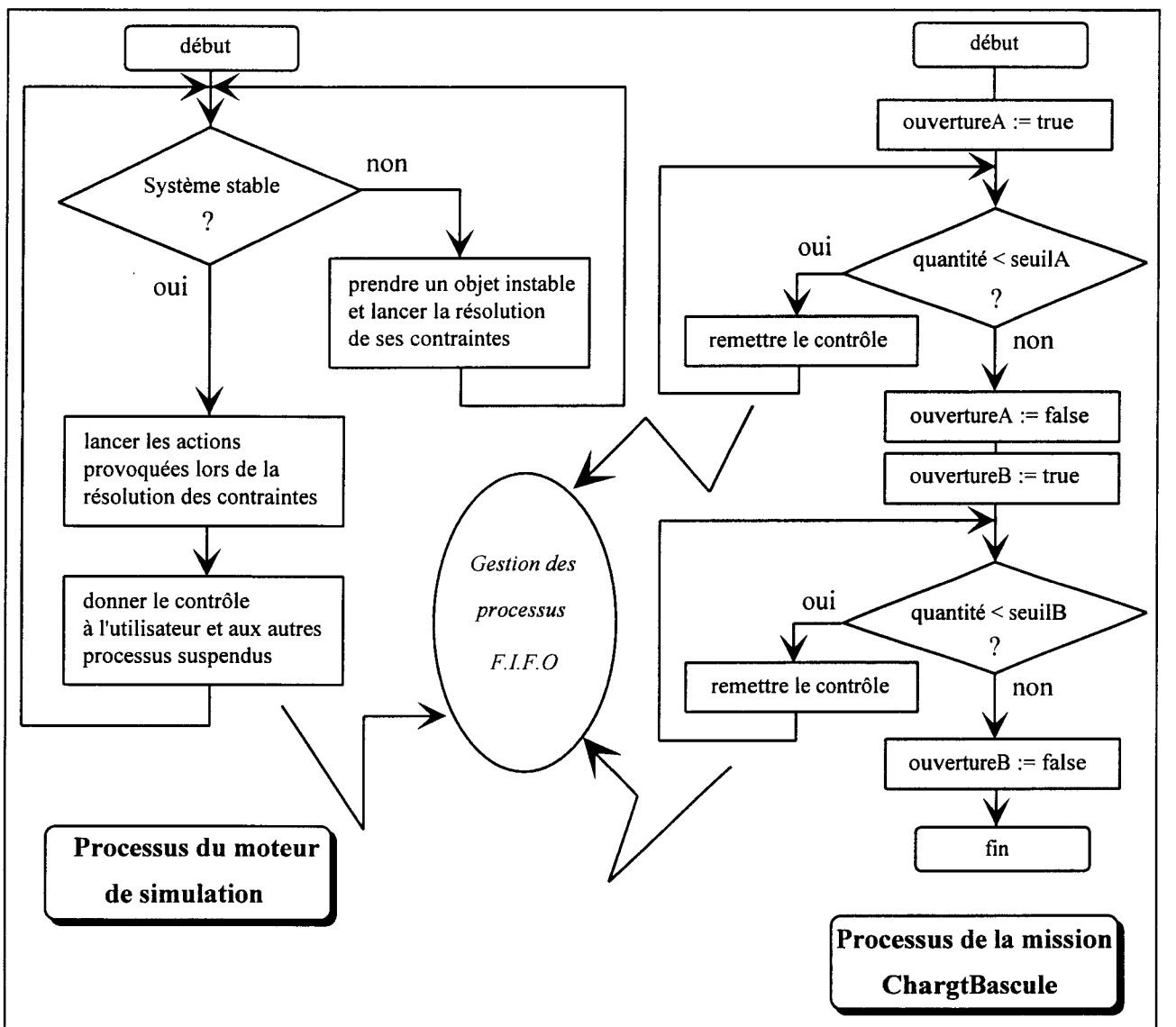


Figure II.14 : les organigrammes des processus de simulation

II.9/ CONCLUSION

Dans ce chapitre, nous avons donné la décomposition orientée objet d'un système industriel correspondant au système O.S.L.O. Cette décomposition comporte des objets opératifs dont le rôle est de représenter le comportement de la matière d'oeuvre, des objets missions contrôlant les différentes missions du processus. Les interactions entre ces objets missions et opératifs s'effectuent par les objets technologiques.

Une des principales caractéristiques des éléments de cette décomposition est leur généricité.

Cette technique présente toutefois deux inconvénients :

Le premier concerne la frontière entre la partie opérative et la partie technologique. En effet, si la partie technologique est facilement repérable dans un système, il n'en est pas de même pour la partie opérative. Certains objets possèdent les deux composantes à la fois (opérative et technologique), ce qui nous amène à définir deux objets distincts correspondant à un seul objet physique du système.

Le second inconvénient est le manque de modélisation de la connectique entre les différents objets.

Nous avons présenté dans une deuxième partie, une généralisation de la modélisation des objets par l'introduction d'objets de simulation et des objets liens assurant leur communication. En effet, un objet de simulation est une structure plus complète pouvant être composée à la fois d'une partie opérative, d'une partie technologique et d'une partie commande. Ceci permet, par exemple, de définir un comportement défaillant pour un objet technologique.

Une fois le modèle défini, la simulation est mise en oeuvre par un moteur à résolution des contraintes. Ce moteur considère le modèle comme un ensemble d'acteurs ayant chacun un rôle dans la simulation.

Chaque acteur interagit avec l'environnement grâce aux objets liens afin d'assurer le fonctionnement global du système simulé. Dans le cas de la simulation interactive, on peut distinguer deux types d'acteurs : on trouve d'une part les acteurs correspondant aux objets de simulation et d'autre part l'acteur principal qu'est l'opérateur humain.

Comme tout autre acteur, l'opérateur a un rôle important qui consiste à piloter l'ensemble du système. Il sera donc amené, selon le contexte, à prendre part à la simulation en accédant à n'importe quel point et à tout moment (pendant ou en dehors de la simulation).

De ce fait apparaît l'intérêt de l'interface de simulation dont le rôle peut être définie en deux points importants :

- la présentation ergonomique des informations afin que l'utilisateur appréhende facilement le système et sache à tout moment ce qui se passe à l'intérieur de la machine et ce qu'il peut faire pour réaliser ses buts,

- la prise en compte et la gestion des actions de l'opérateur, soit lors de la conception du modèle, soit pendant la simulation.

Nous verrons dans les chapitres suivants, les éléments nécessaires liés à la conception des interfaces homme-machine.

Chapitre III

Conception des Interfaces Utilisateurs

Conception des Interfaces Utilisateurs

III.1/ INTRODUCTION

Quand on parle d'application interactive, il faut distinguer l'application abstraite de l'interface utilisateur qui a pour rôle de gérer l'interaction d'une manière générale.

De ce fait, lors de la conception d'une simulation interactive, il existe deux types d'intervenants :

- les concepteurs de l'application ayant des connaissances sur la modélisation des systèmes et sur le fonctionnement du moteur de simulation,
- les concepteurs de l'interface utilisateur qui sont, par contre, spécialisés dans les problèmes d'interaction homme-machine et l'ergonomie des logiciels d'une manière générale.

La conception de l'interface utilisateur occupe une part très importante dans la réalisation d'un projet interactif. C'est même un projet en soi auquel il faut penser dès le début de la réalisation. Comme dans tout projet, une méthodologie est indispensable : elle guide les phases d'analyse, de spécification et d'évaluation.

III.2/ UNE PHASE D'ANALYSE

Cette phase constitue la première partie de la conception. Elle a pour objet de définir les grands objectifs du système, de les spécifier qualitativement et quantitativement, grâce à une analyse des tâches et de l'activité de l'utilisateur. Quel

que soit le type de l'utilisateur auquel le système est destiné, deux problèmes se posent lors de son interaction avec l'interface et qu'il faut résoudre au moment de la conception de cette dernière (figure III.1) :

- le premier problème concerne l'interaction dans le sens (*utilisateur >> interface*). Dans ce cas, l'utilisateur essaie d'exploiter ses capacités intellectuelles et son savoir faire afin d'inférer une séquence d'opérations ou d'actions sur les objets du modèle mental qu'il s'est fait par l'utilisation. L'objectif de la conception est de rapprocher le modèle conceptuel des potentialités d'appréhension de l'utilisateur (figure III.2);

- le deuxième problème se pose lors de l'interaction dans le sens (*interface >> utilisateur*). Le rôle de l'interface est de donner une image, la plus représentative de l'application, en respectant les stratégies d'apprentissage de l'opérateur. L'objectif dans ce cas, est de faciliter l'acquisition des connaissances, c'est-à-dire d'élargir le modèle mental de l'utilisateur aux potentialités du modèle conceptuel (figure III.2).

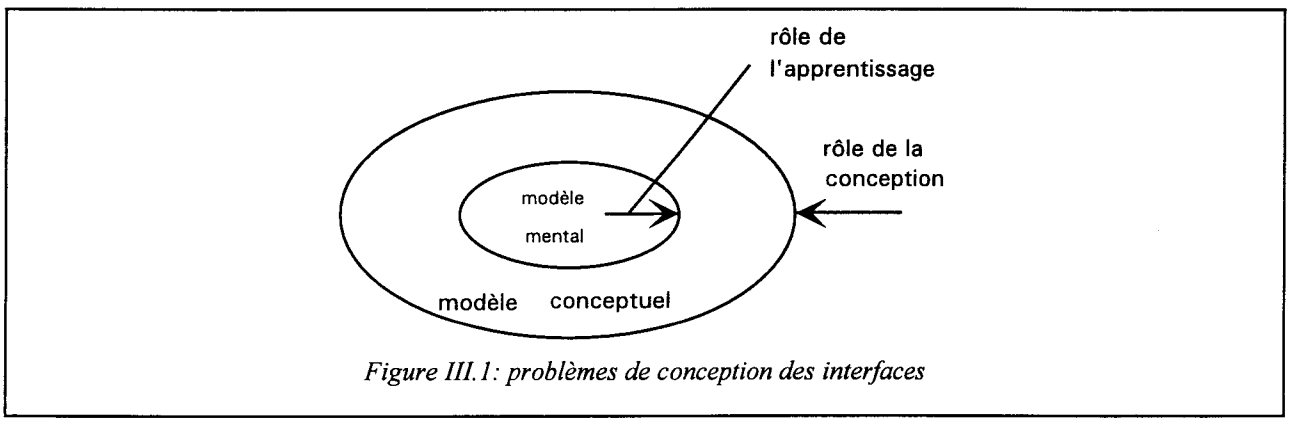


Figure III.1: problèmes de conception des interfaces

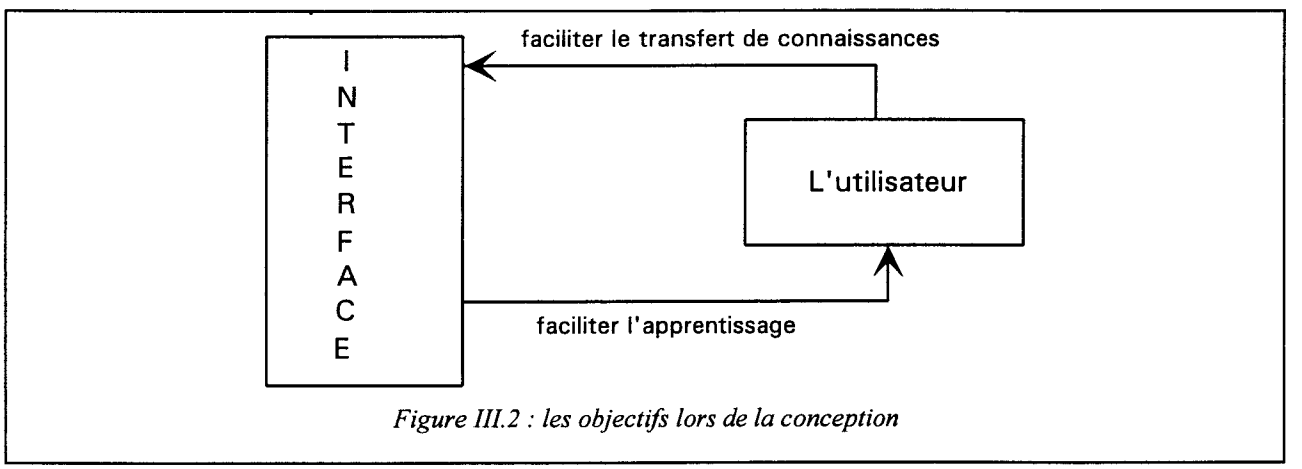


Figure III.2 : les objectifs lors de la conception

III.3/ REGLES DE CONCEPTION DES INTERFACES UTILISATEURS

L'interface utilisateur joue un rôle primordial dans le cas des applications interactives et sa construction prend un temps beaucoup plus important que celui de l'application elle même. Ceci est dû aux diverses exigences posées afin d'avoir un produit convivial et réutilisable; ce qui nous amène à introduire quelques règles méthodologiques pour la conception d'interfaces utilisateurs.

III.3.1/ Séparer la conception de l'interface de celle de l'application

Afin d'éviter que l'interface ne soit utilisable que pour l'application en cours du développement ou déjà développée, il est préférable de séparer la conception de l'interface de celle de l'application. Cette méthodologie possède de multiples avantages comme :

- La possibilité d'adapter la même interface à une variété d'applications : une interface de simulation peut être utilisée pour simuler des processus par lots, électriques ou encore de transports automatisés.

- Une meilleure maintenance : en séparant l'interface de l'application, on réduit la zone de maintenance. Un problème peut apparaître soit au niveau de l'application soit au niveau de l'interface. Dans le premier cas, une bonne interface utilisateur doit contribuer à la détection de l'anomalie et la mise en oeuvre d'une séquence d'opérations permettant de résoudre le problème. Dans le second cas, le problème devient plus facile à détecter du fait qu'on se limite, au niveau diagnostic, à la partie interface sans se préoccuper de l'application.

III.3.2/ Concevoir l'interface en parallèle avec l'application

Cette deuxième règle a un grand avantage aussi bien pour l'interface que pour l'application. Les exigences lors de la conception de l'interface utilisateur nous amène à développer une application ayant une structure très modulaire donc une architecture très homogène. En contre partie, la conception interactive de l'interface nous donne

l'occasion de tester au fur et à mesure du développement de l'application, les qualités ergonomiques de l'interface utilisateur.

III.4/ SPECIFICATION DES INTERFACES UTILISATEURS [COU 90] [MEN 91]

III.4.1/ Introduction

Du fait de l'absence de stratégies standards de spécification des interfaces utilisateurs, la méthode classique, issue des travaux de MORAN [MOR 81] et de ses successeurs, consiste à définir l'interface utilisateur comme un langage, avec une structure en couches grammaticales, faisant passer du monde conceptuel et sémantique au monde perceptuel, pour se traduire par un échange physique. On retrouve ainsi quatre niveaux de spécifications :

- spécification conceptuelle,
- spécification sémantique,
- spécification syntaxique,
- spécification lexicale.

III.4.2/ Spécification conceptuelle

Cette spécification sert à décrire le modèle de l'application de manière conceptuelle et les fonctions que l'on peut faire sur ce modèle. En d'autres termes, le modèle conceptuel sert de notice d'utilisation de l'application, utile à l'utilisateur afin qu'il puisse comprendre le fonctionnement de l'outil et inférer les différentes actions à mettre en oeuvre pour réaliser ses objectifs.

La spécification conceptuelle regroupe les éléments suivants :

- les types d'objets manipulés par l'application,
- les relations entre objets,

- les opérations sur les objets, leurs attributs et leurs relations.

Le modèle conceptuel ainsi créé a pour objet d'aider l'utilisateur à comprendre le système et à en déduire ce qu'il peut en faire.

En effet, il faut distinguer le modèle conceptuel créé par le concepteur de l'interface et la représentation mentale que s'en fait l'utilisateur. Afin de réduire le temps d'élaboration du modèle mental et que ce modèle soit le plus proche de celui du modèle conceptuel, le concepteur d'interface se base sur le savoir antérieur des utilisateurs ciblés. Ce qui se traduit en d'autres termes, par l'adoption d'une *"métaphore"*.

III.5.2.2/ Les métaphores

Une métaphore est une transposition analogique d'un domaine sur un autre domaine. On retrouve ainsi la métaphore de la machine à écrire pour les débutants en traitement de texte, celle du tableau de bord pour les pilotes, et celle du bureau, très utilisée actuellement par la plupart des interfaces utilisateurs de type WIMP (Windows, Icons, Menus, Pointers).

Dans le cas de la métaphore du bureau, l'écran représente le plan d'un bureau. Les icônes, dessins représentant des objets du système manipulables par l'utilisateur, sont en fait des métaphores élémentaires. Elles peuvent représenter aussi bien des programmes, que des données, des documents, des graphiques, des périphériques, des opérations ou des objets définis par l'utilisateur.

La souris, manipulée par la main de l'utilisateur, permet de désigner les objets sur l'écran, de les déplacer sur le plan du bureau. C'est la manipulation directe qui n'est autre que la transcription métaphorique du geste de la secrétaire qui manipule les objets sur son bureau.

Et pour revenir à la définition générale, une métaphore fournit un cadre conceptuel permettant la compréhension des nouveaux éléments. Pour éviter la sélection des métaphores inadéquates et pour faciliter l'apprentissage et le transfert, la conception doit proposer des métaphores adaptées.

III.4.3/ Spécification fonctionnelle

La spécification fonctionnelle constitue le niveau sémantique de la conception de l'interface utilisateur. Elle vient compléter le niveau conceptuel en faisant une description fonctionnelle des opérations décrites et de leurs effets sur l'application (figure III.4).

La spécification sémantique ou fonctionnelle des commandes est décrite en langage naturel. Elle est indépendante de la méthode utilisée pour les réaliser, et des dispositifs qui seront pris en compte aux niveaux inférieurs.

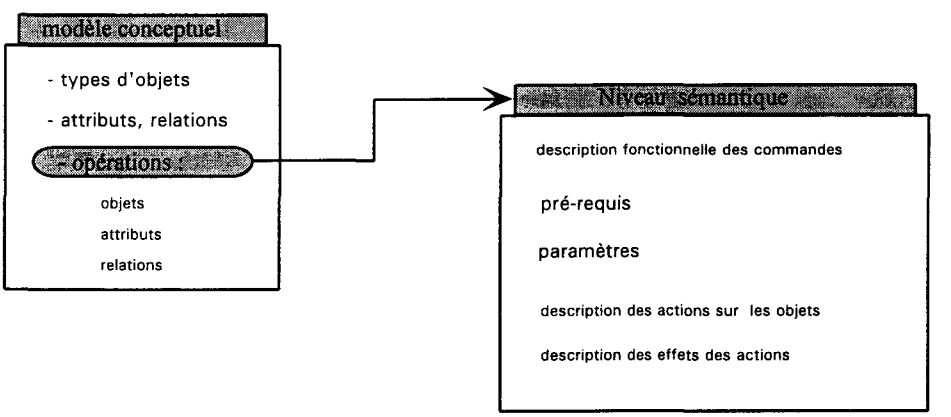


Figure III.4 : spécification fonctionnelle

III.5.3.1/ Considérations ergonomiques

Lors de la spécification sémantique, des choix ergonomiques judicieux doivent être adoptés. Ces choix sont répartis en deux catégories :

Choix ergonomique des entrées

Dans ce cas, on spécifie les commandes à proposer à l'utilisateur. Ce choix est basé sur un certain nombre de critères à prendre en compte lors de la conception :

- universalité : l'existence de commandes à caractère universel qui se trouvent sur tous les systèmes,
- cohérence : la possibilité de généraliser une commande sur des objets différents dans des contextes différents,

- flexibilité : la possibilité d'une flexibilité sémantique dans le système de commande permettant d'arriver au même but par différentes méthodes au choix de l'utilisateur.

Choix ergonomique des sorties

Elle doit permettre à l'utilisateur de comprendre et d'interpréter facilement l'effet des commandes choisies. Dans ce cas, on doit tenir compte de deux points importants :

- éviter l'attente inutile de l'utilisateur et présenter constamment des informations permettant de connaître l'état de l'application et de la machine,
- une gestion pédagogique des erreurs qui a un effet régulateur sur le comportement de l'utilisateur et qui joue un rôle important dans l'apprentissage.

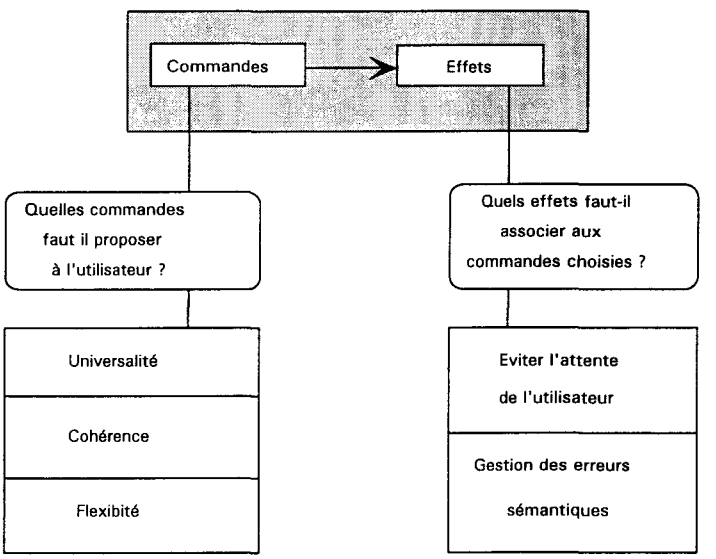


Figure III.5 : choix ergonomiques au niveau sémantique

III.4.4/ Spécification syntaxique (définition du dialogue)

La spécification syntaxique décrit la syntaxe du dialogue qui permet de réaliser les commandes définies au niveau sémantique (figure III.6). Elle concerne les points suivants :

- le style général du dialogue et son contrôle, soit par l'ordinateur soit par l'utilisateur,
- la syntaxe du langage d'interaction,
- la structuration du dialogue et de l'interaction avec les relations entrée et sortie.
- les retours d'information,
- l'ergonomie des transactions.

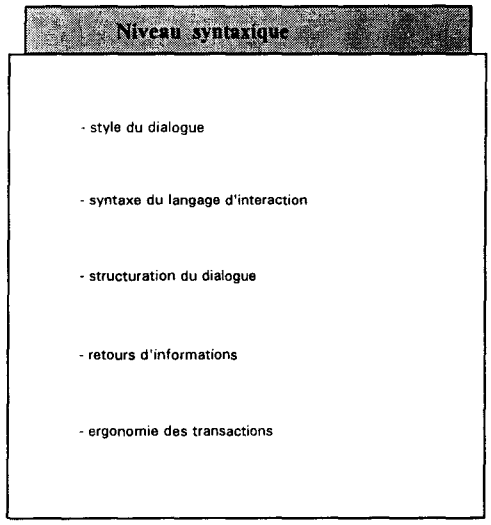


Figure III.6 : spécification syntaxique

III.4.4.1/ Style du dialogue

Plusieurs styles de dialogues ont été développés ces dernières années, en allant du dialogue par question-réponse, en passant par des dialogues utilisant le système de menus ou les langages de commande naturels, pour arriver aux interfaces introduisant le style par manipulation directe qui permet à l'utilisateur d'agir directement sur les objets visibles à l'écran. La plupart des interfaces évoluées actuelles, appelées interfaces événementielles, utilisent simultanément tous les types de dialogues.

III.4.4.2/ La syntaxe du langage d'interaction

Le langage d'interaction comprend le langage d'entrée et celui de sortie. Pour le langage d'entrée, la syntaxe est, bien sûr, spécifique au mode de dialogue choisi pour l'interaction, par contre pour le langage de sortie, elle permet de structurer les données informationnelles sur l'écran, par exemple, définir l'ordre des items dans les menus, l'emplacement exact des menus, des boîtes de dialogue et des messages, ...etc.

III.4.4.3/ La structuration du dialogue

Un *dialogue* est un ensemble de plusieurs *transactions*. Une *transaction* constitue l'élément de base d'un *dialogue*. Elle est caractérisée par un but élémentaire, comme la sélection d'un objet par exemple, et par un ensemble d'*interactions* permettant d'atteindre ce but.

L'*interaction* est définie comme un sous ensemble de la transaction, établissant la relation entre l'entrée élémentaire de l'utilisateur et la sortie correspondante qui la confirme par un affichage quelconque sur l'écran (figure III.7).

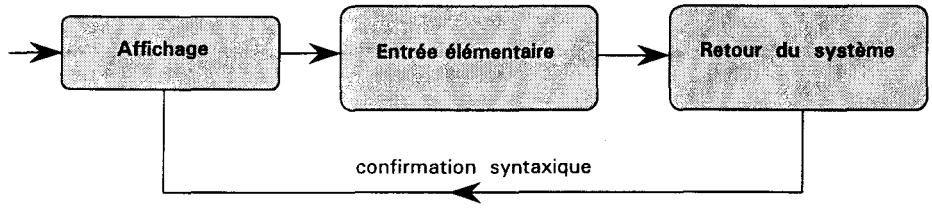


Figure III.7 : interaction élémentaire

exemple :

- *Entrée élémentaire* : passage sur objet, touche souris enfoncée / manipulation souris.
- *Sortie* : objet en vidéo inverse / déplacement curseur.

Ainsi, la sélection d'un objet sur l'écran est une transaction composée de plusieurs interactions données par le tableau suivant :

interaction	*entrée*	*sortie*
1	manipulation de la souris	déplacement du curseur
2	passage du curseur sur l'objet	l'objet s'affiche en inverse vidéo
3	touche souris enfoncée	bordure spéciale autour de l'objet

Là aussi, les règles traditionnelles d'ergonomie sont à prendre en considération. Au niveau affichage, on détermine le type de dialogue (menu, commande, ... etc.), on introduit des messages de guidage et on modifie, par exemple, la forme du curseur selon les données requises.

Au niveau des entrées, on détermine les dispositifs de commande à utiliser (clavier, souris, ...etc.). Enfin, au niveau du retour système on assure particulièrement la gestion des entrées illégales, dues aux différentes erreurs syntaxiques.

III.4.4.4/ Les retours d'informations

Le retour d'informations est nécessaire à chaque instant de l'interaction car c'est la seule chose que l'utilisateur interprète pour analyser les résultats de son action. Il permet la régulation du comportement et l'élaboration des règles d'utilisation. Ces retours d'informations sont d'autant plus utiles qu'ils apportent à l'utilisateur les outils nécessaires facilitant son interprétation, par exemple:

- les fonctions de forçage (confirmation de la commande),
- les alarmes préventives,
- les moyens de contrôle des entrées.

III.4.4.5/ Ergonomie des transactions

Au-delà du choix ergonomique fondamental du style de dialogue (menu, formulaire, langage de commande, langage naturel, objet-action) et des règles ergonomiques s'appliquant à la syntaxe du langage d'interaction, il faut assurer la cohérence des constituants élémentaires d'une interaction :

- la cohérence des affichages,
- la cohérence des entrées utilisateur,
- la cohérence des confirmations.

III.4.5/ Spécification lexicale

Du point de vue entrée, il s'agit de déterminer les actions matérielles primitives qui vont se combiner pour former l'un des éléments significatifs d'une interaction (entrée utilisateur). Et du point de vue sortie, il s'agit de décider de la présentation des objets (figure III.8). Le niveau lexical regroupe plusieurs éléments :

- dénomination des commandes, des options de menus,
- le texte des messages et des boîtes de dialogue,
- définition des formats des icônes et des curseurs,
- le codage des couleurs, polices de caractères, du clignotement, des surbrillances ... etc.

Les choix ergonomiques à ce niveau sont essentiels, car directement visibles par l'utilisateur, dès la première approche.

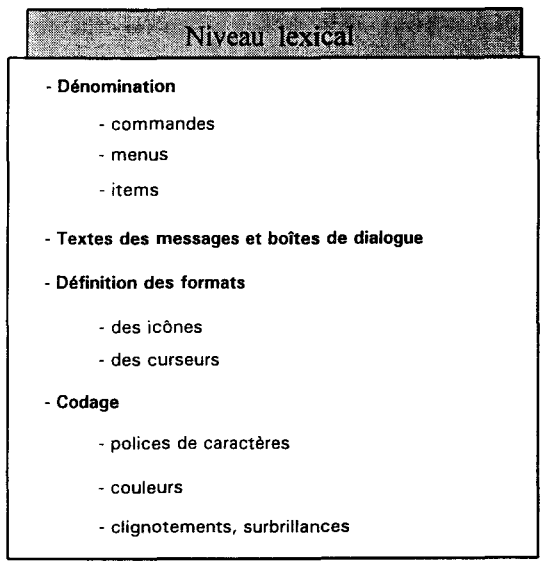


Figure III.8 : spécification lexicale

III.5.1/ Introduction

Le rôle de l'interface est de traduire le langage abstrait de l'application en un langage humain compris par l'utilisateur et vice versa. La raison pour laquelle la réalisation des interfaces utilisateurs est très difficile, est due à la diversité des applications (*domaines très variés*) et la diversité des opérateurs humains (*modélisation très complexe*). Cette difficulté est d'autant plus faible que le système interactif est structuré sous forme modulaire.

Cette modularité présente deux avantages importants : le premier est la possibilité d'adapter l'interface à des applications variées (figure III.9).

Le second avantage est celui de distinguer les mécanismes internes abstraits (fonctions dépendantes du domaine de l'application) des politiques de présentation (aspects dépendants de l'utilisateur).

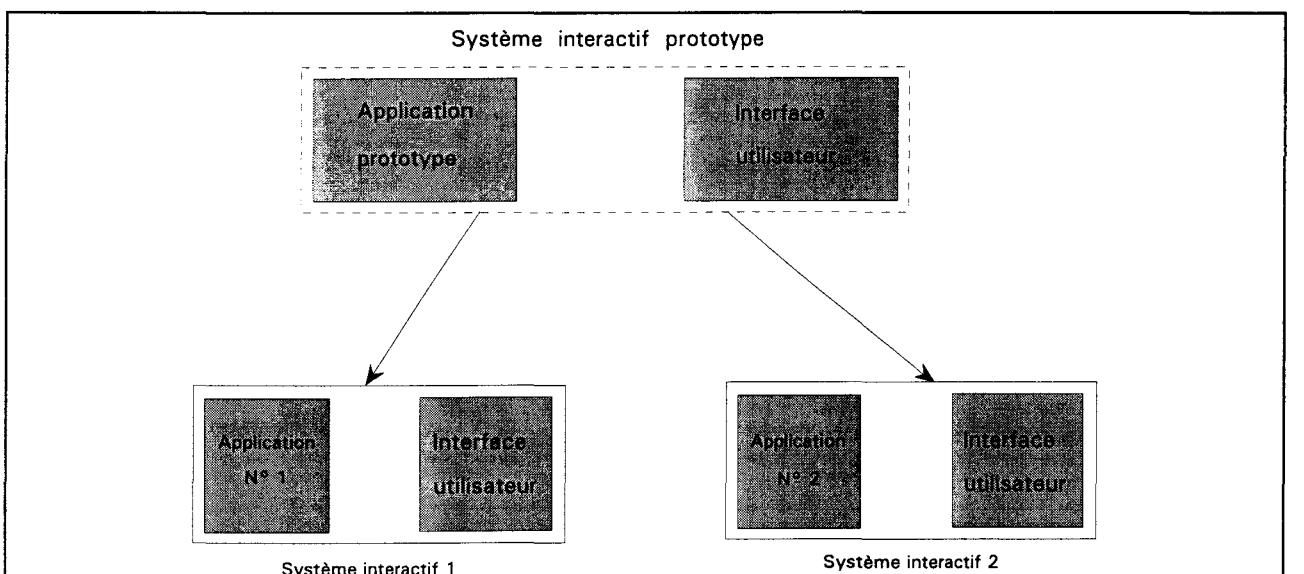


Figure III.9 : réutilisabilité des systèmes interactifs

Les principes de la modularité sont très connus par les différents concepteurs mais, leur application à la réalisation des systèmes interactifs est difficile à mettre en oeuvre en l'absence de modèle.

L'objet du modèle est de fournir au réalisateur une structure générique à partir de laquelle il est possible de construire un système interactif particulier. Cette structure constitue un composant élémentaire du système interactif.

Plusieurs modèles ont été développés et proposés au cours de ces dernières années ayant chacun sa propre structure et apporte une solution au problème de réalisation d'un système interactif. Ces modèles peuvent être divisés en deux catégories principales : les modèles classiques et les modèles multiagent basés sur l'approche orientée objet.

III.5.2/ Les modèles classiques

Ces modèles s'inspirent beaucoup des niveaux de spécification des interfaces utilisateur. Deux d'entre eux sont décrits dans ce chapitre, il s'agit du modèle langage et du modèle E/S.

III.5.2.1/ Le modèle langage

Le modèle langage est composé de trois grandes couches distinctes situées, l'une derrière l'autre, entre les dispositifs matériels d'entrées/sorties de l'utilisateur et l'application abstraite (figure III.10).

Le modèle langage utilise les concepts connus qu'on a déjà introduit lors de la spécification des interfaces utilisateurs.

- **La présentation** : le composant présentation constitue le niveau lexical du modèle et se charge de définir la présentation visuelle des objets internes et abstraits de l'application. D'autre part, il sert à traduire les actions de l'utilisateur et des données externes en éléments syntaxiques compatibles avec le niveau voisin représenté par le composant contrôle du dialogue.

- **Dialogue** : le composant dialogue ou le niveau syntaxique définit la construction des phrases du langage à partir des éléments syntaxiques.

♦ **Interface avec l'application** : ce niveau sémantique définit comment l'application voit l'interface et vice-versa. Il assure ainsi l'indépendance interface - application.

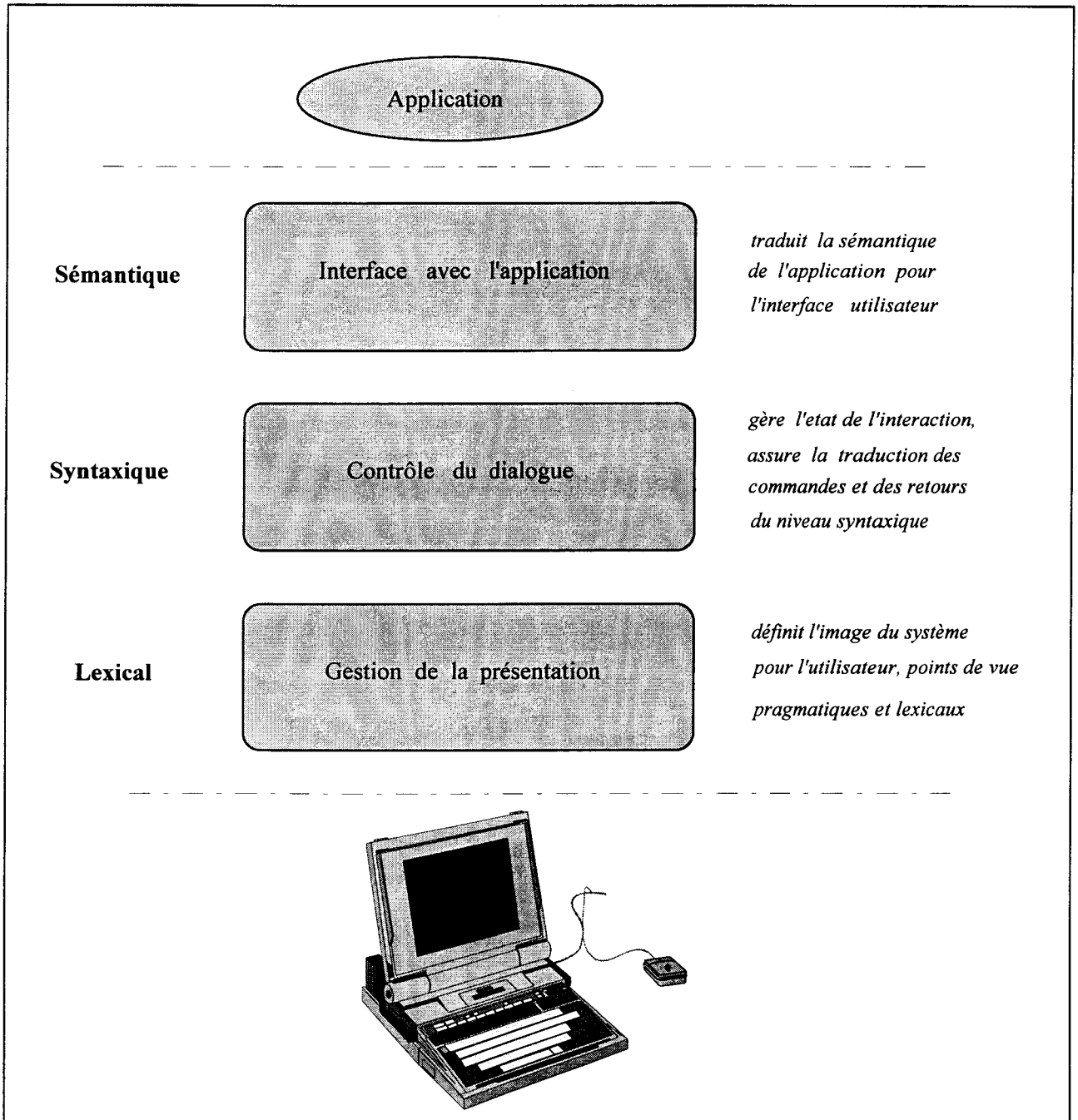


Figure III.10 : le modèle langage

III.5.2.2/ Evaluation du modèle langage

Le modèle langage s'inspire des théories et techniques connues des traitements des langages, ce qui lui confère une structure générique très claire avec la séparation des niveaux sémantique, syntaxique et lexical. Rajouter à cela sa généricité et sa généralité dues au fait que ce modèle ne donne aucune hypothèse ni aucun détail sur la nature des applications. A ces avantages, viennent se substituer les inconvénients majeurs suivants :

- imprécision des liens entre les différents niveaux et en particulier entre le niveau sémantique et l'application qui est difficile à mettre en oeuvre en pratique,
- découpage du langage d'interaction en deux sous-langages : un langage d'entrée et un langage de sortie. Cette dichotomie vient en contradiction avec des besoins courants en communication.

III.5.2.3/ Le modèle entrée / sortie

Ce modèle est structuré en plusieurs niveaux d'abstraction ou encore en plusieurs machines abstraites, chaque machine permettant de passer d'un niveau d'abstraction à un niveau supérieur (figure III.11).

Ces machines sont situées entre les dispositifs physiques d'entrée / sortie et l'application abstraite. La similitude avec le modèle langage s'arrête là, car contrairement au modèle précédent, les couches d'abstraction du modèle entrée / sortie ne correspondent pas directement aux niveaux de spécification des interfaces utilisateurs. Le modèle est représenté par la figure III.11.

Une deuxième différence majeure est que dans ce modèle, l'application communique avec tous les niveaux d'abstraction et n'est plus obligée de passer par les différentes couches pour accéder aux dispositifs physiques. Ceci permet, comme on peut le constater, des gains de temps d'exécution considérables.

De ce fait, l'architecture de ce modèle n'est pas seulement vue comme une structure en couches d'abstraction mais comme une structure modulaire.

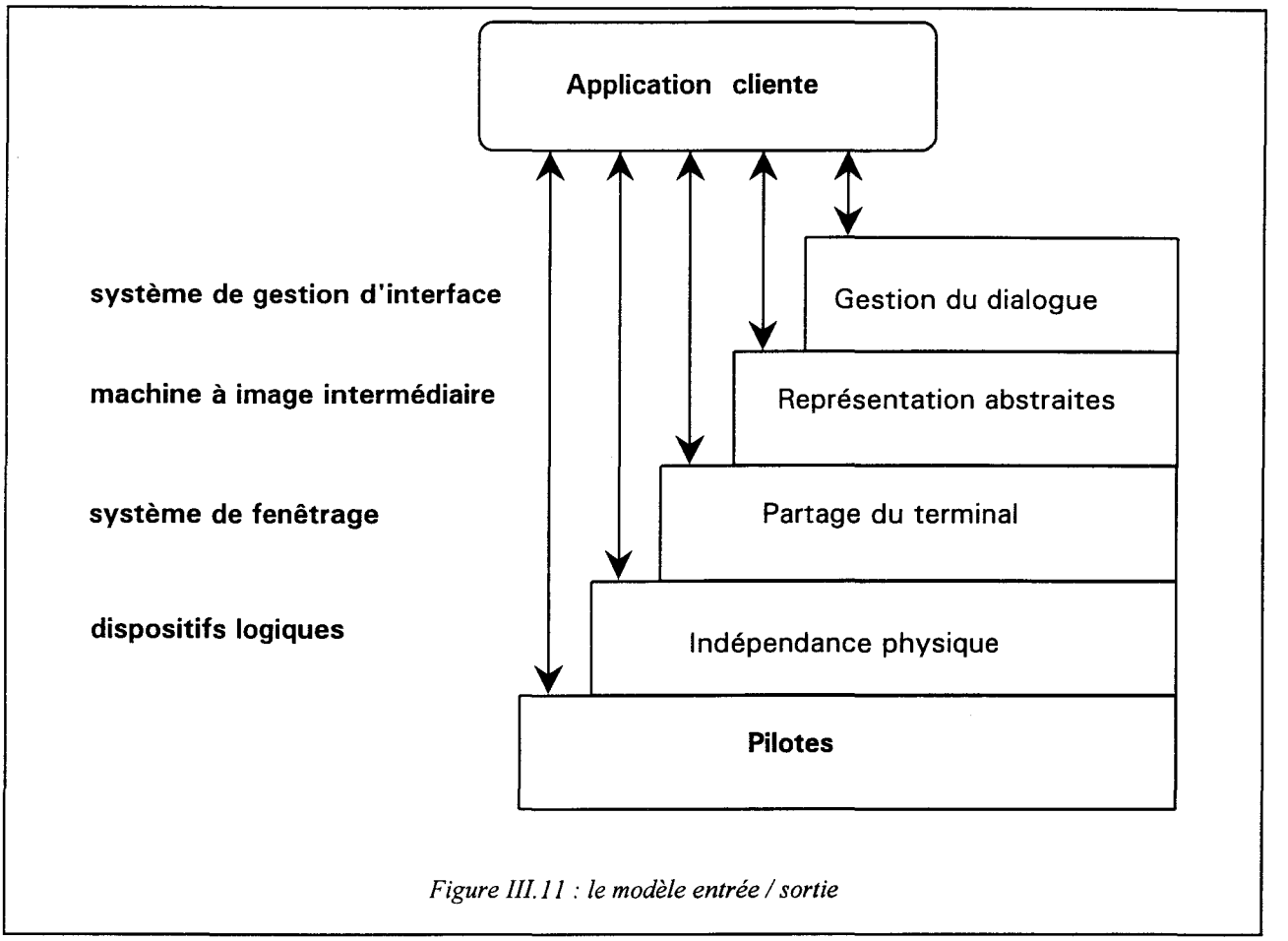


Figure III.11 : le modèle entrée / sortie

III.5.2.4/ Dispositifs logiques

Les dispositifs logiques ont pour rôle d'isoler la partie application abstraite des dispositifs physiques utilisés pour visualiser cette application. Ce concept s'appelle '*concept d'appareils virtuels*', Il assure la portabilité de l'application vis-à-vis des différents dispositifs d'entrée/sortie (figure III.12).

III.5.2.5/ Partage du terminal

Le système de fenêtrage assure la gestion des différentes fenêtres présentes sur l'écran d'un terminal. Ces fenêtres peuvent appartenir à des processus différents, c'est pourquoi cette couche d'abstraction est très utile dans un environnement multiprocessus (figure III.13).

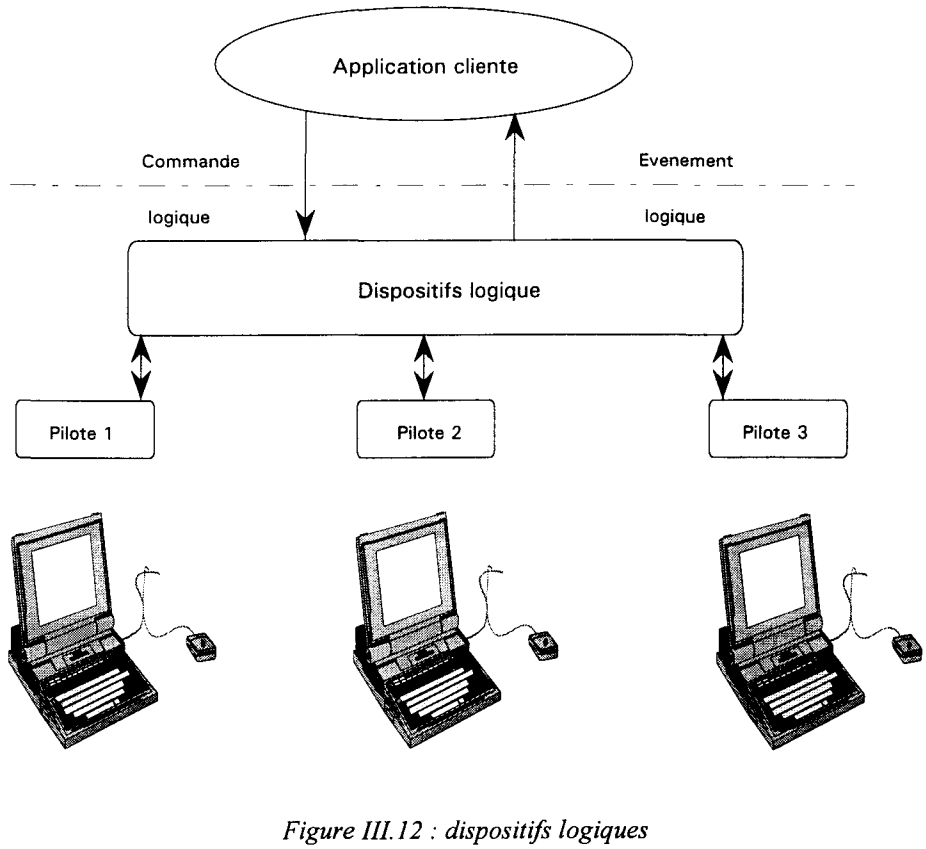


Figure III.12 : dispositifs logiques

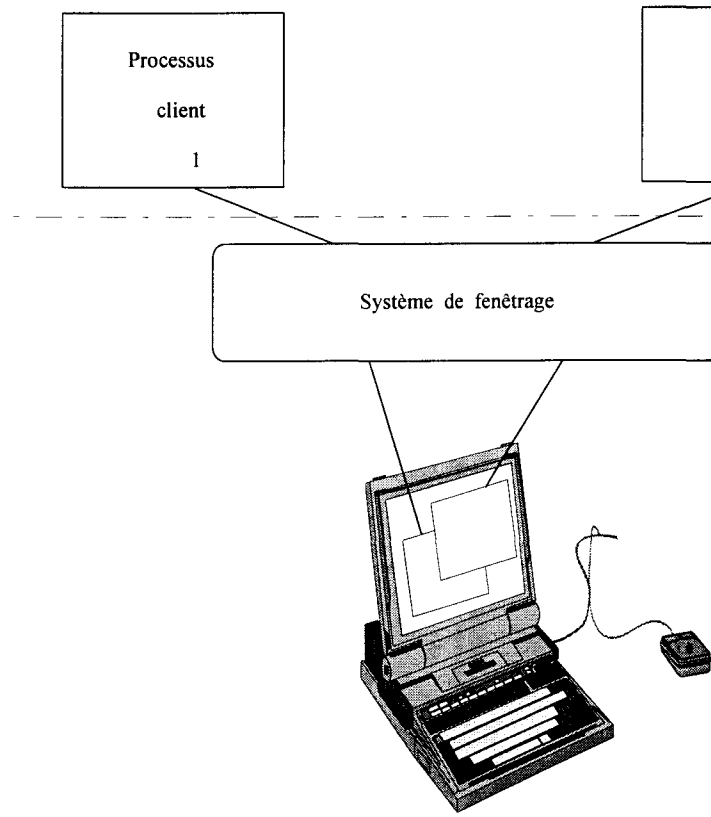


Figure III.13 : partage du terminal

III.5.2.6/ Machine à image intermédiaire

L'affichage d'information peut se voir comme une cascade de transformations, qui à partir de structures de données internes, produit une image réelle affichable en passant par l'image abstraite. L'acquisition des informations implique une suite de transformations inverses. La machine à image intermédiaire, pour passer du formalisme abstrait à l'image réelle, introduit des attributs et des règles de restitution indépendants du modèle de représentation des objets dans l'application.

Par exemple, l'attribut '*mise en évidence*' d'un élément sélectionné peut se traduire, selon le cas, par un affichage épais, un clignotement ou vidéo inverse.

III.5.2.7/ Gestion du dialogue

Cette machine constitue le quatrième niveau d'abstraction et assure l'abstraction du dialogue entre l'application et l'utilisateur. Le rôle du gestionnaire du dialogue est l'analyse syntaxique et lexicale des commandes d'une part, la décoration syntaxique et lexicale des retours d'autre part (figure III.14).

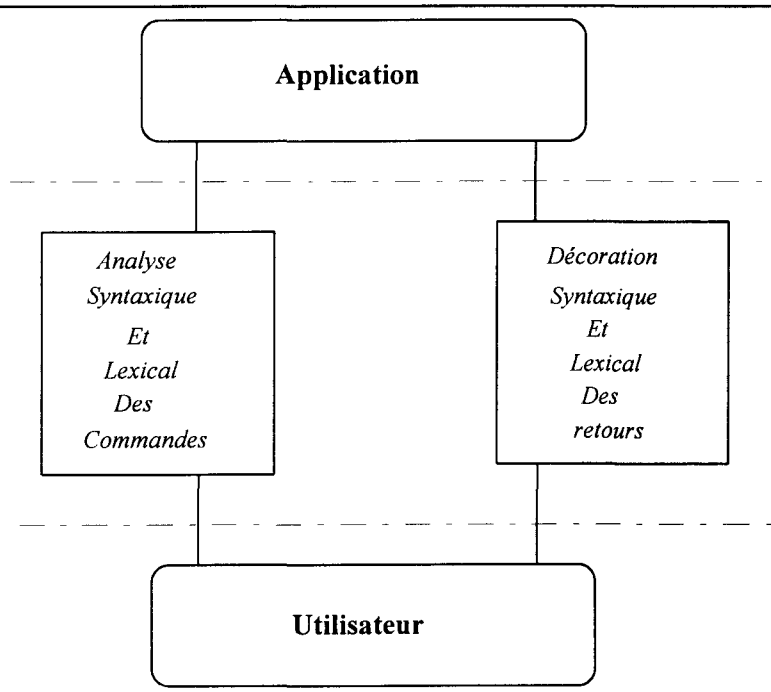


Figure III.14 : gestion du dialogue

III.5.2.8/ Evaluation du modèle d'E/S

Le modèle entrée / sortie est caractérisé par le fait que l'application peut avoir accès directement aux différents niveaux, permettant ainsi des gains de performances considérables.

D'autre part, en s'inspirant de la structure en couches abstraites, déjà utilisée et disponible sur les stations de travail, il assure aux systèmes interactifs une adaptation garantie (il s'agit bien sûr, des couches : système de fenêtrage et indépendance physique).

L'inconvénient du modèle d'entrée / sortie est qu'il ne propose pas d'avantage de techniques de réalisation de l'interaction, du côté de l'application (réalisation du gestionnaire du dialogue).

III.5.3/ Les modèles multiagent ou les modèles objet

On a constaté, durant ces dernières années, une évolution considérable des interfaces, accompagnée d'un changement radical de la philosophie d'interaction.

En effet, avec la nouvelle génération d'interfaces, on passe d'un ancien type de dialogue contrôlé par l'ordinateur, qui présentait à l'utilisateur des écrans successifs, à un type de dialogue contrôlé par les actions de l'utilisateur, traduites sous forme d'événements auxquels l'ordinateur doit répondre. L'interface devient événementielle. Une des difficultés majeures dans la construction des interfaces utilisateurs réside dans le contrôle de ces actions et la gestion du dialogue d'une manière générale. Avec les modèles précédents, on n'avait pas de stratégie claire et facile à mettre en oeuvre pour le contrôle du dialogue. Avec le modèle multiagent, l'utilisation du paradigme objet nous amène à répartir le contrôle sur une multitude d'objets de présentation appelés aussi objets interactifs.

Ces objets interactifs (fenêtres, icônes, processus, boutons, menus, formulaires) manipulés dans l'interface, apparaissent comme des médiateurs entre le monde abstrait du système et le monde concret de l'utilisateur avec un double comportement : un

comportement interne vis à vis de l'application et un comportement de surface vis à vis de l'utilisateur, image du comportement interne dans la vision naïve de ce dernier.

Les événements, actions de l'utilisateur sur des objets, se traduisent en messages adressés à ces objets. Ces messages sélectionnent les méthodes responsables des comportements internes et externes des objets.

Son principe de répartition du dialogue lui confère la possibilité de gestion de la composition et l'interaction de la multitude d'objets mis en oeuvre dans le dialogue.

L'utilisation du paradigme objet autorise une grande ouverture vers la dimension sociale de l'interface Homme - Ordinateur. Sa forte modularité, au niveau d'abstraction, permet une conception itérative des interfaces utilisateurs.

III.5.4/ Quelques exemples des modèles orientés objet

Plusieurs modèles ont été proposé, et s'appuient tous sur le paradigme objet (agent) et ont tous, comme point commun, la distinction entre l'abstraction appartenant à l'application et la présentation de l'interface. On retrouve le modèle MVC (Modèle, Vue, Contrôleur) de Smalltalk, le modèle PAC (Présentation, Abstraction, Contrôleur) de Joëlle COUTAZ. Après une petite description de chacun des modèles, nous présentons le modèle P.Os.T (Présentation, Objet de simulation , Traducteur) qu'on a développé et combiné avec d'autres modèles pour la réalisation de notre interface graphique de simulation.

III.5.4.1/ Le modèle MVC [GOL 83]

C'est un modèle proposé par Smalltalk, en tant que premier langage totalement orienté objet. Il repose sur trois composantes:

- le modèle, qui représente les données manipulées par l'application,
- la vue des données, représentée par un ensemble de fenêtres sur l'écran, constitue l'interface visuelle entre l'utilisateur et les données,
- le contrôle des données, constitue l'interface d'entrée permettant à l'utilisateur d'agir sur les données, à l'aide du clavier et de la souris.

Alors que le modèle des données est spécifique à chaque application, la manière de présenter ces données à l'écran ou de les manipuler est identique pour la plupart des applications. Editer un texte, sélectionner un item dans une liste ou designer une icône avec la souris, nécessite la mise en oeuvre des mêmes outils, quelle que soit l'application. Le système Smalltalk propose une gamme pré-définie de classes, permettant de créer des vues et des contrôleurs, adaptés à différentes manières de présenter les données.

III.5.4.2/ Exemple d'utilisation du modèle MVC

L'exemple que nous avons choisi, pour expliquer le principe d'utilisation du modèle MVC, est un éditeur graphique d'images. L'interface de l'éditeur est présentée en figure III.15, sous forme de fenêtre composée de plusieurs types de vues.

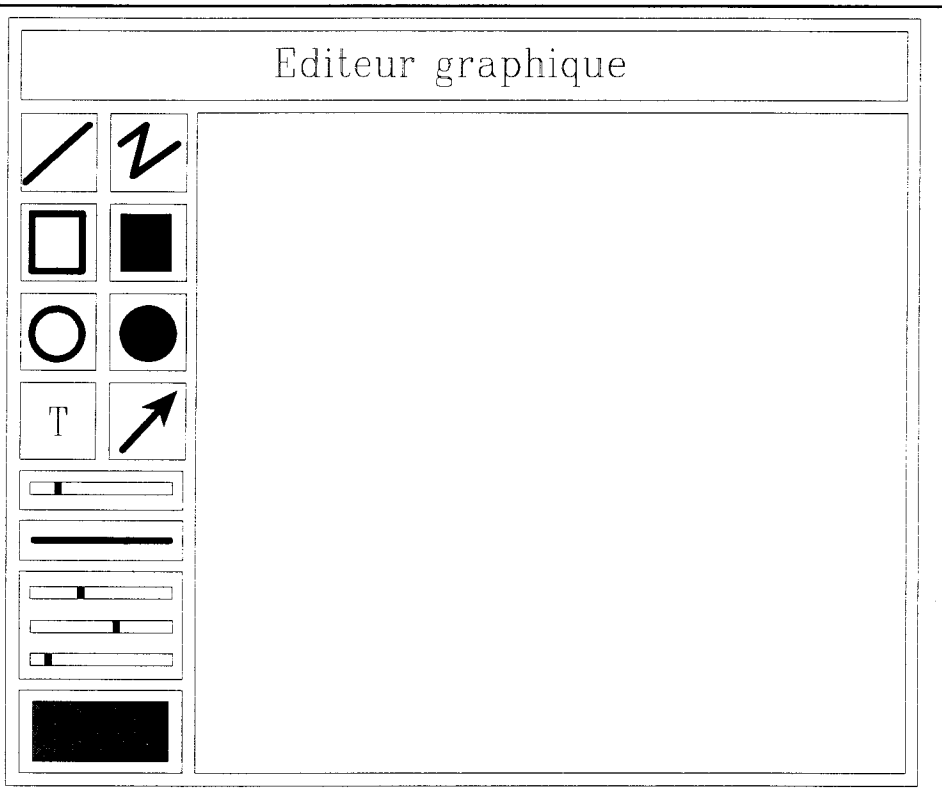


Figure III.15 : exemple d'application du modèle MVC

Pour chaque élément de l'interface, il existe un type de vue et un type de contrôleur assortis (figure III.16). Par exemple, la vue associée à un bouton et le

contrôleur correspondant sont respectivement instances des classes "*LabeledBooleanView*" et "*WidgetControler*". Chaque vue élémentaire (sous vue) assure le contrôle des actions de l'utilisateur grâce à son contrôleur. Lorsque l'utilisateur "appuie" sur un bouton, c'est-à-dire le pointe avec la souris, le contrôleur capte l'événement et en avertit la vue. La matérialisation de l'événement à l'écran est à la charge de cette dernière : en l'occurrence, le fond du bouton est noirci pour signifier à l'utilisateur que son action à été prise en compte.

En plus de la réaction visuelle, l'enfoncement d'un bouton peut provoquer d'autres actions. Le déclenchement des actions et le traitement des communications entres les vues sont dévolus à un objet particulier, "le modèle" de l'application, qui joue en quelques sorte, le rôle de chef d'orchestre pour les différentes vues.

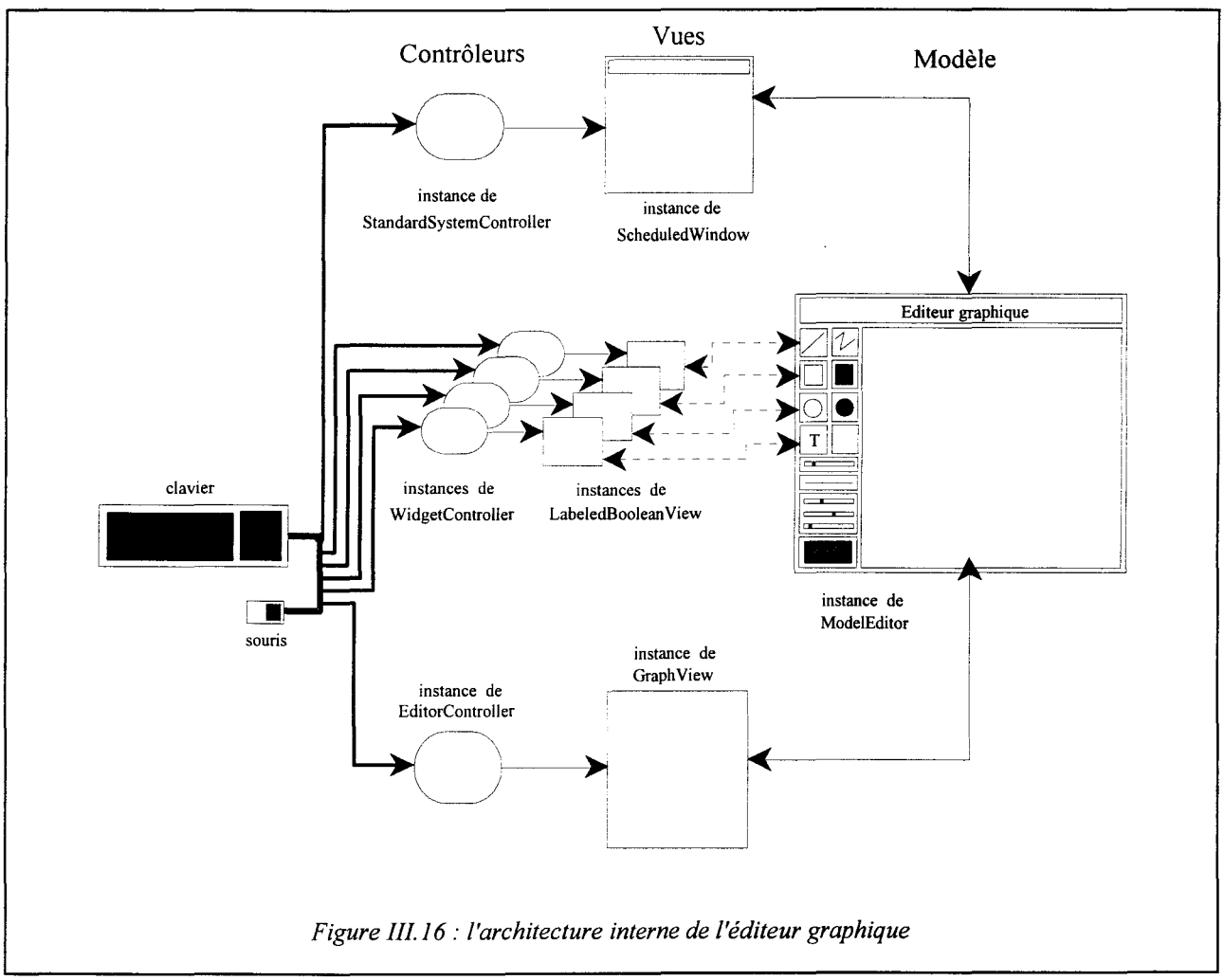


Figure III.16 : l'architecture interne de l'éditeur graphique

Un ***modèle*** est défini par une classe spécifique du type de composant dont il fait partie. Un modèle connaît les fenêtres qui dépendent de lui. Il peut les avertir des éventuels changements les concernant grâce à un ensemble de méthodes.

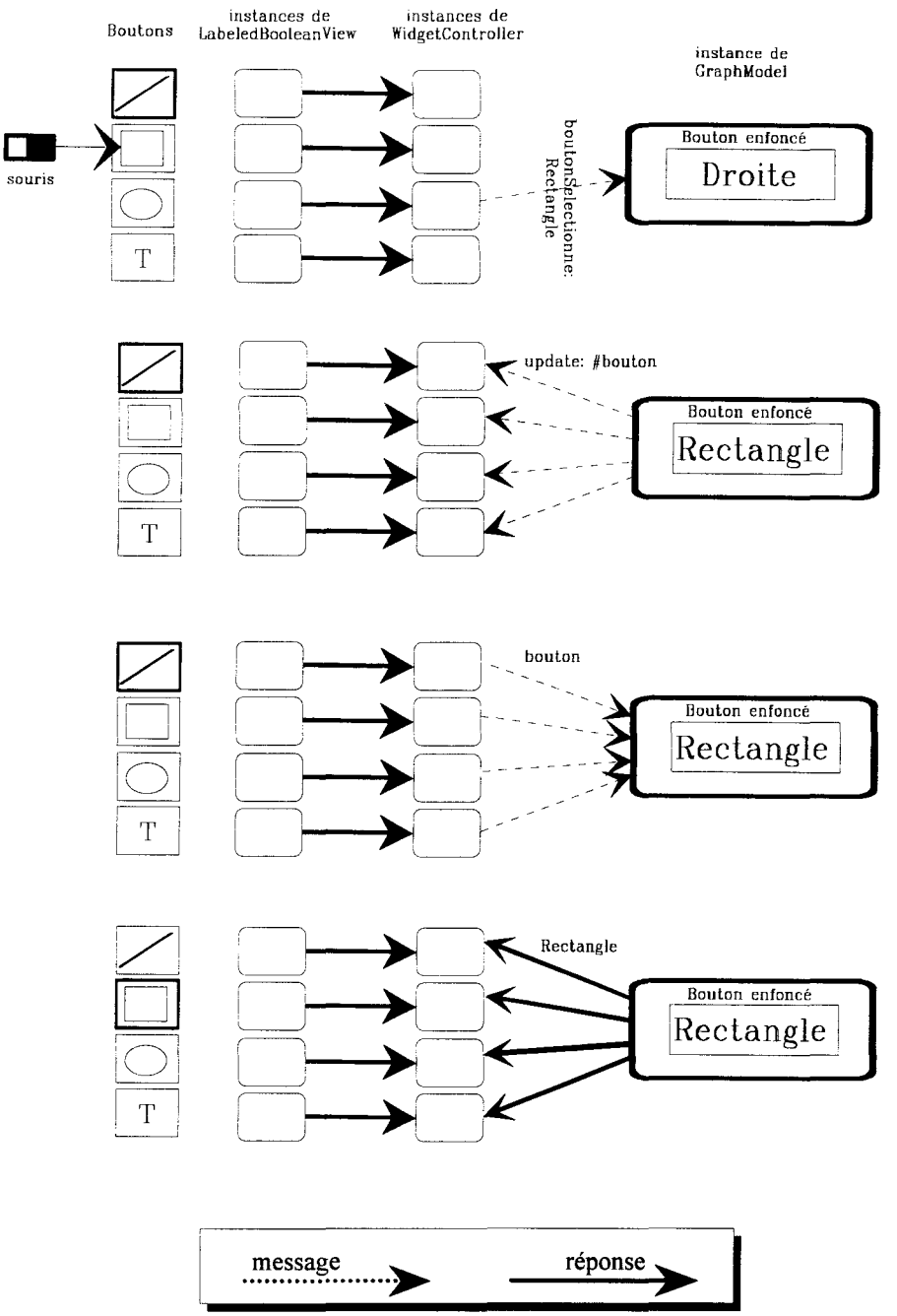


Figure III.17 : les échanges d'informations entre le modèle et les boutons

Par exemple si l'utilisateur appuie sur un bouton alors qu'un autre bouton est déjà "enfoncé", le modèle envoie un message à ce dernier pour qu'il se "relève". Réciproquement, chaque fenêtre connaît le modèle dont elle dépend grâce à une

variable d'instance. Elle peut ainsi l'interroger en lui envoyant des messages, en particulier pour connaître l'aspect qu'elle doit prendre.

Dans notre cas, le modèle associé à l'éditeur graphique est instance de la classe de *"ModelEditor"*. Il dispose de toutes les informations nécessaires pour déterminer l'aspect et le contenu des fenêtres: la fonction du bouton enfoncé, la couleur choisie, l'épaisseur du trait sélectionnée, ...etc.

La figure III.17 récapitule les échanges entre les vues et le modèle, lorsque l'utilisateur enfonce un bouton, par exemple.

III.5.4.3/ Evaluation du système MVC

Le principe MVC reste, à l'heure actuelle, l'une des meilleures solutions commercialisées pour la construction rapide d'interfaces, en ayant un minimum de pratique et d'expérience de la bibliothèque Smalltalk.

Il faut cependant remarquer que la programmation des interfaces utilisant le principe MVC, reste en grande partie procédurale. Ceci entraîne nécessairement des échanges par messages qui risquent d'allonger le temps de réaction du système.

Par ailleurs, dans une structure MVC on distingue l'abstraction, représentée par le modèle "M", de la présentation représentée par la vue "V". Le passage de l'abstraction à l'affichage sur écran est une opération assurée par un certain nombre de méthodes, réparties entre le modèle et la vue. L'inconvénient est que le modèle MVC n'offre pas de stratégie pour la réalisation de nouvelles vues et de méthodes de transformations de l'image abstraite à l'image réelle.

III.5.4.4/ Le modèle PAC (Présentation, Abstraction, Contrôleur)

Le modèle PAC [COU 90] structure l'architecture d'un système interactif en une multitude d'agents (ou objets interactifs), organisés en trois classes de compétence (figure III.18).

- la présentation définit l'image de l'agent, c'est-à-dire son comportement perceptible à l'utilisateur,

- l'abstraction définit les fonctions et les attributs internes de l'agent, c'est-à-dire son comportement vis-à-vis d'autres agents,
- le contrôle maintient la cohérence entre les deux perspectives.

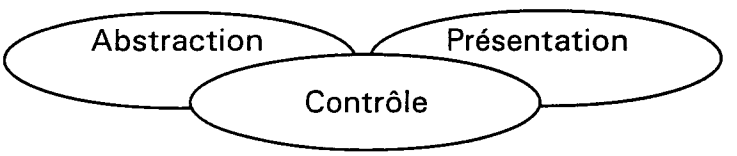


Figure III.18 : structure du modèle PAC

III.5.4.5/ Vue d'un système interactif par le modèle PAC

Un système interactif est vu comme une hiérarchie d'objets PAC, allant du niveau le plus haut au niveau le plus bas (figure III.19).

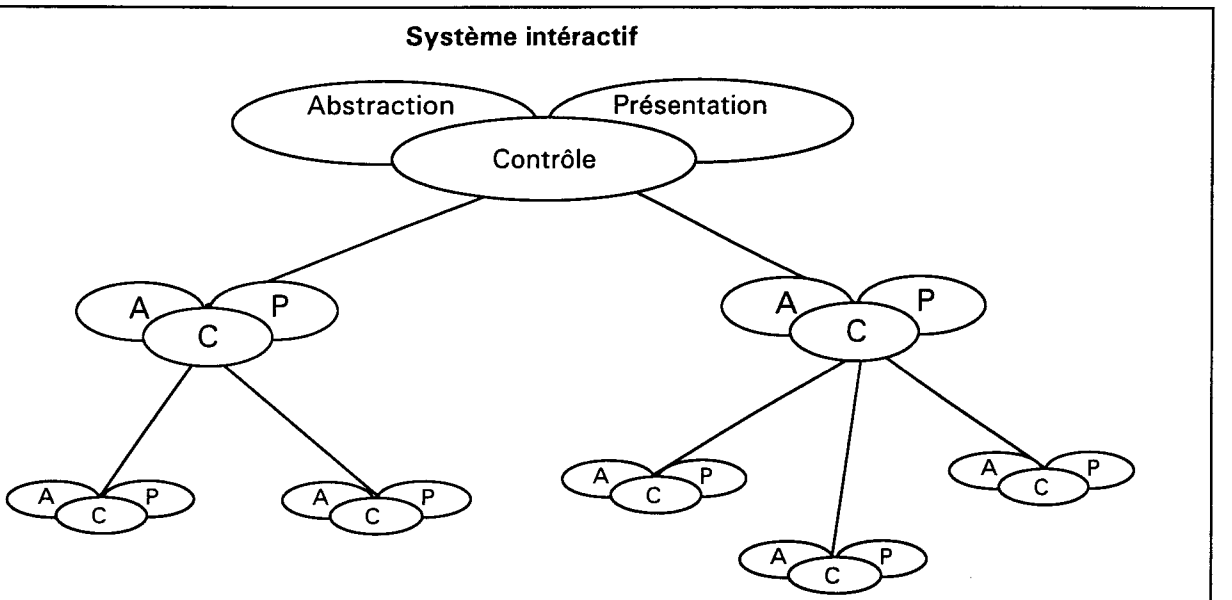


Figure III.19 : vue d'un système interactif

Du point de vue du contrôle du niveau le plus haut, l'application est un "producteur / consommateur" de valeurs qui modélisent les concepts du domaine. Une des fonctions du contrôle du niveau le plus haut est de mettre en correspondance les

concepts de l'application avec les abstractions des objets interactifs du niveau le plus bas. Un objet interactif peut être élémentaire ou composé :

- un objet interactif élémentaire est une unité de compétence et d'exécution répartie en trois classes de fonctions: Présentation, Abstraction et Contrôle,
- un objet interactif composé définit une nouvelle unité de compétence, organisée selon les mêmes critères qu'un objet élémentaire, mais qui en plus, hérite du comportement des objets constituant et les soumet à ses propres règles.

Exemple d'application du modèle PAC (THERMO)

Le système Thermo a été réalisé en "C" dans un environnement mono tâche [COU 90] et à pour objet de présenter les relations entre les notions de chaleur, de température et de volume d'eau : un thermomètre indique la température actuelle, un réchaud fait office de source de chaleur, un récipient sert à recevoir un certain volume de liquide et un distributeur permet d'obtenir de l'eau.

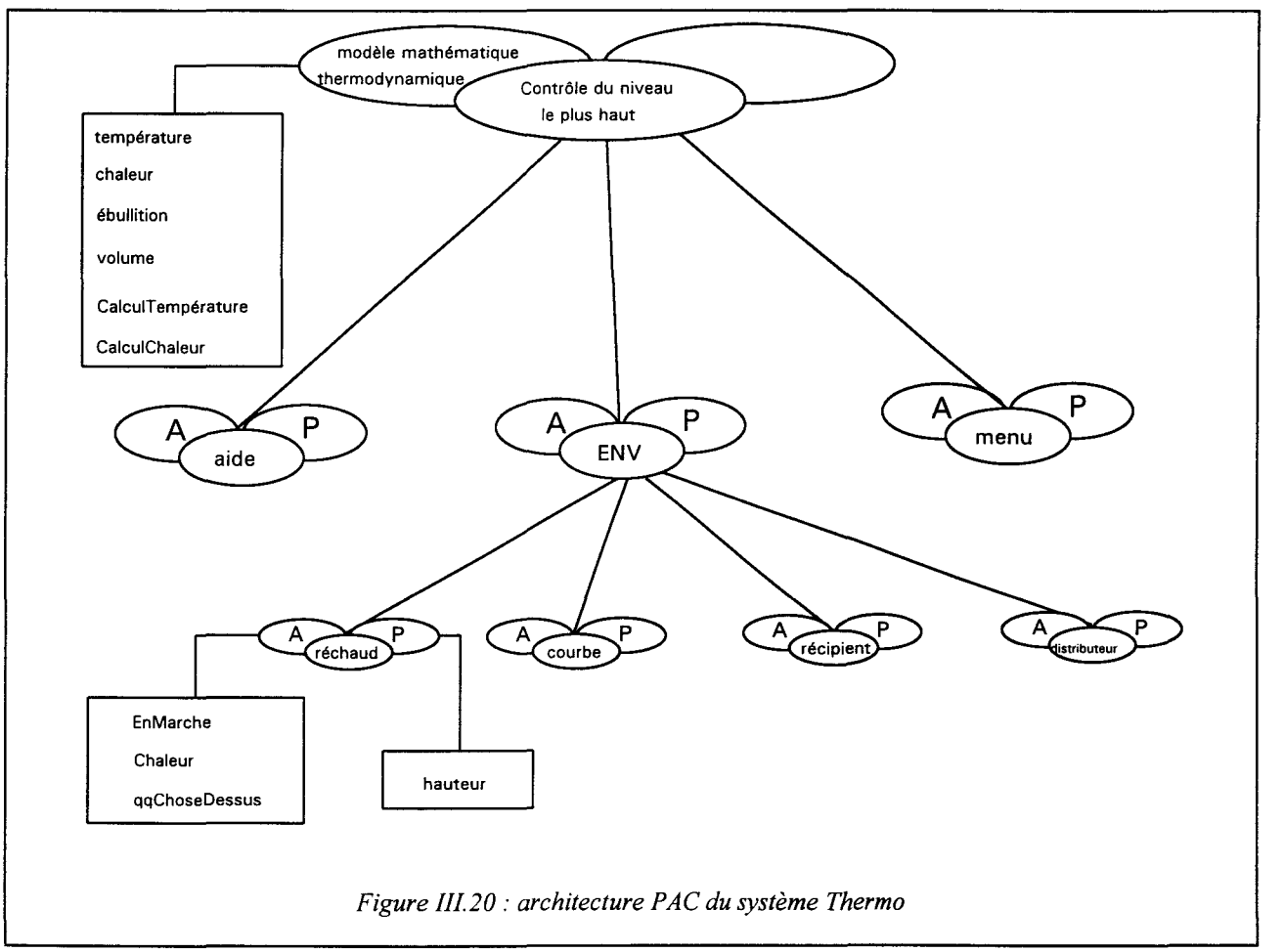


Figure III.20 : architecture PAC du système Thermo

La figure III.20 décrit l'architecture PAC du système Thermo. L'abstraction du niveau le plus haut, modélise les lois de la thermodynamique. Elle est capable de déterminer, à tout instant, la chaleur produite par une résistance, de calculer la température d'un volume d'eau et de déterminer si cette eau est en état d'ébullition ou non. La présentation du niveau le plus haut, n'a pas d'attributs particuliers : elle est considérée comme inexistante.

Les objets composants se répartissent en deux catégories: ceux qui reproduisent des objets du monde réel, en rapport étroit avec les fonctions du système telles que le Réchaud, le Récipient et le Distributeur, et ceux d'utilité générale tels que l'aide, les boutons et les menus.

III.5.4.6/ Echange entre l'application et l'interface

Dans le cas du système Thermo, l'application parle de température mais doit tout ignorer des techniques de présentation. De même, le thermomètre du récipient peut servir à présenter d'autres valeurs que celle du système Thermo. L'application et le récipient ne communiquent jamais directement mais passent par une indirection gérée dans le niveau le plus haut.

Dans le cas du système Thermo, la technique des tables de correspondance a été choisie pour réaliser les indirections dont les liaisons les plus significatives sont représentées par la figure III.21.

Dans le sens application ==> interface, l'entier "température" qui modélise la température est lié à l'abstraction "Temp" d'un objet de classe "Récipient"; l'entier "chaleur" est lié à l'abstraction "Chaleur" d'un objet de classe "Réchaud".

Dans le sens interface ==> application, l'abstraction "EnMarche" d'un objet de classe "Réchaud" est liée à la fonction "CalculChaleur" de l'application et l'abstraction "qqChoseDessus" à la fonction "CalculTempérature".

Ainsi, lorsque l'application modifie un concept, le contrôle exprime le changement dans les termes des abstractions des objets qui lui sont associées. Les contrôles de ces objets prennent le relais jusqu'à ce que l'effet se traduise dans les présentations des objets élémentaires. Dans le sens inverse, les actions de l'utilisateur sont interprétées par les présentations du plus bas niveau, puis colportées en remontant la hiérarchie jusqu'à ce que des abstractions d'objets interactifs soient en liaison directe avec les concepts de l'application.

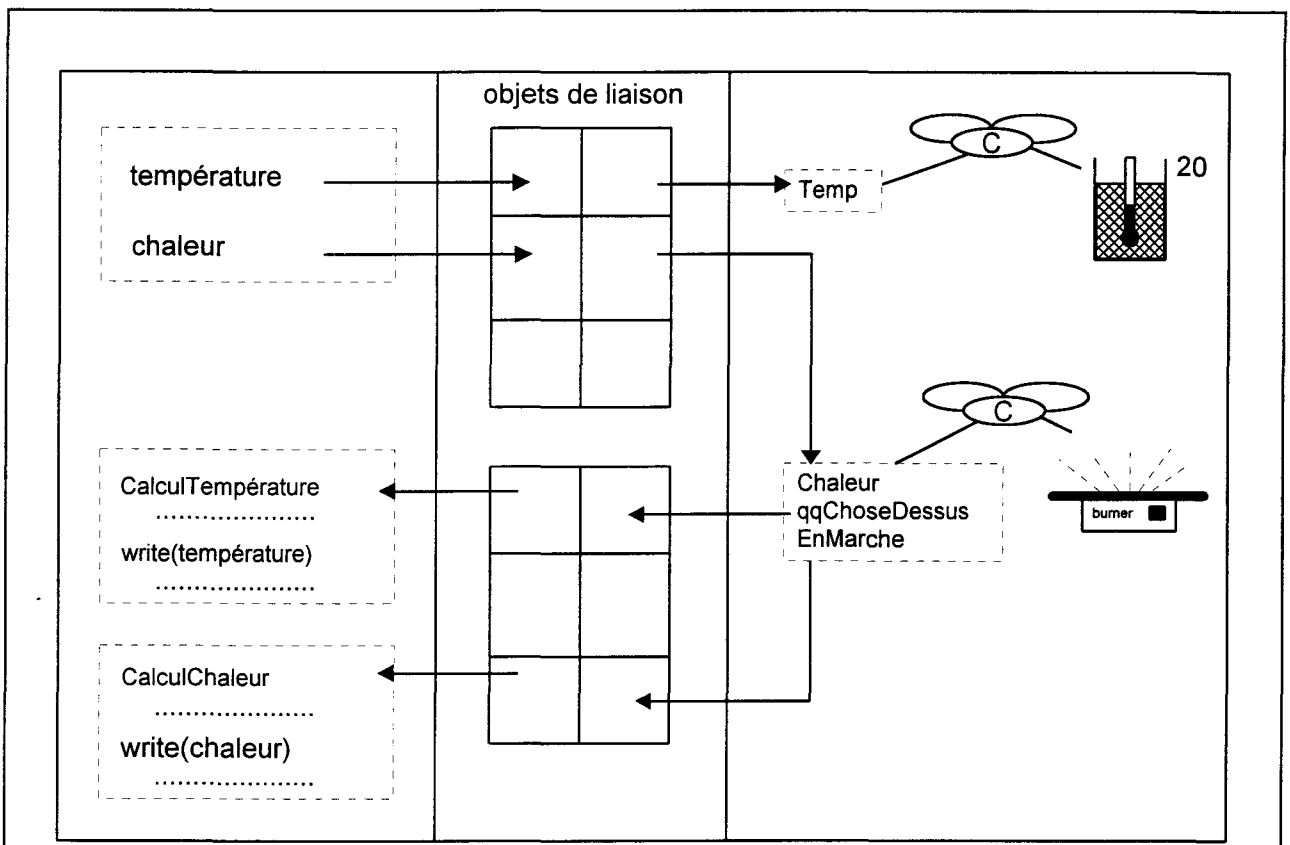


Figure III.21 : échanges entre l'application et l'interface

III.5.4.7/ Evaluation du modèle PAC

Le modèle PAC a l'avantage de s'appuyer sur les principes directeurs du modèle multiagent, ce qui lui confère une organisation modulaire et une ouverture vers les traitements physiquement distribués. Evaluer le modèle PAC nous amène à regrouper les apports et les limites du modèle dans les paragraphes suivants :

Apports du modèle PAC

♦ Il repose sur une structure modulaire et générique applicable à tous les niveaux d'abstraction d'un système interactif.

♦ Il distingue les services abstraits des techniques d'interaction et il introduit la notion de contrôle qui permet à deux perspectives de communiquer par indirection avec l'avantage de modifier l'une sans mettre en cause le fonctionnement de l'autre.

Les limites du modèle PAC

♦ Comme il a été précédemment décrit dans l'exemple Thermo, l'architecture PAC d'un système interactif est représentée par une structure hiérarchique. Au sommet de la hiérarchie se trouve l'abstraction et le contrôle du niveau le plus haut. L'abstraction constitue le modèle mathématique de l'application concernée. Celle du système Thermo, par exemple, modélise les lois de la thermodynamique.

Au bas de la hiérarchie sont répartis les modèles PAC élémentaires (agents) dont les abstractions n'ont de rôle que celui d'un intermédiaire, entre les présentations élémentaires et l'abstraction du niveau le plus haut.

En fait, le principe de répartition sur une multitude d'agents n'est appliquée que sur la partie présentation graphique. Ceci limite, dans la phase de modélisation, l'application du principe métaphorique du monde réel qui considère les objets interactifs, qu'ils soient élémentaires ou composés, comme des objets complets, ayant un comportement interne, pour décrire les lois physiques de l'objet représenté, et un comportement externe pour la communication avec l'environnement.

♦ Une retombée négative apparaît, en conséquence à la limitation précédente, celle du risque d'allonger le temps de réaction du système.

En effet, si un objet subit une modification interne ou externe de l'utilisateur, cette modification est interprétée par la présentation du niveau de l'objet. Elle est reflétée dans l'abstraction du même niveau, puis colportée en remontant la hiérarchie jusqu'au niveau le plus haut, ou se décidera les réactions nécessaires et les conséquences à cette modification pour une éventuelle mise à jour.

- Le modèle PAC a l'avantage de s'appuyer sur le principe de distinction entre abstraction et présentation. Il introduit la notion de contrôle ayant la charge de faire cohabiter deux environnements totalement indépendants, en appliquant le principe d'indirection. Dans le sens application ==> interface, toute modification d'abstraction est signalée au contrôle, celui ci en traduit l'effet sur l'ensemble des présentations liées à cette abstraction. Le modèle PAC ne précise pas comment se fait cette traduction ni comment on peut créer et gérer l'animation.

III.6/ CONCLUSION

La conception d'une interface graphique destinée à la simulation interactive est un projet d'une grande complexité. Ceci est dû :

- à la complexité du dialogue homme-machine,
- à la diversité des utilisateurs,
- à la grande diversité des applications potentielles.

Le but de l'outil de simulation, tout en étant totalement ouvert à cette diversité de types de dialogues, d'utilisateurs et d'applications, est qu'il puisse assurer une complète séparation entre le monde virtuel de l'application et le monde physique des entrées / sorties.

La nouvelle génération des interfaces graphiques a induit de nouvelles méthodes de conception nécessitant des modèles d'architecture informatique. Dans ce chapitre, nous avons classé les modèles en deux catégories :

- les modèles classiques tels que le modèle langage et le modèle E/S. Ces modèles conduisent à de très gros modules, ce qui est peu compatible avec la multiplicité des possibilités dans un dialogue piloté par l'utilisateur;
- les modèles multiagent (à objet), ou le dialogue se répartit sur une multitude d'objets interactifs et s'inscrit directement dans le cadre de l'approche objet - action du dialogue homme-machine. Dans cette catégorie, nous nous sommes particulièrement intéressés aux modèles MVC de Smalltalk, PAC de Jöelle Coutaz.

Le chapitre suivant sera dédié à la présentation d'une architecture d'interface de simulation interactive, basée sur un modèle baptisé P.Os.T. (Présentation, Objet de Simulation, traducteur). Ce modèle est décomposé en trois parties : l'abstraction, décrivant l'aspect modélisation du système simulé qu'on a vu au chapitre II, la présentation graphique, chargée de l'aspect visuel de l'abstraction et en fin le traducteur, permettant d'assurer la cohérence entre les deux parties précédentes.

Chapitre IV

Le modèle P.Os.T

Le modèle P.Os.T

IV.1/ INTRODUCTION

Une des conclusions importantes induites du chapitre précédent, est la nécessité de construire l'interface graphique homme-machine sur la base d'un modèle architectural ayant une structure adaptée aux différents besoins d'interaction lors du dialogue homme-ordinateur. Nous avons remarqué que les modèles multiagent apportent beaucoup d'avantages par rapport aux modèles classiques. L'utilisation du paradigme objet les conduit à un dialogue homme-machine piloté par l'utilisateur et non par le programme.

Par ailleurs les concepteurs d'interfaces utilisateurs ne disposent pas, maintenant, d'un modèle standard capable de s'adapter à toutes les applications avec les mêmes performances souhaitées. Ils ont par contre la possibilité de choisir un modèle déjà existant se rapprochant de leurs besoins ou alors d'en concevoir un nouveau.

Au début de nos travaux, nous étions confrontés à ce problème de choix et nous avons essayé d'apporter notre contribution dans ce sens, en proposant une architecture informatique des interfaces graphiques destinées à la simulation interactive.

Cette architecture est basée sur la combinaison du modèle MVC, que nous avons décrit dans le chapitre précédent, et du modèle P.Os.T (Présentation, Objet de Simulation, Traducteur), l'objet de ce chapitre. La raison d'être du modèle P.Os.T réside dans sa structure très générique, conforme aux modèles multiagent dont les avantages ont été cités précédemment, et dans les outils de sa conception. Ces outils

permettent la création interactive de représentations graphiques très parlantes des modèles de simulation, et la génération automatique d'animations de synoptiques.

IV.2/ L'ARCHITECTURE DE L'INTERFACE

L'interface de simulation, présentée sur la figure IV.1, est décomposée en deux catégories d'éléments :

- les éléments représentant les fonctions de la simulation, par exemple, les boutons de commande (marche/arrêt, charge/sauver, etc) ou les menus déroulants,
- Les éléments représentant les objets de simulation.

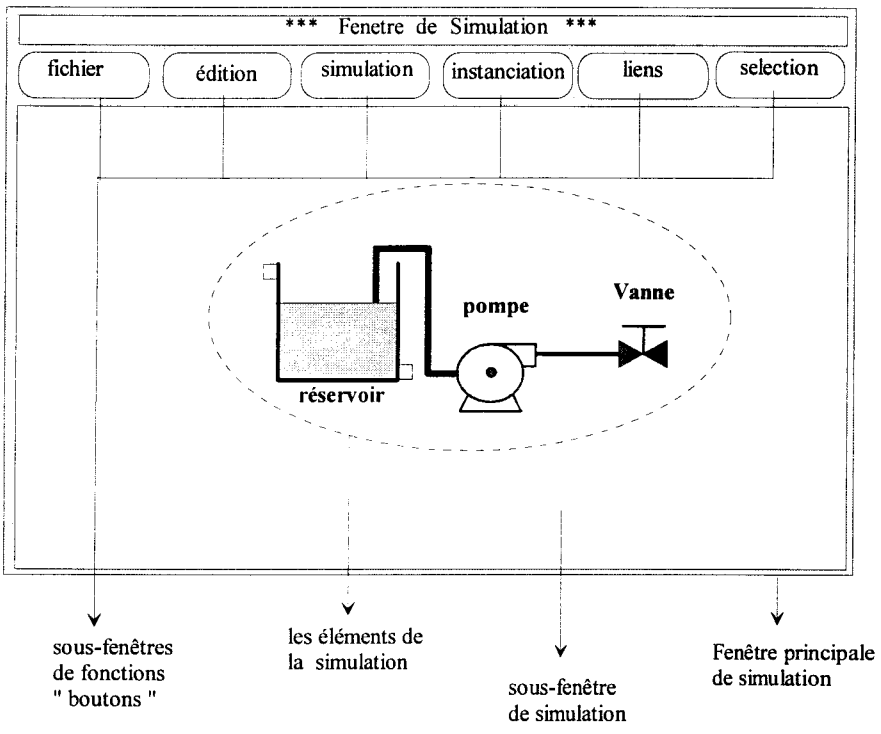


Figure IV.1 : structure de l'interface

Lors de la construction de l'interface utilisateur, nous avons adopté la configuration suivante :

- le modèle MVC de Smalltalk comme base de conception des éléments de la première catégorie,

- le modèle P.OS.T, que nous avons développé pour représenter ceux de la deuxième catégorie.

L'architecture informatique de l'interface est présentée sur la figure IV.2. Dans cette architecture, nous pouvons distinguer trois niveaux hiérarchiques.

Au sommet de la hiérarchie, on trouve le modèle MVC de la fenêtre principale de l'interface. La vue (*ScheduledWindow*) et le contrôleur (*StandardSystemControler*) associés à ce modèle prennent en charge toutes les opérations standards concernant la fenêtre : ouverture et fermeture de la fenêtre, gestion des rafraîchissements, réception et gestion des événements utilisateur, etc.

Le deuxième niveau contient d'une part un ensemble de modèles MVC correspondant aux différents boutons de fonction décrits précédemment et d'autre part un autre modèle MVC représentant la sous fenêtre de la plate-forme de simulation. Cette fenêtre est destinée à contenir les objets de simulation. Son modèle constitue le

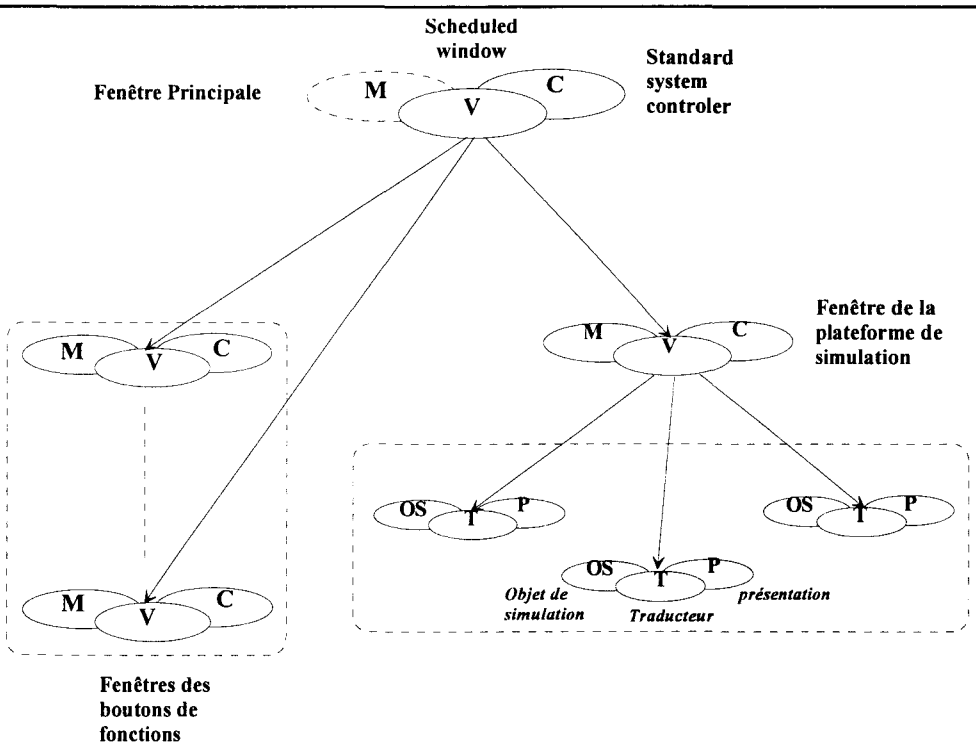


Figure IV.2 : architecture informatique de l'interface graphique de simulation

modèle de l'application, il connaît à tout moment le bouton de la fonction sélectionnée. Lors d'un changement d'un bouton de fonction, il est chargé d'informer le contrôleur de

cette même fenêtre. Ce contrôleur appelé " contrôleur graphique " peut en conséquence changer l'aspect de contrôle en fonction du bouton sélectionné. Il constitue le deuxième interlocuteur dans la communication entre l'utilisateur et l'environnement de la simulation (le premier étant le contrôleur de la fenêtre principale). Cet environnement est représenté par tous les modèles P.Os.T qui apparaissent au dernier niveau de la structure de la fenêtre.

IV.3/ LE CONTROLEUR GRAPHIQUE

Le contrôleur graphique est chargé de contrôler la communication entre l'utilisateur et le modèle de simulation. Il assure ainsi les deux tâches élémentaires d'une interaction : l'affichage du synoptique et des informations nécessaires au dialogue d'une part et la prise en compte des actions utilisateur d'autre part.

Dans la première tâche, le contrôleur se sert d'une liste qui contient toutes les présentations des objets de simulation à afficher, à un instant donné. L'affichage du synoptique se réduit au simple affichage des éléments de la liste.

Dans la deuxième tâche, le contrôleur utilise deux paramètres supplémentaires : l'***élément sélectionné*** et le ***bouton de fonction sélectionné***.

La souris, utilisée comme dispositif d'entrée principal, contient trois boutons :

bouton de gauche : le bouton d'action,

bouton du milieu : réservé au menu des opérations du contrôleur graphique,

bouton de droite : réservé au menu système de la fenêtre principale. Les actions de l'utilisateur sur ce bouton sont interprétées par le contrôleur principal.

Les actions de l'utilisateur sur le bouton gauche ou celui du milieu sont traduites sous forme d'événements envoyés au contrôleur graphique. La réaction de ce dernier s'adapte en fonction du contexte courant : bouton de fonction sélectionné, bouton de souris sélectionné (gauche ou milieu) et position de la souris. Un organigramme d'interprétation des actions de l'utilisateur est montré en figures IV.3 et IV.4.

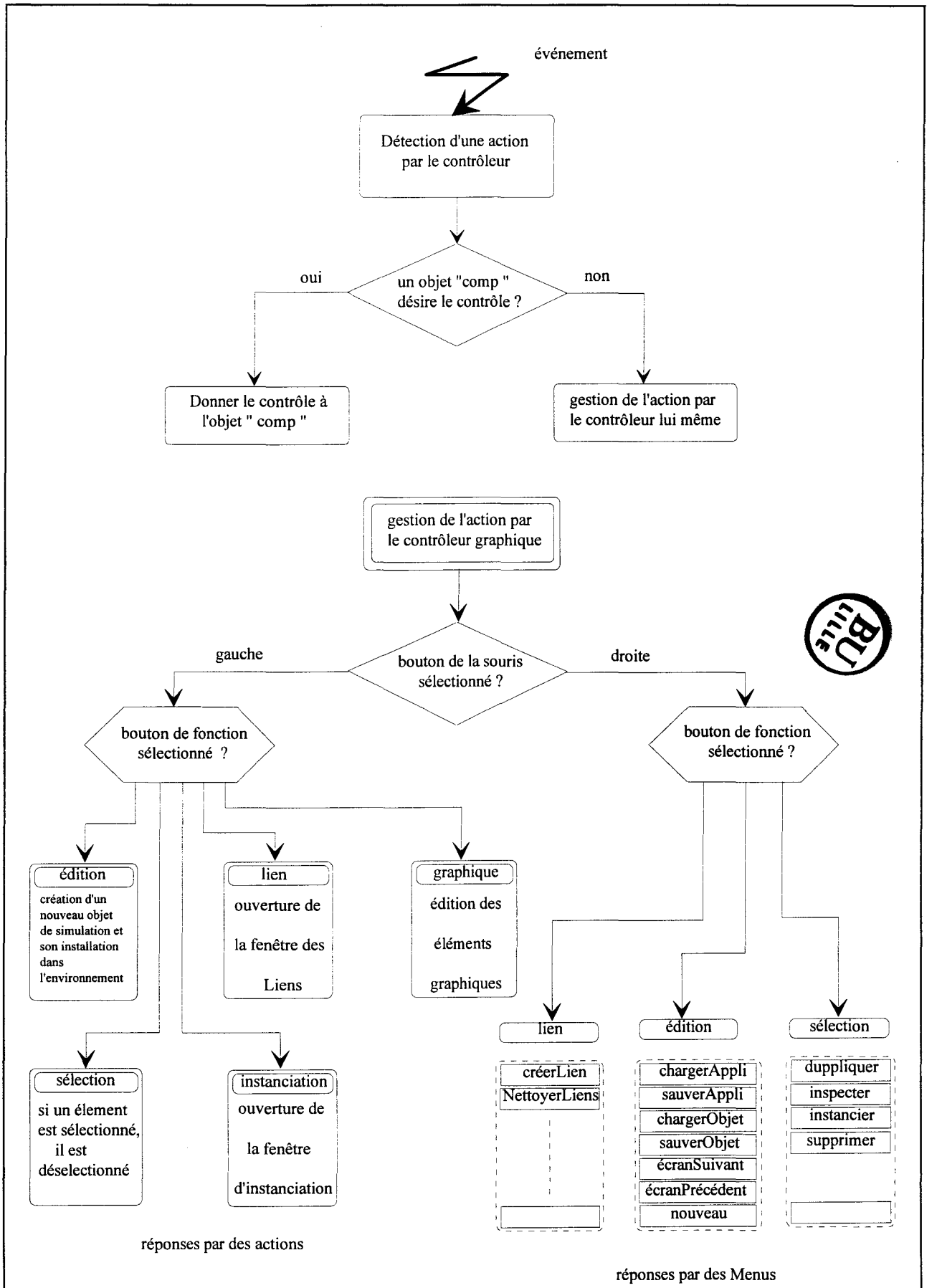


Figure IV.3 : organigramme de gestion des actions utilisateur (1)

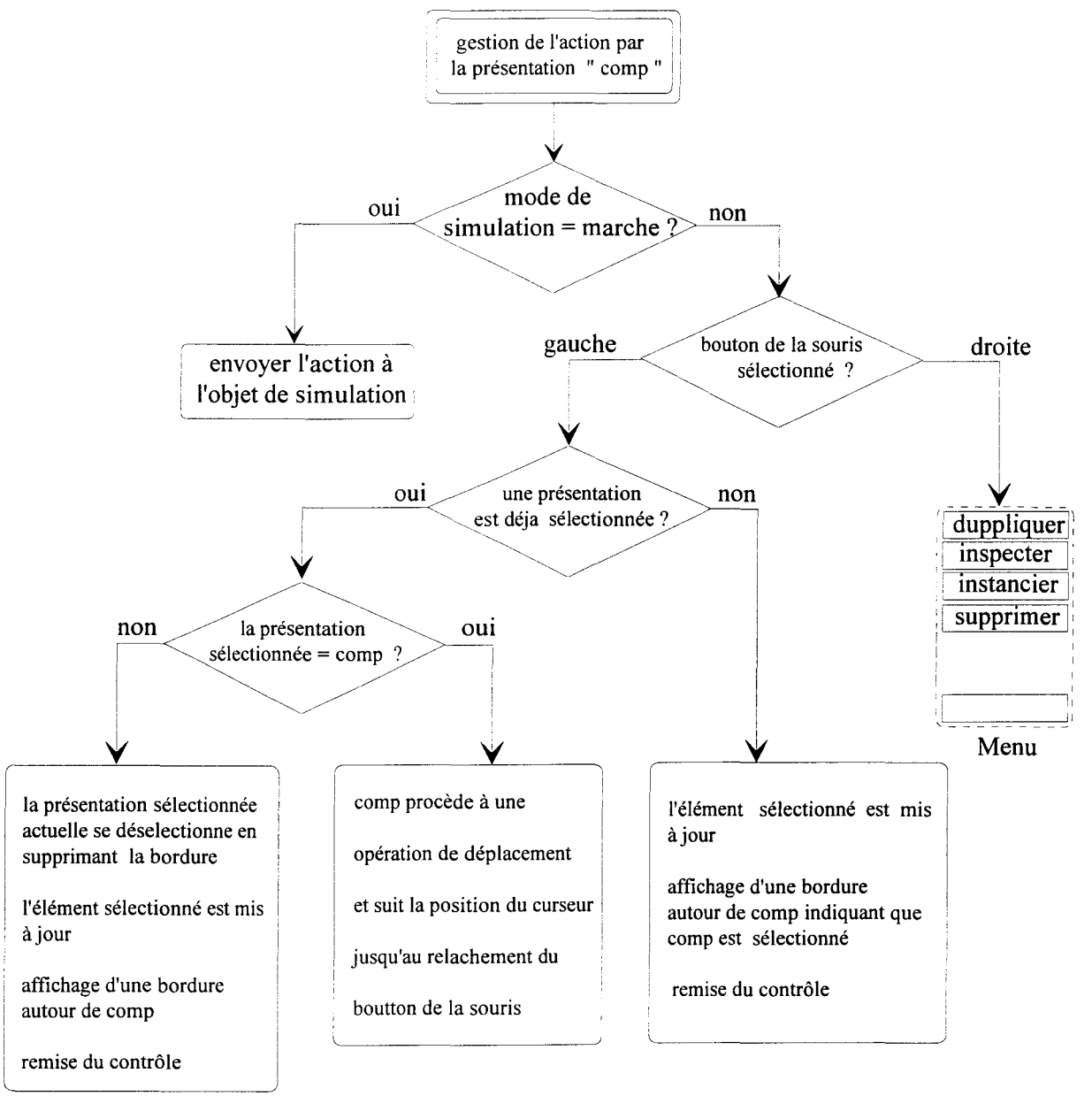


Figure IV.4 : organigramme de gestion des actions utilisateur (2)

Pour un fonctionnement cohérent de la gestion du contrôle de l'interaction, un certain protocole de communication entre le contrôleur graphique et les présentations des objets de simulation, a été établi. Ce protocole, transparent à l'utilisateur, contient les méthodes permettant d'identifier les présentations ayant besoin du contrôle ou celles permettant d'attribuer le contrôle à une présentation quelconque de la liste, etc.

Par ailleurs, les éléments de la liste peuvent être de types différents (textes, graphiques simples, graphiques composés, etc.), ce qui se traduit par des

comportements différents. Ces comportements donnent le même résultat en réponse à une même requête. Un exemple de ce principe est celui d'une opération de déplacement ou de suppression d'un élément.

Enfin, cette répartition du contrôle entre le contrôleur graphique de la fenêtre et les éléments de la simulation a permis l'obtention d'un style de dialogue évolué par manipulation directe sur les objets affichés sur l'écran.

IV.4/ DESCRIPTION DU MODELE P.OS.T [MOS 93]

Le modèle P.OS.T regroupe trois entités distinctes :

- l'abstraction représentée par le modèle d'objet de simulation décrit dans le deuxième chapitre,
- la présentation, chargée de l'aspect visuel de l'abstraction et assurant l'affichage et l'animation de l'objet de simulation,

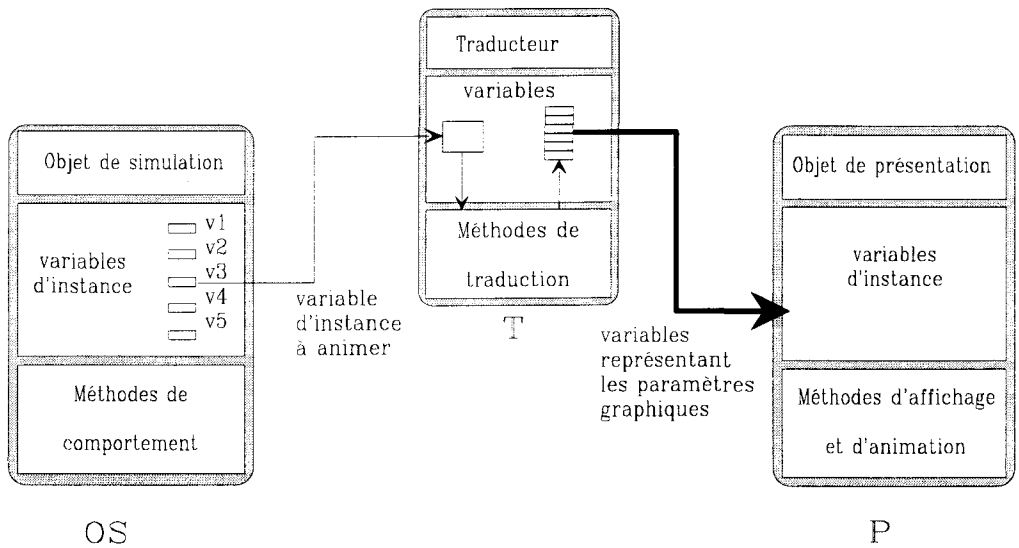


Figure IV.5 : structure du modèle P.OS.T

- le traducteur, prévu pour assurer la cohérence des domaines abstrait et graphique grâce à une bibliothèque de méthodes de traduction extensible par l'opérateur.

L'architecture globale du modèle P.OS.T est représentée par la figure IV.5.

Cette architecture s'adapte parfaitement aux règles de conception des interfaces utilisateurs décrites dans le chapitre précédent. En effet, l'objet de simulation est vu comme le composant élémentaire de l'application abstraite. Son comportement, écrit à base de contraintes, permet de faire évoluer son état dans l'environnement de simulation.

D'un autre côté, la présentation est définie par l'un des éléments graphiques suivants : rectangle, cercle, polygone, texte, etc. Une classe de base modélise les paramètres et les méthodes d'affichage communs de ces éléments. En plus, pour chaque élément graphique, il est associé une sous classe qui définit les paramètres et les comportements spécifiques.

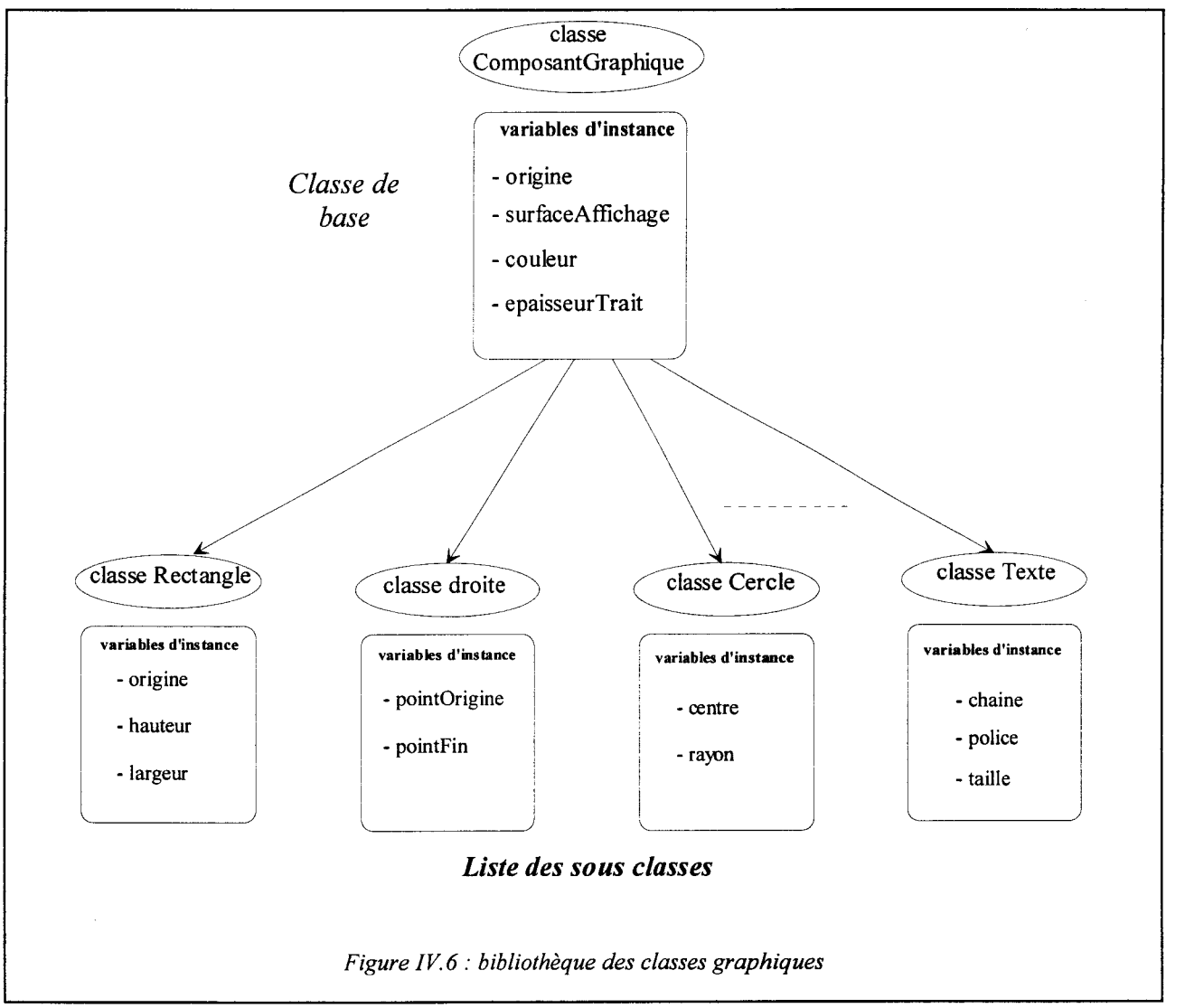


Figure IV.6 : bibliothèque des classes graphiques

Par analogie avec l'objet de simulation, la présentation constitue la composante élémentaire du synoptique. La construction de ce dernier revient donc à créer l'ensemble des présentations graphiques qui se fait séparément et en parallèle avec la construction des objets de simulation.

L'objectif de notre outil étant la simulation et l'animation du synoptique. Une communication est établie entre l'application abstraite et le synoptique à l'aide d'un pont constitué de l'ensemble des traducteurs. Chaque traducteur relie un objet de simulation à sa présentation.

Dans la mesure ou un objet de simulation nécessite une animation quelconque, le traducteur définit une ou plusieurs ***méthodes de traduction*** reliant une ou plusieurs variables de l'objet en question aux variables de la présentation correspondante.

Chaque méthode de traduction correspond à une ***contrainte de traduction*** qui est différente des contraintes des objets de simulation par sa fonctionnalité, même si sa résolution est provoquée par le même moteur de simulation MARC.

En effet, à chaque fois qu'un objet de simulation est invité à résoudre les contraintes décrivant son comportement, il demande à son traducteur de vérifier la stabilité de ses contraintes de traduction, c'est-à-dire si ses variables concernées par l'animation ont changé. Lorsqu'une ***contrainte de traduction*** est instable, le traducteur exécute automatiquement la ***méthode de traduction*** correspondante.

Voici un exemple simple de contrainte de traduction dans le cas d'un objet de simulation ***compteur*** dont la valeur est contenue dans une variable *compt.* L'animation choisie est la représentation alphanumérique. La présentation correspondante est donc un objet *texte* avec comme variable caractéristique une chaîne de caractères *chaîne*.

Lors de la vérification de stabilité, le traducteur regarde si la valeur de *compt* a changé. Dans ce cas, la résolution de la contrainte est assurée par la relation suivante :

chaîne := compt printString.

PrintString est une méthode Smalltalk qui permet de transformer une valeur en chaîne de caractères. L'animation du même objet peut être envisagée autrement. Pour cela, il faudrait définir le type de la présentation, identifier les variables caractéristiques correspondantes et enfin établir une relation entre ces variables et la

variable *compt*. Nous verrons dans le chapitre suivant quelques exemples sur les possibilités d'animation.

IV.5/ COMPOSITION DES OBJETS

La composition peut apparaître à deux niveaux :

- la composition élémentaire des composants graphiques,
- la composition des modèles P.Os.T.

IV.5.1/ Composition des objets graphiques

Cette composition est indispensable pour une représentation plus parlante de l'état des objets de simulation. Elle consiste à représenter un objet par une composition d'éléments graphiques simples de types différents : rectangles, cercles, textes, ...etc.

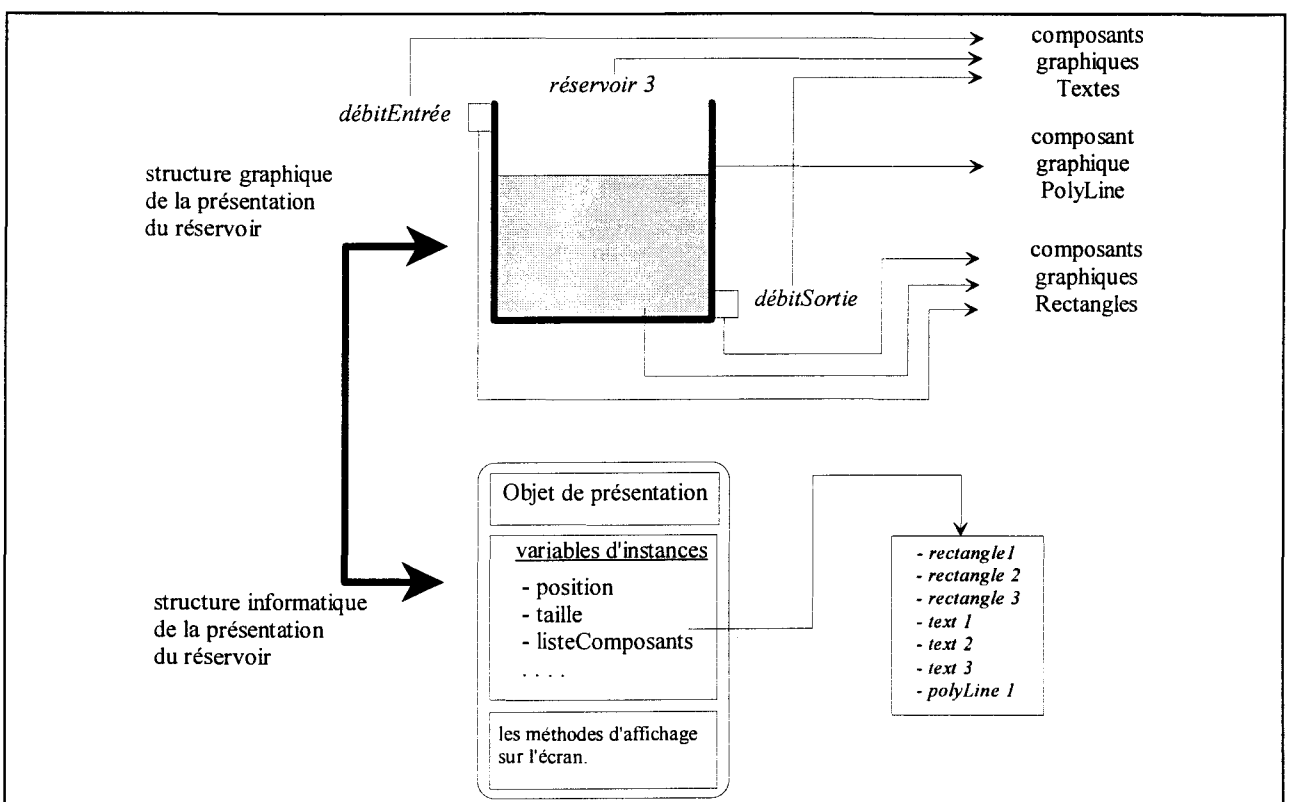


Figure IV.7 : composition graphique

Cette technique présente de nombreux avantages :

1) D'abord la composition permet une meilleure représentation graphique de l'objet de simulation par l'adjonction de plusieurs commentaires textuels ou des symboles graphiques explicites de l'objet représenté (figure IV.7). Ainsi la structure informatique de la présentation devient un peu plus complexe. En effet, l'objet présentation est constitué d'un ensemble de variables correspondant aux paramètres d'affichage : position, largeur, hauteur, ... etc. Cette même présentation fait référence, à l'aide d'une autre variable, à une liste de composants graphiques élémentaires ayant chacun ses propres paramètres d'affichage : position (relative à la présentation), largeur, hauteur, couleur, etc. Lors de l'affichage, la présentation fait appel à l'ensemble des composants graphiques de la liste, pour s'afficher un à un, en tenant compte de la position de référence commune de la présentation.

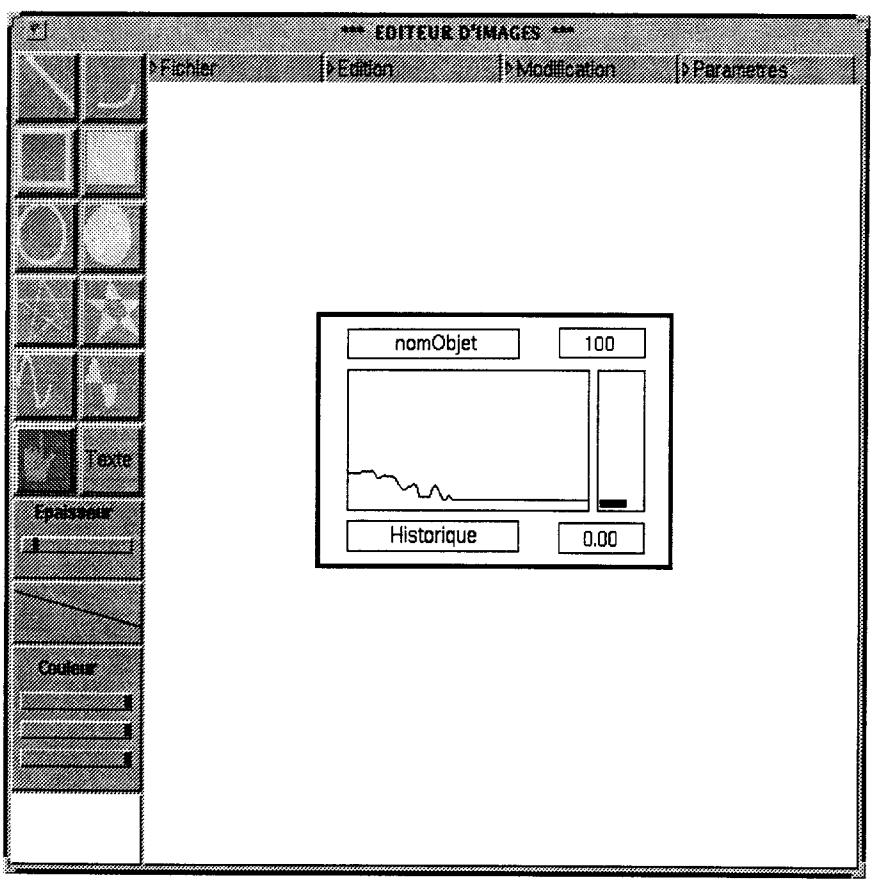


Figure IV.8 : éditeur graphique des présentations

Lors de la phase de conception de la simulation, un éditeur graphique a été développé spécialement à cet effet, afin de permettre à l'utilisateur de choisir sa propre présentation. Cet éditeur contient un ensemble de boutons de fonctions associées aux différents types d'éléments graphiques existants.

Equipé du système de menus et de boutons de réglage, l'opérateur peut à volonté choisir ses propres paramètres d'affichage : couleur, largeur de trait, etc.

Enfin, dès lors que l'utilisateur aura fini de construire sa présentation, il pourra la sauvegarder dans une bibliothèque de fichiers, afin de la récupérer, soit pour la modifier, soit pour l'affecter à un objet de simulation lors de la phase de prototypage.

2) Le deuxième avantage est la possibilité de visualiser l'évolution de plusieurs variables d'un même objet de simulation en associant à chaque variable, un élément graphique (figure IV.9).

Dans ce cas de figure, le traducteur définit autant de contraintes que de variables à animer. Chaque contrainte correspond à une méthode de traduction qui relie une variable de l'objet de simulation à une ou plusieurs variables du composant graphique correspondant.

Lors de l'animation, le traducteur vérifie de manière séquentielle la stabilité de chaque contrainte. Pour chaque cas d'instabilité, il lance la méthode de traduction correspondante afin de satisfaire la contrainte.

3) Le troisième avantage est la possibilité de représenter une même variable par différentes manières. On peut par exemple visualiser une même température à la fois sous forme alphanumérique et sous forme graphique.

Dans ce cas, le traducteur définit deux contraintes différentes : la première relie la variable *température* à un composant graphique *texte* avec un paramètre d'affichage *chaîne* et la deuxième relie la même variable *température* à un autre composant graphique *rectangle* avec le paramètre d'affichage *hauteur*.

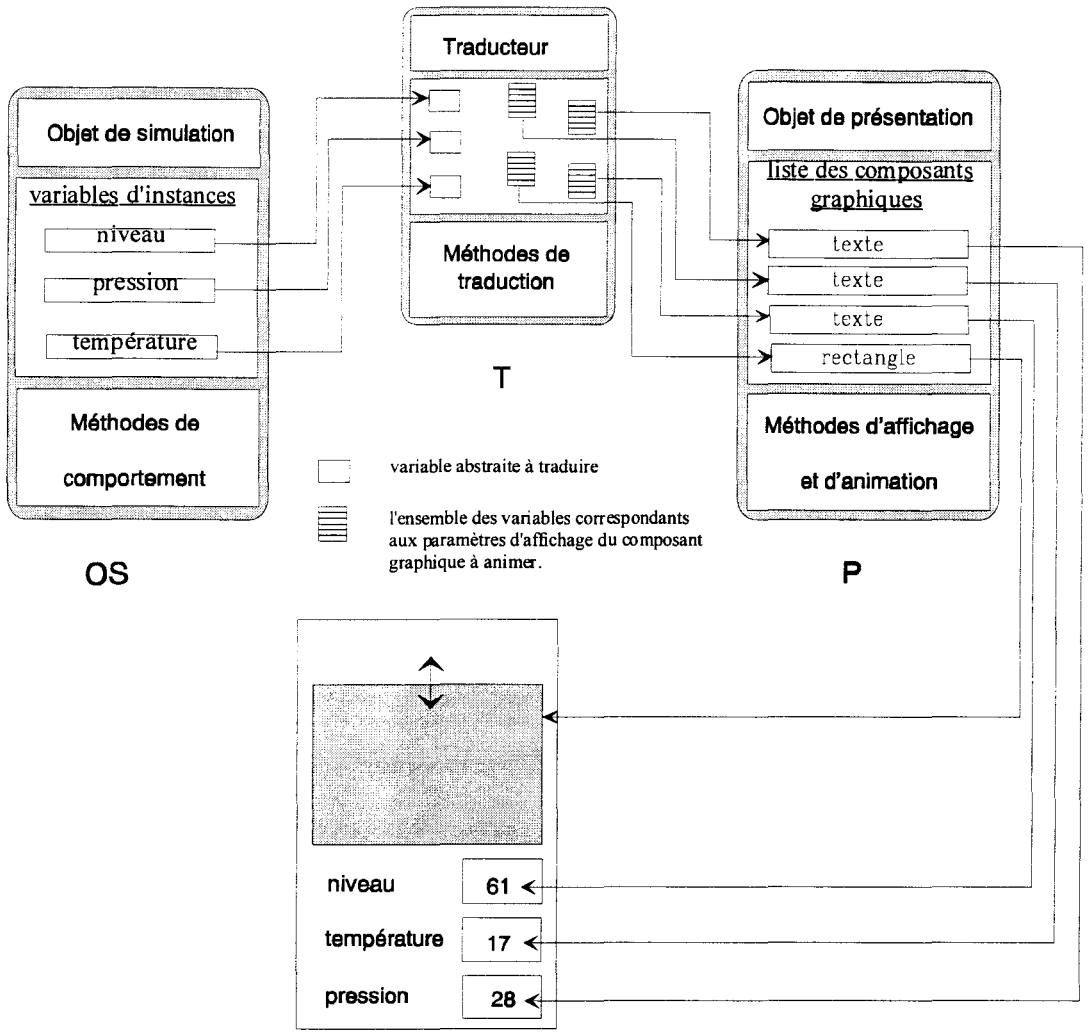


Figure IV.9 : représentation graphique de plusieurs variables

La résolution des deux contraintes, dans ce cas, se traduit en termes de méthodes de traduction par les deux relations suivantes :

l'une concernant le texte (*chaîne := température printString*),

et l'autre concernant le rectangle (*hauteur := température*).

4) Enfin, le dernier avantage est l'optimisation considérable de la zone mémoire occupée lors de la sauvegarde des présentations graphiques d'une part, et du temps consacré à l'animation d'autre part. Ceci a été constaté en comparant cette solution avec un autre type de présentation utilisant des objets images.

Un objet image est défini comme une zone mémoire rectangulaire composée d'un ensemble fini de points (pixels). Chaque point possède ses propres caractéristiques

d'affichage et peut être codé sur un ou plusieurs bits selon la profondeur utilisée par l'image. Dans une image avec une profondeur de 1, les pixels prennent les valeurs 0 ou 1, alors qu'avec une profondeur de 2, les valeurs vont de 0 à 3 inclus.

En plus de la profondeur, l'image définit une palette qui détermine comment seront interprétées les valeurs des pixels. Une palette est donc représentée par une collection de couleurs associées aux valeurs que peuvent prendre les pixels.

Prenons l'exemple d'une image ayant une profondeur de 2, les valeurs des pixels sont 0, 1, 2, ou 3 auxquels on peut associer la palette suivante [blanc, jaune, rouge, noir]. Tous les points, codés par la valeur 2, seront affichés sur l'écran avec la couleur rouge et ainsi de suite.

Sauvegarder une image revient à sauvegarder les quatre paramètres suivants : la taille de l'image (*taille*), la profondeur (*profondeur*), la collection des couleurs utilisées (*palette*) et la collection des valeurs de tous les pixels. Ceci présente un grand inconvénient car l'information à sauvegarder est dix fois plus importante que celle que nous avons adoptée.

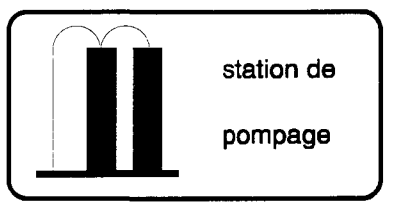
Le deuxième inconvénient se pose au niveau de l'animation. En effet, dans une image il n'y a pas de frontière entre la partie statique et la partie dynamique ce qui rend difficile la conception d'une méthode d'animation. D'un autre côté, le rafraîchissement d'une image se fait en un temps beaucoup plus important que celui des composants graphiques simples.

IV.5.2/ Composition des modèles P.OS.T

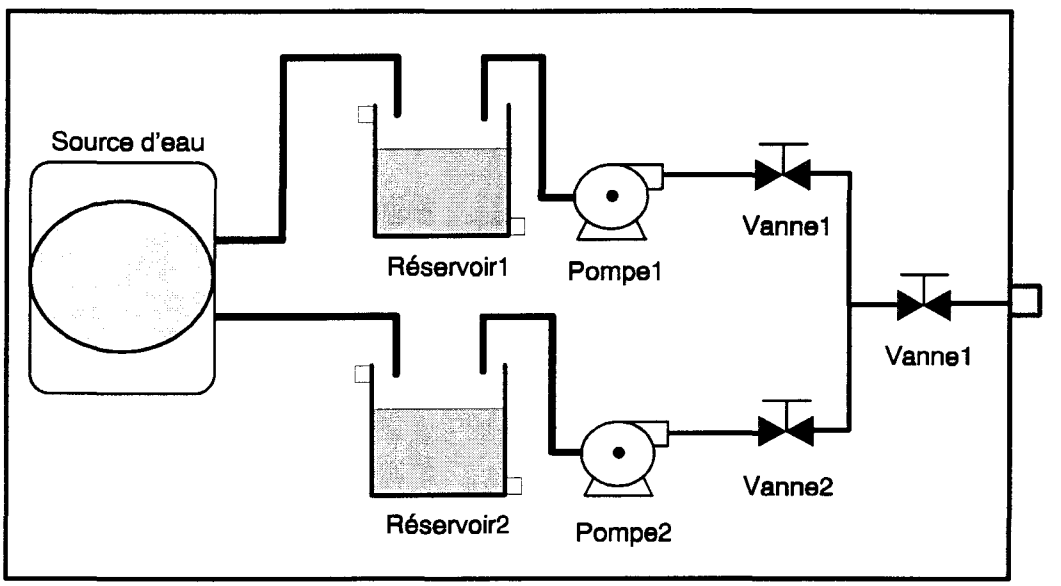
Cette composition permet de structurer toute la plate-forme de simulation sous forme hiérarchique. Au niveau perceptuel, cette présentation est sans doute la mieux adaptée à l'opérateur puisqu'elle réduit au minimum la charge mentale de ce dernier et l'interprétation devient plus facile.

Ainsi, grâce à la composition, on obtient un modèle P.Os.T structuré de la même manière que les modèles P.Os.T simples, c'est à dire : un objet de simulation, une présentation et un traducteur. La particularité de la partie objet de simulation est la définition d'une nouvelle variable *objets,* une liste de tous les objets le constituant.

Chacun de ces objets fait partie d'un modèle P.Os.T simple ou composé. Un modèle P.Os.T composé est donc visualisé de deux manières différentes : composée (la présentation du modèle lui-même) ou détaillée (l'ensemble des présentations des objets constituants) (figure IV.10).



1. Présentation composée



2. Présentation détaillée

Figure IV.10 : présentation graphique des objets composés

- Dans le premier cas, la présentation fait appel aux différents constituants pour que chacun affiche sa présentation graphique, et définit une contrainte spatiale pour la position de chacun d'eux.
- Dans le deuxième cas, on affiche la présentation du modèle lui-même qui doit être bien choisie pour qu'elle soit représentative des objets de simulation composés.

IV.6 / L'INTEGRATION DU MODELE P.OS.T DANS L'INTERFACE

Rappelons que l'interface de simulation que nous avons développée, est constituée de deux parties essentielles : la première est relativement figée et regroupe la fenêtre principale et des sous-fenêtres de fonctions (boutons), la deuxième concerne l'ensemble des modèles P.Os.T représentant l'application interactive.

La deuxième partie est entièrement conçue par l'utilisateur de l'interface. Cette possibilité présente un avantage primordial en termes de spécification fonctionnelle et de choix ergonomiques des entrées et des sorties.

En effet, cette souplesse permet à l'interface de s'adapter à la diversité des utilisateurs et les autorise à personnaliser leur interface à volonté, en fonction de leurs connaissances. On obtient ainsi des synoptiques différents pour un même modèle de simulation.

A un niveau plus élevé de spécification, l'interface de simulation est basée sur un modèle conceptuel orienté vers une métaphore du monde réel. Ce modèle est constitué d'un ensemble d'*objets de simulation* correspondants aux objets matériels et immatériels du système à simuler. Les objets sont reliés entre eux par des objets *liens.* Toutes les opérations permettant la manipulation de ces objets sont prévues dans des menus. Il s'agit bien sûr de l'installation d'un objet, sa suppression, sa duplication, etc.

Pour une bonne convivialité de l'interface, des considérations ergonomiques sont adoptées concernant la spécification fonctionnelle. Il était question de respecter les critères d'universalité, de cohérence et de flexibilité.

En plus de l'intérêt qu'apporte la structure interne des modèles P.Os.T dans l'édition du synoptique et son animation, son principe de composition, présenté dans le paragraphe précédent, a fortement contribué à l'intégration de ces modèles dans l'interface de simulation et à l'application du principe de navigation.

Grâce à ce principe de navigation, il est possible :

- de montrer à l'utilisateur une vision globale de l'application lui permettant de voir et de comprendre ce qui peut être fait, en utilisant des représentations iconiques et une métaphore adaptée;
- de fournir automatiquement aux utilisateurs ce dont ils ont besoin selon le contexte;
- de se promener dans la base de donnée pour accéder, par exemple, à un objet particulier se trouvant au milieu de la hiérarchie.

IV.6.1 / Navigation Verticale et Horizontale

La base de données à gérer a été structurée de manière à décomposer le modèle de simulation en trois parties (cette décomposition n'est pas unique) :

- synoptique : vue des objets du système à simuler (ex : vannes, réservoirs, ...),

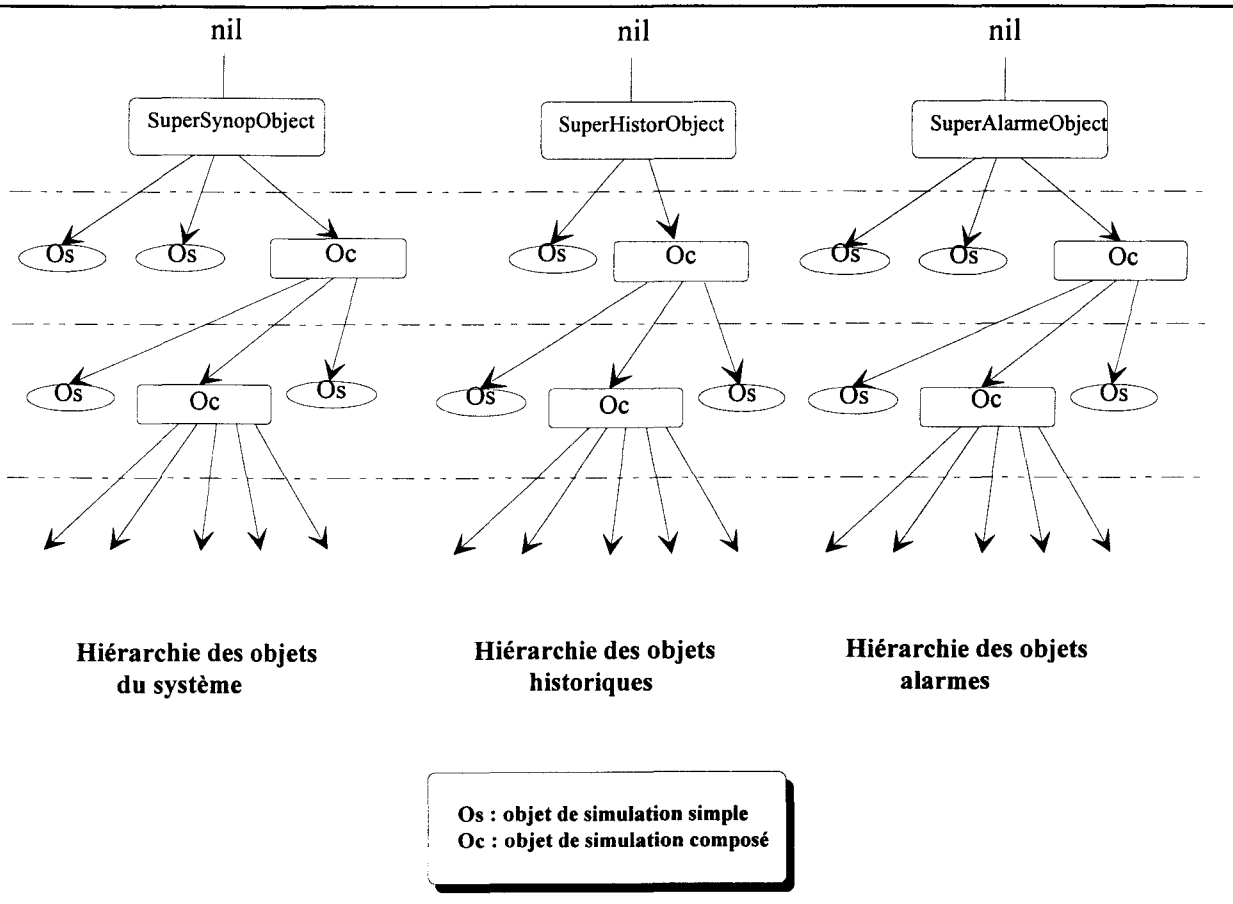


Figure IV.11 : structure de la base de données

- historique : vue de l'ensemble des historiques visualisant l'évolution des variables des objets du système en question,
- alarme : vue de l'ensemble des alarmes indiquant l'état des variables des objets du système. Ces alarmes peuvent être utilisées pour le pilotage et la surveillance du système ou le diagnostic dans le cas de pannes.

De ce fait, trois super-objets composés sont crées pour représenter le sommet de chacune des trois hiérarchies. On obtient la structure de la figure IV.11.

IV.6.2 / Navigation verticale

Cette opération permet de se promener, au sein d'une même hiérarchie, pour accéder à n'importe quel niveau. Pour chaque hiérarchie on définit un objet composé mobile qui, à tout moment, fait référence à un objet de la hiérarchie et indique au contrôleur graphique la liste des objets à afficher sur l'écran.

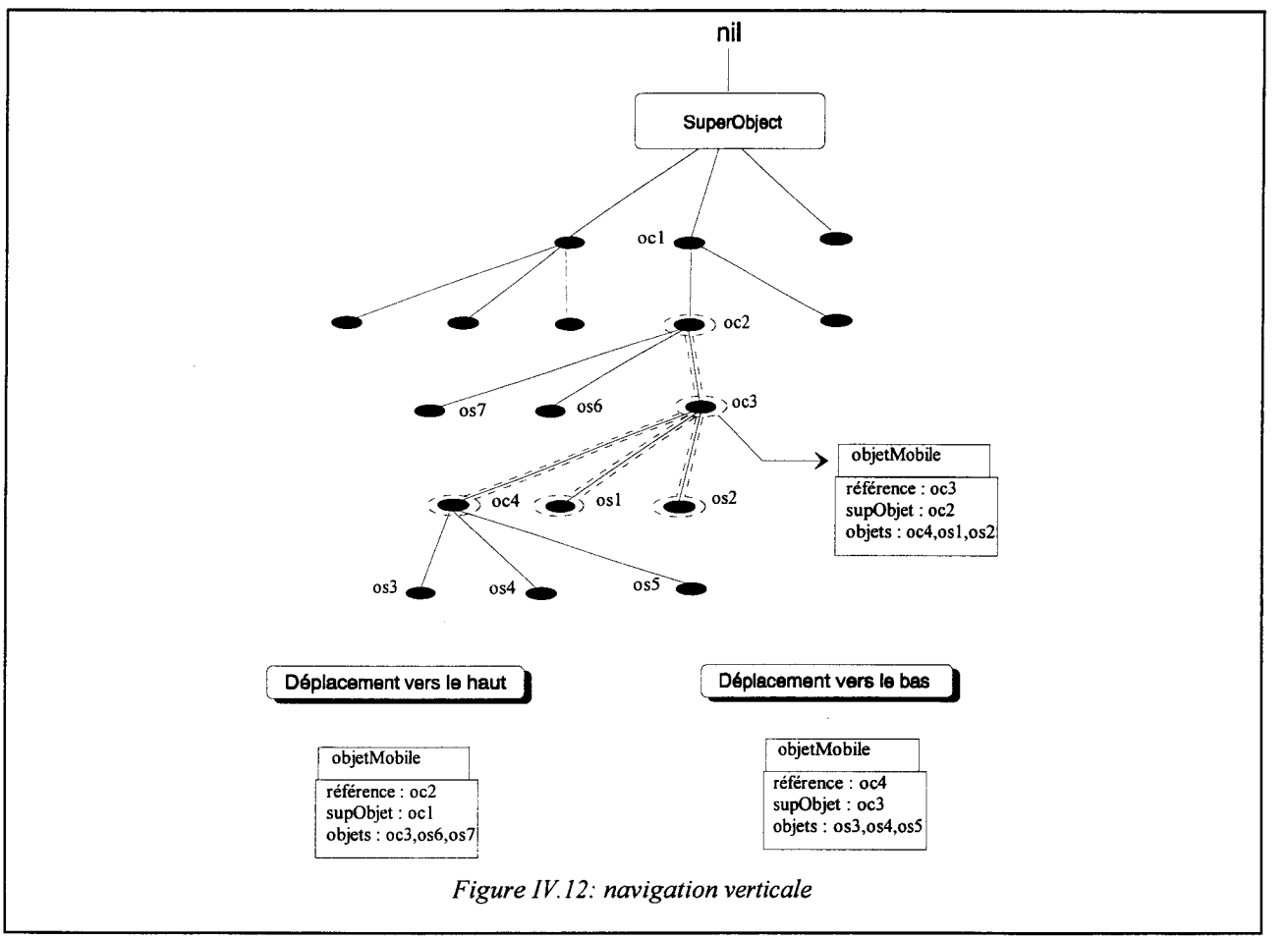


Figure IV.12: navigation verticale

Il lui permet également, de passer du niveau actuel au niveau supérieur ou inférieur. Le passage d'un niveau à un autre se traduit par le déplacement de l'objet mobile en remplaçant le pointeur actuel par un pointeur :

- sur le super objet dans le cas d'un déplacement vers le haut, ou bien
- sur l'objet sélectionné, dans le cas d'un déplacement vers le bas (figure IV.12).

IV.6.3 / Navigation horizontale

Contrairement à la navigation verticale, cette opération permet de passer d'une hiérarchie quelconque à une autre, parmi les trois structures : synoptique, historique, alarme.

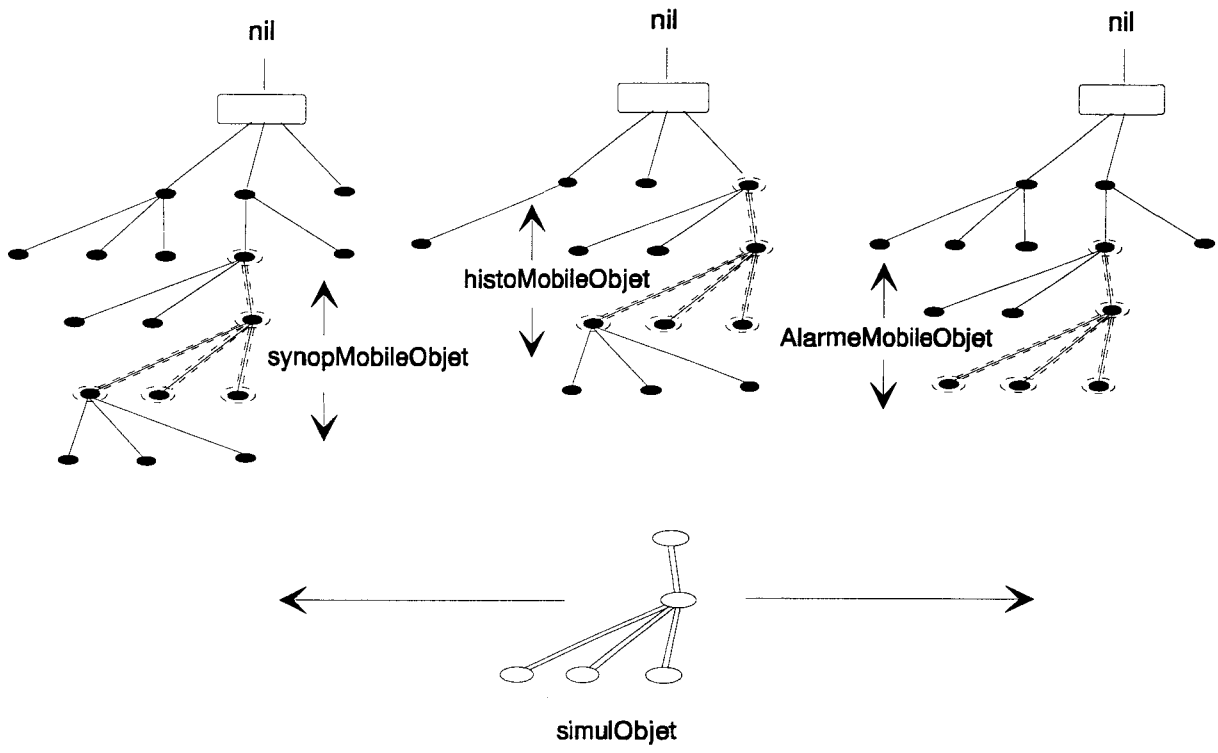


Figure IV.13: navigation horizontale

Pour cela, et en plus des trois objets mobiles permettant la navigation verticale dans chacune des trois hiérarchies, on a défini un nouvel objet composé afin de gérer le passage d'un objet mobile à un autre.

IV.7 / LES PHASES DE CONCEPTION D'UNE SIMULATION INTERACTIVE

L'outil de simulation que nous avons développé, se situe dans la catégorie des générateurs d'applications interactives. Il permet en effet aux concepteurs d'aller de la construction de leurs propres modèles jusqu'à la simulation et l'animation de leurs synoptiques. Cette opération peut être décomposée en trois phases essentielles : prototypage, construction du synoptique et enfin simulation interprétation des résultats.

IV.7.1 / Prototypage

La réussite d'une simulation dépend énormément de la phase de prototypage. Comme son nom l'indique, cette phase consiste à construire une bibliothèque de prototypes génériques de modèles de simulation (P.Os.T). Chaque prototype complet regroupe un prototype d'objet de simulation, un prototype de présentation et un prototype de traducteur. Ainsi, cette phase composée passe par trois phases élémentaires pour créer chacun des trois prototypes.

Nous avons vu dans les paragraphes précédents, comment il est possible de créer des modèles de présentations graphiques et de les sauvegarder dans une première bibliothèque dite de prototypes de présentations. Ceci permet d'assurer une partie de la phase de prototypage. Une deuxième partie, en parallèle, consiste à créer des modèles génériques d'objets de simulation à contraintes.

L'état actuel de notre travail permet de la réaliser en se servant du "*browser*" de classes de l'environnement Smalltalk, ce qui limite l'utilisation aux seuls connaisseurs de la programmation de ce langage entièrement orienté objet. Pour les autres, ils pourront se contenter d'une bibliothèque de prototypes prévue pour une utilisation minimale du simulateur.

En termes de langage à objet, le prototypage d'un modèle d'objet de simulation est le résultat de la création d'une instance de la classe à laquelle appartient le modèle. Cette instance est, à l'origine, à l'état vierge car ses variables de fonctionnement ne sont pas encore affectées.

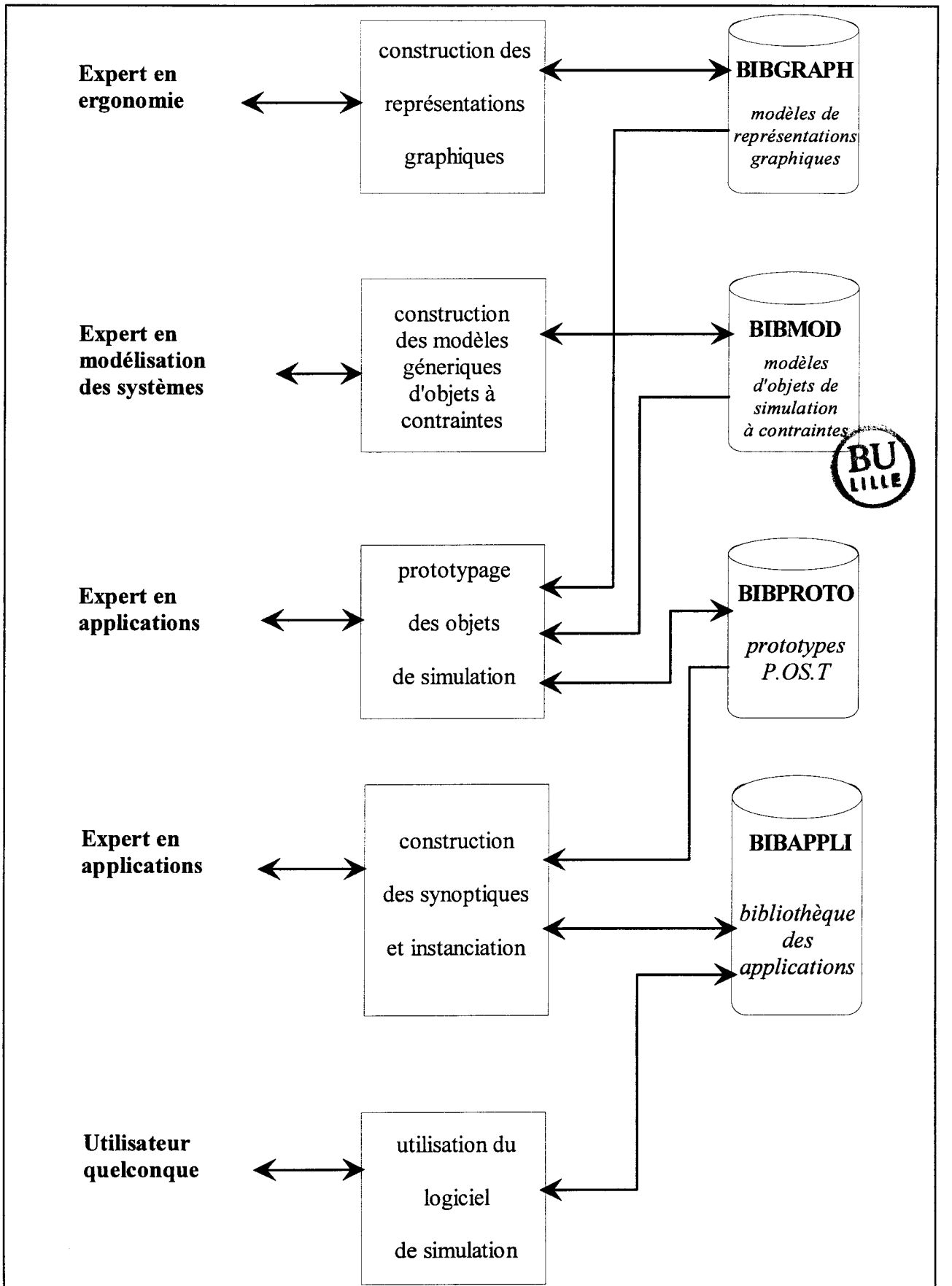


Figure IV.14 : les phases de conception d'une simulation interactive

Afin que le prototype ait un sens dans la simulation, il faudrait l'initialiser grâce à une méthode d'initialisation. L'utilisateur pourra à volonté modifier cet état initial grâce à des outils prévus à cet effet.

Enfin, une fois que le prototype d'objet de simulation et celui de la présentation sont prêts, la dernière partie consiste à créer un prototype traducteur qui viendra se situer entre les deux premiers pour former ainsi un prototype P.Os.T complet.

De plus, tout en étant dans la fenêtre de prototypage, l'utilisateur devra sélectionner un à un les composants graphiques qu'il souhaiterait animer et leur affecter à chaque fois une méthode d'animation qu'il aura choisie dans la bibliothèque des méthodes de traduction.

Une fois que la phase de prototypage est terminée, le modèle est sauvegardé d'une part dans un fichier pour une utilisation ultérieure, et d'autre part, dans une variable d'instance de la classe à laquelle appartient l'objet de simulation, pour une utilisation immédiate.

IV7.2 / Construction du synoptique

Cette phase est entièrement réalisée par l'interface d'édition. Elle consiste à construire graphiquement le synoptique du modèle de simulation. L'utilisateur dispose pour cela, de la bibliothèque des prototypes (P.Os.T) sous forme de menu. D'un point de vue perceptuel, la sélection d'un modèle quelconque dans le menu fait apparaître la présentation graphique sur l'écran qui va suivre la position du curseur de la souris, en attendant la validation de sa position finale dans le synoptique.

Du point de vue conceptuel, aussitôt que ce positionnement graphique est fait, une ***copie*** du prototype est réalisée, suivie de son ***installation*** dans l'environnement de simulation.

IV.7.2.1 / copie des modèles P.Os.T

Le prototype du modèle P.Os.T est une entité complexe et sa duplication est aussi complexe. Une opération de copie standard consiste à créer une autre instance de la

même classe et d'initialiser ses variables d'instances en leur affectant les mêmes valeurs que celles du prototype d'origine. Cette technique n'est efficace et valable que pour les objets dont les valeurs des variables d'instances sont des valeurs numériques ce qui n'est pas le cas lorsqu'une variable pointe sur un autre objet (encapsulation). La figure IV.16 montre le résultat d'une copie d'un objet dans le cas d'une encapsulation.

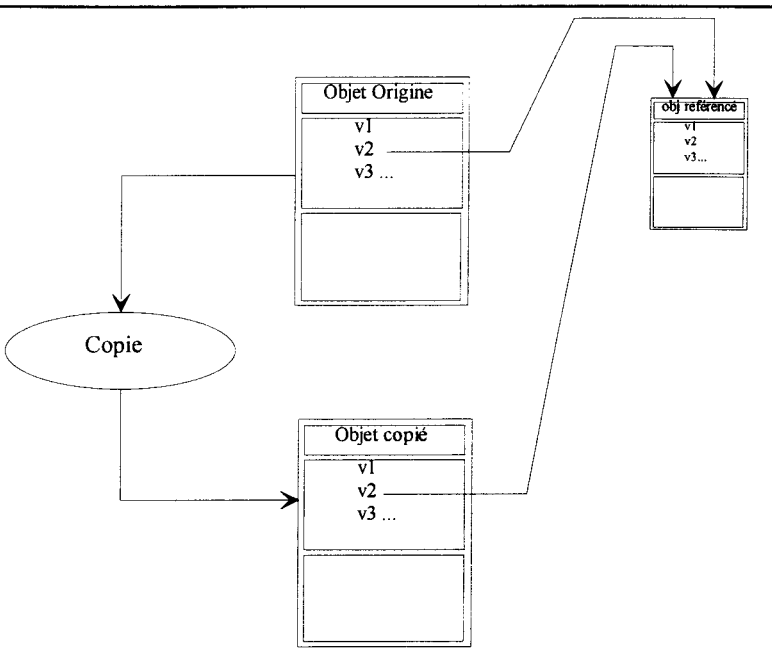


Figure IV.15: copie standard d'un objet

La copie obtenue est partielle. C'est pourquoi une méthode de copie à été créée spécialement pour ces modèles P.Os.T afin d'avoir, après une opération de copie, deux entités entièrement indépendantes. Cette méthode consiste, en plus de la copie partielle, à copier les objets encapsulés. Cependant, un problème apparaît dans le cas ou deux variables d'instances de l'objet copié font référence à un même objet. C'est le cas lors de la copie du traducteur dont deux de ses variables d'instances font respectivement référence à la présentation et à un composant graphique de la même présentation (le composant concerné par l'animation).

En effet, le résultat obtenu est une copie dont les deux variables d'instance en question ne pointent plus sur un même objet mais sur deux objets différents.

De ce fait, une mise à jour de l'objet copié est indispensable et consiste dans ce cas à réaffecter la deuxième variable en identifiant l'élément graphique de la présentation copiée correspondant à l'élément graphique copié (figure IV.16).

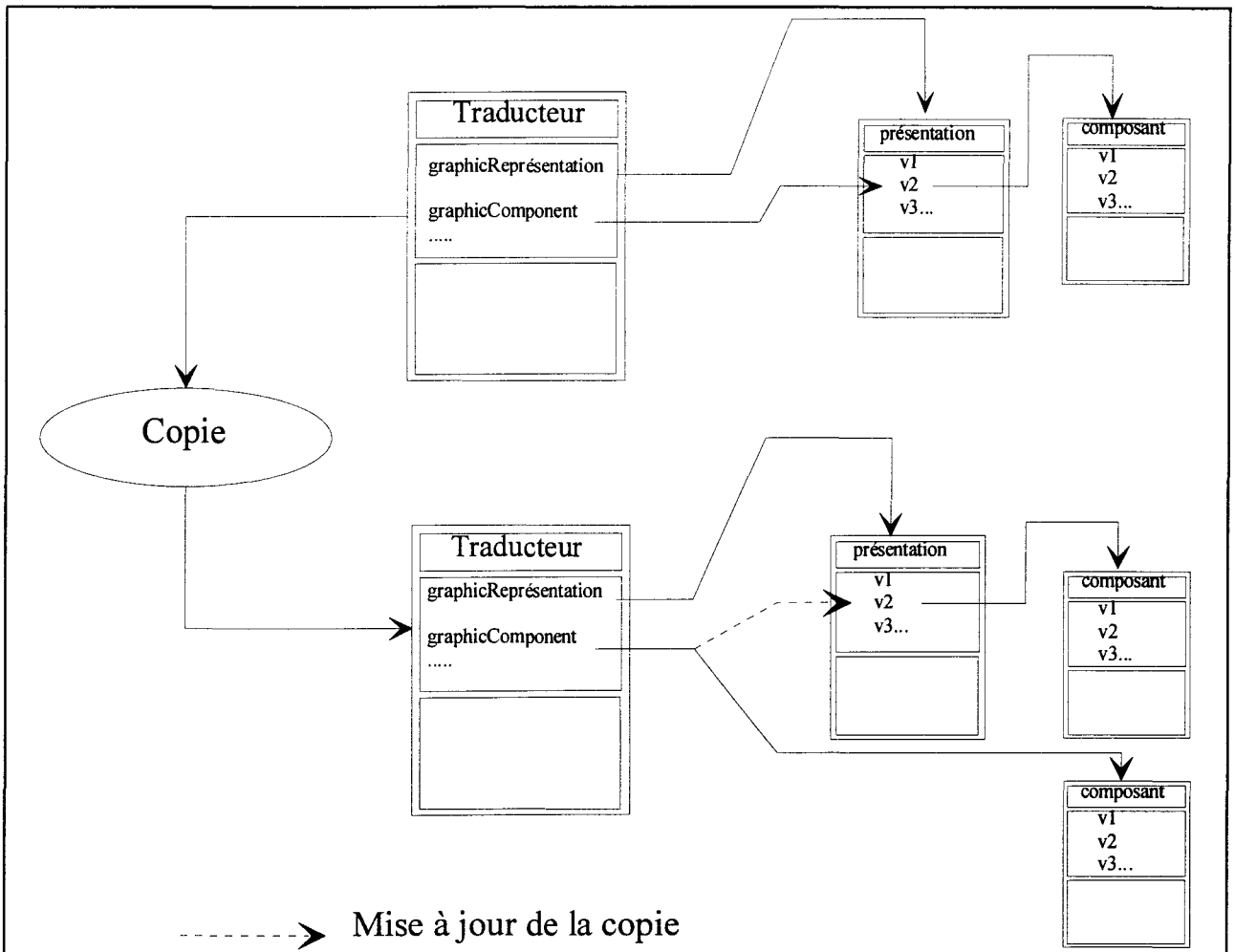


Figure IV.16: copie spéciale des objets de simulation

IV.7.2.2 / Installation et suppression des modèles P.Os.T

Dans l'environnement de simulation, il faut distinguer deux parties : le moteur de simulation qui met en oeuvre la simulation et le contrôleur graphique qui gère l'affichage et les actions de l'utilisateur (Figure IV.17).

Lors de l'installation d'un modèle P.Os.T, l'objet de simulation est rajouté à une liste des objets déjà installés appartenant au moteur de simulation, la présentation correspondante est rajoutée à une liste des présentations affichées sur l'écran appartenant au contrôleur graphique de la fenêtre.

Lorsque l'utilisateur décide de supprimer un objet de l'application, il suffit de le sélectionner grâce à la souris et faire appel au menu pour choisir l'option *supprimer.*

Suite à cette action, une procédure inverse est déclenchée et consiste à enlever l'objet de la liste du moteur, la présentation de la liste du contrôleur, suivie d'une vérification de ses variables d'instances pour savoir si cet objet est lié aux autres objets de simulation. Auquel cas, il faut casser les liens et les remplacer par leurs valeurs numériques.

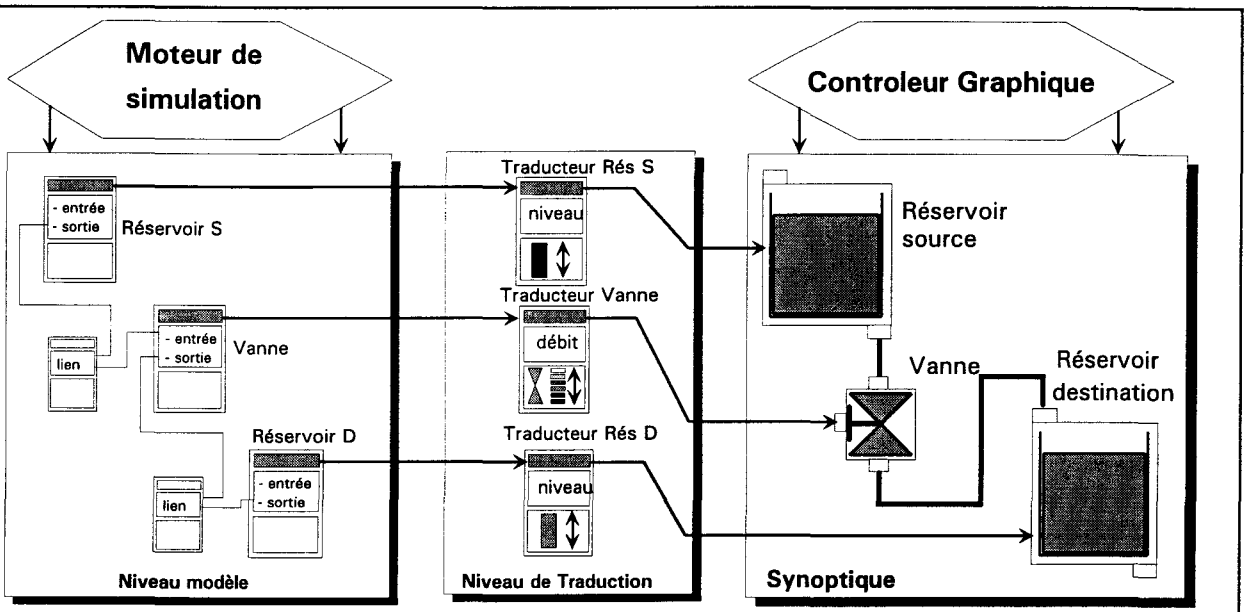


Figure IV.17 : architecture globale de la plate-forme de simulation

L'opération de suppression de l'objet ne s'arrête pas là car ce dernier occupe une zone mémoire non négligeable qui ne sera pas libérée puisqu'il n'y a pas de méthodes explicites pour le faire. Le cumul des objets supprimés peut engendrer un dépassement de la capacité mémoire de la machine et son blocage.

Le seul moyen disponible pour libérer les mémoires inutilement occupées est le gestionnaire de mémoire qui procède par identification des objets non accessibles depuis un moment donné. Cette opération est souvent appelée ***garbage collection*** ou ***ramasse-miettes***.

De ce fait, tous les objets P.Os.T supprimés de l'environnement de simulation subissent automatiquement une opération de destruction en plusieurs objets élémentaires indépendants, non accessibles, afin qu'il puissent être identifiés par le gestionnaire de mémoire.

IV.7.3 / Instanciation

L'instanciation consiste simplement à affecter des valeurs initiales aux variables des objets de simulation afin de spécifier leurs caractéristiques (physiques et fonctionnelle) propres. Pour cela, on peut procéder de deux manières différentes :

instanciation par objet (figure IV.18) : elle consiste à sélectionner l'objet en question par action directe sur la souris et à choisir l'option " *instancier* " du menu. Une fenêtre apparaît contenant toutes les variables d'instances avec leurs valeurs actuelles : l'utilisateur peut ainsi les modifier à volonté.

instanciation globale (figure IV.19) : une fenêtre contenant tous les objets de simulation répertoriés par types de classes, est prévue à cet effet. Dans ce cas, l'instanciation se fait par la sélection de l'objet, la suite de l'opération étant identique à la première méthode.

IV7.4 / Création des Liens

Les objets liens, décrits dans le chapitre I, font partie intégrante de la base de données. Ils servent à représenter les interactions entre les objets de simulation, en permettant le partage d'une même donnée à travers des variables d'instances de différents objets.

Comme dans la phase d'instanciation, la création d'un lien peut se faire de deux manières différentes :

La première consiste à choisir l'option du menu " *créerLien* " et à parcourir, un à un, les objets à lier en les sélectionnant par la souris. La sélection dans ce cas se traduit par l'apparition d'un menu propre à l'objet sélectionné et contient toutes ses variables d'instances. L'utilisateur devra à ce moment là, choisir la variable concernée par le lien en cours de création. Une fois que le parcours est terminé, il pourra cliquer sur le bouton droit de la souris pour mettre fin à l'opération.

La deuxième méthode est réalisée par une fenêtre spéciale qui contient une liste des objets de simulation, une liste des liens déjà existants et enfin une liste des

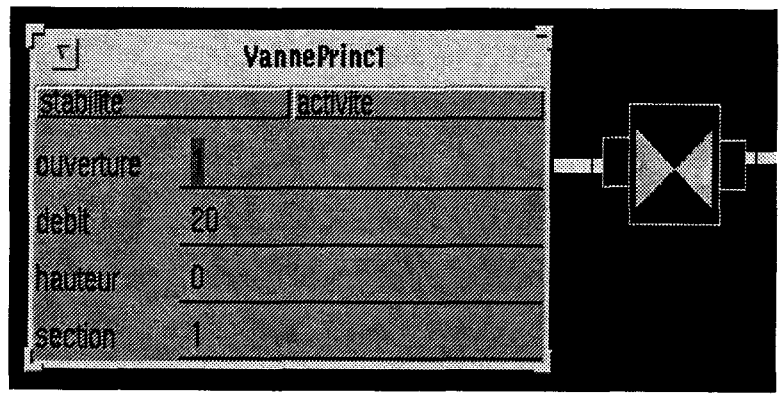


Figure IV.18 : instanciation par objet

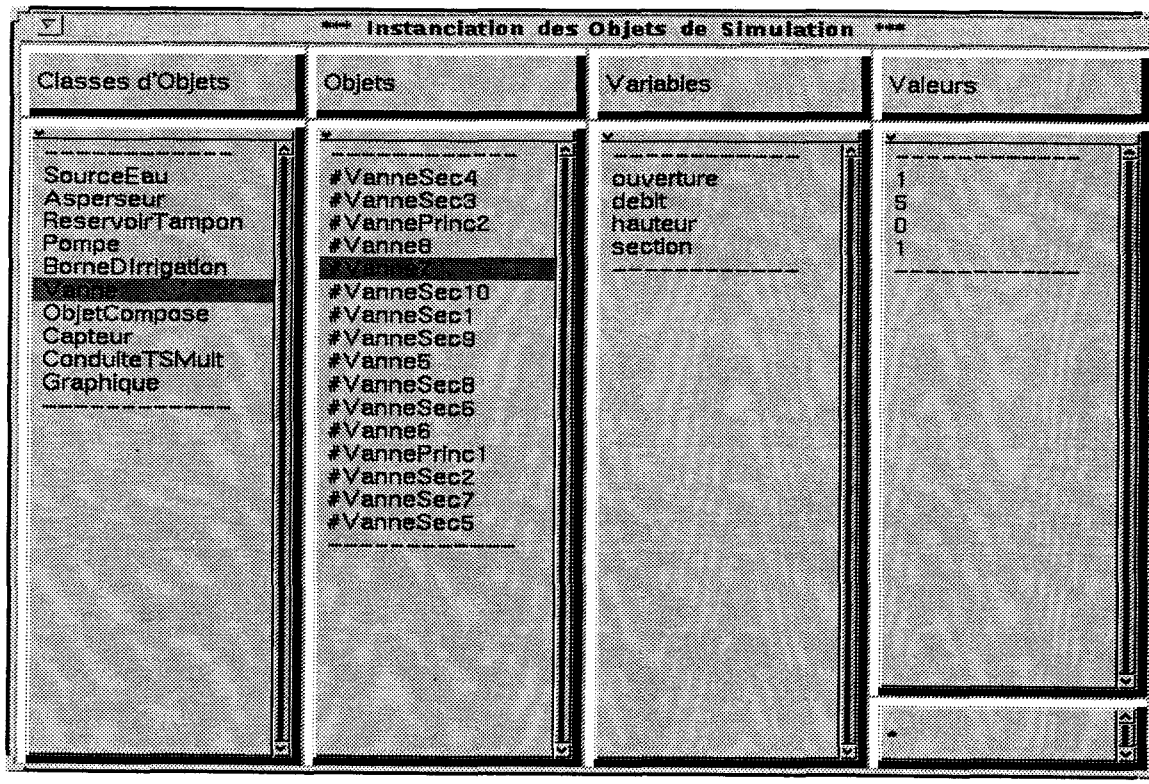


Figure IV.19: fenêtre d'instanciation

variables liées (figure IV.20). L'utilisateur peut visualiser la liste de tous les liens ou bien ceux concernant un objet particulier. Il peut aussi créer ou détruire un lien, rajouter une variable à un lien quelconque ou en supprimer.

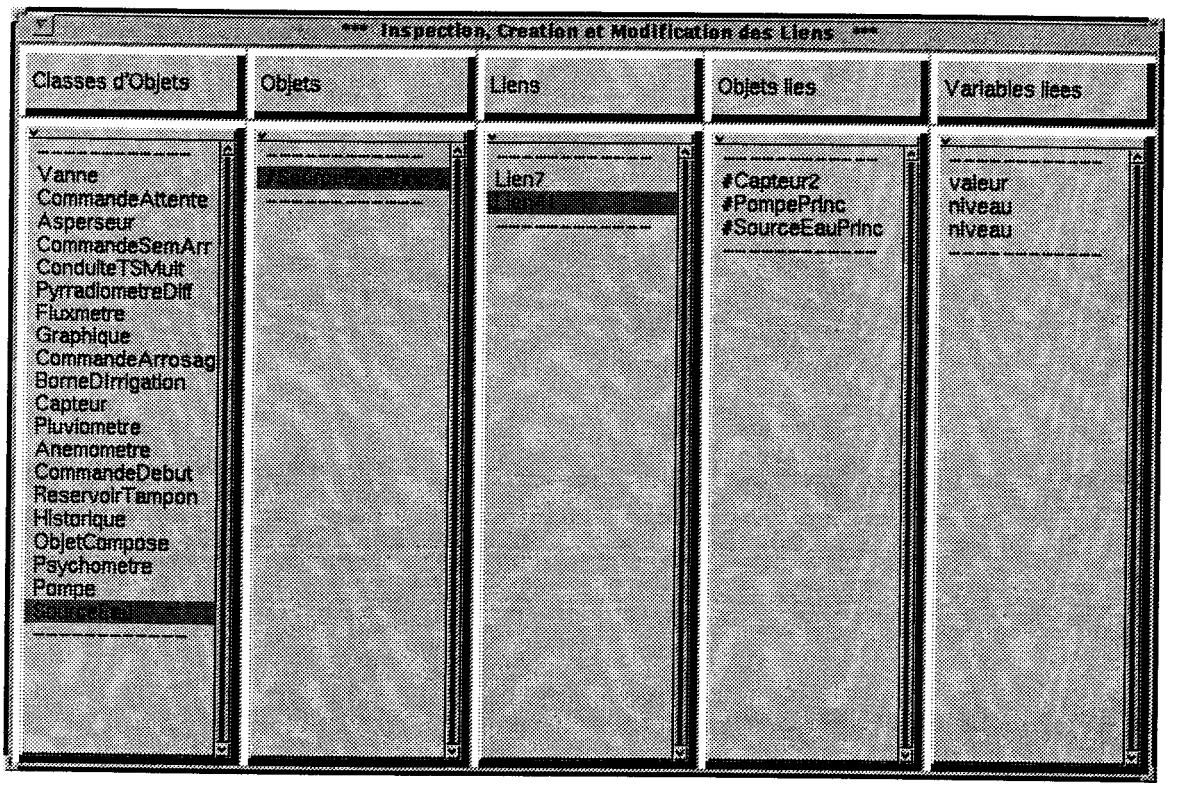


Figure IV.20 : fenêtre de création automatique des Liens

Contrairement à la méthode précédente, cette fenêtre permet de lier deux objets quelconques de l'application, situés dans deux écrans synoptiques différents.

IV.7.5 / Simulation

Dans cette dernière étape, l'opérateur est chargé de deux missions importantes : le lancement de la simulation et le pilotage du système simulé. Pour la mise en oeuvre de la simulation, deux commandes (marche/arrêt) sont prévues. Pendant la phase de pilotage, l'utilisateur dispose du même contrôle sur les objets qu'en dehors de la simulation. En plus, et comme le montre l'organigramme de gestion de contrôle en Figure IV.4, il est possible de définir des actions au niveau des objets de simulation.

Dans ce cas, l'action utilisateur interceptée par la présentation graphique est interprétée différemment. Cette action passe d'abord par une phase d'identification du composant graphique simple concerné. Ensuite, elle est envoyée à l'objet de simulation, avec comme argument la variable *identificateur* du composant précédent, lui permettant de lancer l'action correspondante.

Soit l'exemple d'un objet de simulation ***compteur*** dont la présentation est montrée par la Figure IV.21.

Deux composants graphiques *polygones* sont prévus pour incrémenter ou decrémenter la valeur du compteur. Dans ce cas, deux identificateurs 'plus' et 'moins' sont attribués aux deux composants lors de la construction de la présentation. La méthode d'analyse de l'action utilisateur est définie dans l'objet de simulation compteur et se présente sous la forme :

actionUtilisateur: unIdentificateur

identificateur = 'plus' ifTrue: [incrémentation].

identificateur = 'moins' ifTrue: [decrémentation].

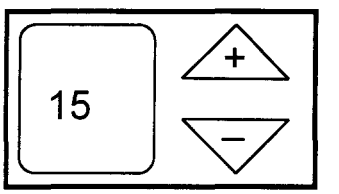


Figure IV.21 : présentation d'un compteur

IV.8 / CONCLUSION

Ce chapitre décrit les éléments de base pour la construction d'une interface de simulation interactive. Cette interface est essentiellement articulée autour d'un contrôleur graphique et des éléments de la simulation basés sur le modèle P.Os.T.

Le contrôleur graphique se charge de gérer l'interaction entre l'utilisateur et l'environnement général de la plate-forme de simulation. Il permet par exemple, de

charger une application quelconque, de faire l'instanciation du modèle, de mettre en marche ou en arrêt la simulation, etc.

Le modèle P.Os.T constitue l'élément fondamental à la construction du modèle de simulation, du synoptique et de son animation.

En effet, les objets de simulation servent à modéliser les composants du système à simuler. La présentation se charge de l'affichage de l'objet de simulation sous forme graphique et la prise en compte des actions de l'utilisateur sur ce même objet. Le traducteur quant à lui, s'occupe de la traduction de l'évolution abstraite de l'objet de simulation sous forme de paramètres graphiques de sa présentation.

Dans le chapitre suivant, nous verrons comment cette structure peut être appliquée sur des exemples concrets. Nous traiterons l'exemple du malaxeur étudié au chapitre I, un exemple de système d'irrigation automatique et enfin, un système de transport.

Chapitre V

Applications du modèle P.Os.T

Applications des modèles P.Os.T

V.1/ INTRODUCTION

Les premiers objectifs recherchés, qui ont donné naissance au modèle P.Os.T, étaient principalement orientés vers la simulation des systèmes industriels et la génération de plates-formes de simulation interactives.

La structure modulaire et générique du modèle P.Os.T fait de lui un outil performant qui répond non seulement aux objectifs précédents, mais aussi aux problèmes de création interactive d'animations d'une manière générale.

Dans le premier cas, c'est à dire la simulation des systèmes, l'application est représentée par un modèle composé d'un ensemble d'objets de simulation à contraintes. Trois exemples sont traités dans ce chapitre, le premier concerne le système malaxeur déjà décrit dans le deuxième chapitre, le second est un exemple de simulation d'un système d'irrigation automatique, et enfin un exemple de simulation des systèmes de transports automatisés dans lequel sera traitée la possibilité d'intégration des modèles à contraintes pour résoudre les problèmes de synchronisation et l'utilisation des techniques temps réel.

Dans le deuxième cas par contre, l'application est non définie à priori. Elle est représentée par un modèle quelconque, pas spécialement écrit en langage à contraintes. Ce modèle évolue dans le temps et auquel on veut associer une animation sur l'écran. Dans ce cas, des objets de simulation particuliers sont crées à cet effet et appelés objets interactifs. Ces objets possèdent le même comportement que n'importe quels objet de simulation. Leur seule contrainte est de vérifier si les valeurs des variables qu'ils

représentent ont changé ou pas afin de provoquer l'animation du synoptique. Les résultats des travaux orientés vers l'animation des algorithmes sont révélateurs.

V.2/ ETUDE DU SYSTEME MALAXEUR

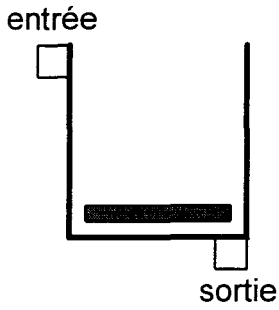
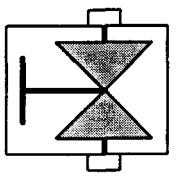
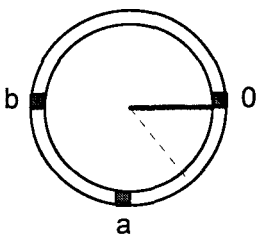
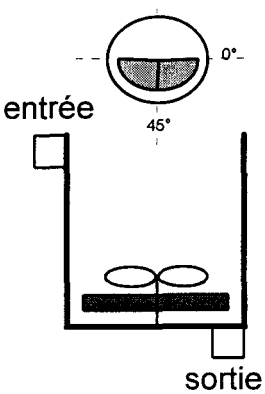
L'exemple du malaxeur est un processus chimique dont le fonctionnement et la modélisation sont détaillés dans le deuxième chapitre. La décomposition en objets de simulation a permis de construire la première partie de la plate-forme de simulation. Il convient, par la suite, de réfléchir à la construction du synoptique d'une part, et des méthodes de traduction pour créer l'animation d'autre part. Ces deux opérations se résument en une seule phase appelée phase de prototypage, vue dans le quatrième chapitre. Rappelons tout de même que le prototypage permet d'obtenir un modèle P.Os.T. générique pour un type d'objet de simulation.

Pour chaque prototype, il faut construire une présentation graphique en se servant de l'éditeur graphique. Ensuite, il faut identifier pour chaque présentation les éléments de la partie statique et ceux de la partie dynamique (à animer).

Enfin, la dernière partie, la plus importante et la plus délicate, consiste à établir une relation entre les éléments dynamiques et les variables de l'objet de simulation concernés par l'animation. Cette relation, appelée aussi contrainte d'animation, permet de modifier les paramètres des éléments graphiques (position, largeur, hauteur, couleur, etc.) en fonction de la valeur des variables de l'objet de simulation en question.

Dans l'exemple du malaxeur, la liste des objets de simulation à prototyper sont : réservoir, vanne, bascule, réservoir malaxeur, convoyeur, générateur de briquettes, capteur et l'ensemble des missions. Le tableau V.1 donne, pour chaque cas, la présentation sur l'écran, les éléments graphiques à animer et le type d'animation.

Les figures V.4 et V.5 montrent respectivement une vue détaillée du synoptique du système malaxeur et celle du synoptique de la commande (missions).

Nom du prototype	Présentation de l'objet de simulation	Variable à animer	Elément à animer	Type d'animation
Réservoir		*niveau*	*rectangle*	*changement de la hauteur du rectangle*
Vanne		*ouverture*	*polygone*	*changement de couleur du polygone*
Bascule		*poids*	*arc*	*changement des angles de l'arc*
Réservoir-Malaxeur		*niveau* *angleMel--angeur* *anglePiv--otement*	*rectangle* *2 cercles* *demi cercle*	*- hauteur du rectangle* *- rotation des cercles* *- rotation du demi cercle*
capteur		*état*	*rectangle*	*flash du rectangle*

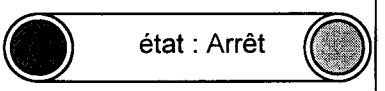
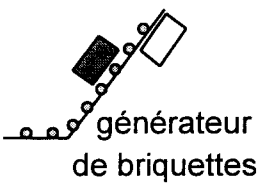
convoyeur		*marche*	*2 cercles*	*rotation*
briquette		*position*	*rectangle*	*changement de position*
tapis d'amenage		*générеUn--Produit*	*rectangle*	*flash*

Tableau V.1 : liste des prototypes d'objets de simulation (système malaxeur)

V.2.1/ Construction des méthodes de traduction

V.2.1.1/ Animation du réservoir

La contrainte à établir relie la valeur du niveau du réservoir et la taille du rectangle (figure V.1). Les paramètres graphiques concernés, dans ce cas, sont *hauteur* et *position*.

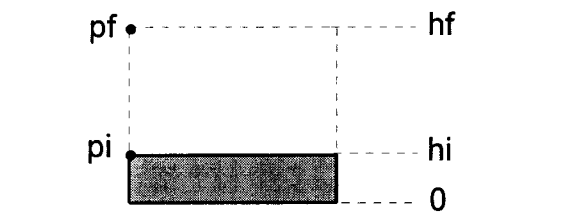


Figure V.1 : animation du réservoir

Lorsque le niveau évolue entre une valeur initiale *niveauI* et une valeur finale *niveauF*, les nouveaux paramètres sont recalculés de la manière suivante :

hf := *niveauF* * *facteur;*

pf := *pi* + *(niveauF* - *niveauI)* * *facteur.*

facteur est le rapport de traduction.

V.2.1.2/ animation de la Vanne

La méthode de traduction de la vanne relie la variable *ouverture* à la couleur de l'élément graphique polygone. La variable *ouverture* peut prendre des valeurs situées entre 0 et 1. La couleur contient trois composantes : *rouge*, *bleu*, *vert*. Un exemple de contrainte d'animation consiste à moduler la composante *rouge* de la couleur par la valeur de la variable *ouverture*. Les valeurs d'autres composantes restent constantes.

Ainsi, pour les deux états extrêmes de la vanne (ouverte ou fermée), on obtient deux couleurs différentes, l'une avec une composante (*rouge*=0) et l'autre avec (rouge=1).

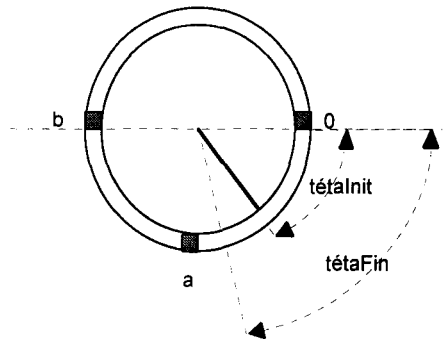
D'autres choix peuvent être adoptés à ce niveau, notamment pour moduler les composantes *bleu*, *vert* ou faire une combinaison des trois.

V.2.1.3/ animation de la bascule

L'animation de la bascule choisie est celle de la rotation d'un arc, dans le sens de l'aiguille d'une montre, en fonction de la valeur de la variable *poids*. Le composant graphique arc, utilisé à cet effet, est caractérisé par un angle initial *angleDébut* et une épaisseur fixe et suffisante pour simuler une aiguille (figure V.2). La traduction consiste à vérifier la relation entre la variable *poids* de la bascule et *angleDébut.* D'une manière générale, cette relation a la forme suivante :

angleDébut := *poids* * *facteur* .

facteur a pour effet de jouer sur le calibre de la mesure.



tétaInit : l'angle initial de l'arc

tétaFin : l'angle final de l'arc

Figure V.2 : animation de la bascule

V.2.1.4/ Animation du réservoirMalaxeur

L'objet *réservoirMalaxeur* contient trois variables à animer et trois types d'éléments graphiques dynamiques, ce qui nécessite autant de méthodes de traduction.

La traduction et l'animation du rectangle, correspondant au niveau, se font de la même manière que celles d'un réservoir simple. La traduction dans le cas du demi cercle correspondant au pivotement du réservoir, et des deux cercles, représentant l'état du moteur mélangeur, est basée sur le même principe que celui de la bascule.

V.2.1.5/ Animation de la partie commande

La partie commande est représentée par l'ensemble des missions vues dans le chapitre II. Chaque mission possède deux états distincts : *activé* ou *désactivé*.

Au niveau de l'animation, la présentation de la commande retenue est celle d'un Grafcet. Chaque mission correspond donc à une étape et est représentée principalement par un composant graphique rectangle dont la couleur sera modulée par l'état de la mission. Contrairement aux étapes, les transitions n'existent pas en tant qu'objets de simulation. Elles se manifestent de deux façons :

- par des objets liens dans le cas des transitions simples,
- par des contraintes dans l'objet mission *malaxeur* dans le cas des transitions complexes (synchronisation).

V.2.2/ Présentation du système malaxeur sur l'écran

La présentation du système est basée sur le principe de navigation étudié au chapitre IV. En effet, vu le grand nombre d'objets de simulation présents, il était judicieux de découper le système en deux parties : un synoptique pour la partie commande et un autre pour la partie opérative.

On obtient ainsi une hiérarchie à deux niveaux. Au premier niveau, l'opérateur dispose d'une vue générale comprenant deux objets de simulation composés correspondants aux deux parties du système (Figure V.3). Au deuxième niveau, l'utilisateur d'une vue détaillée, soit de la partie opérative (Figure V.4), soit de la partie commande (Figure V.5).

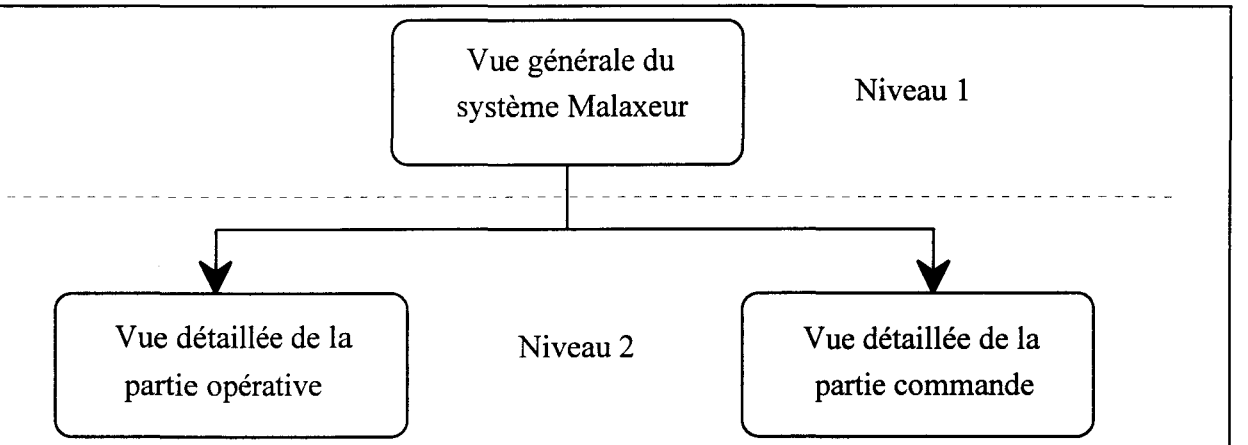


Figure V.3 : hiérarchie du système Malaxeur

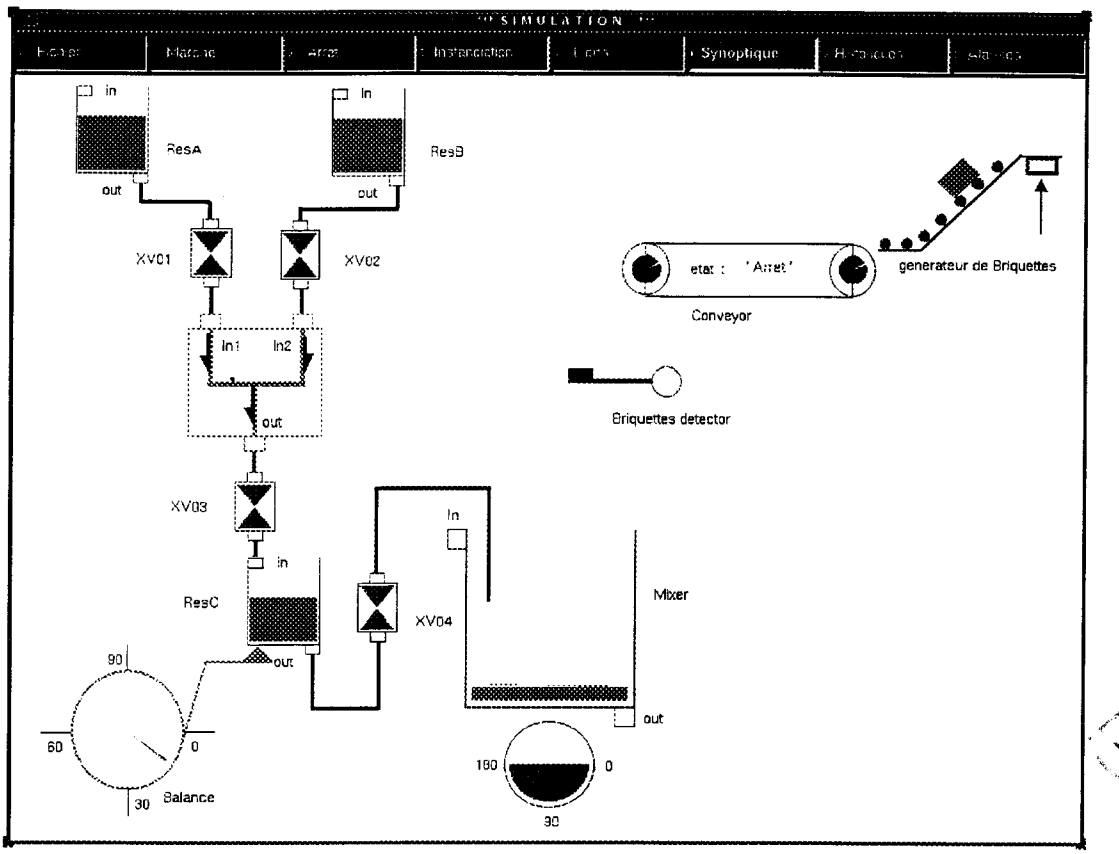


Figure V.4 : partie opérative du système malaxeur

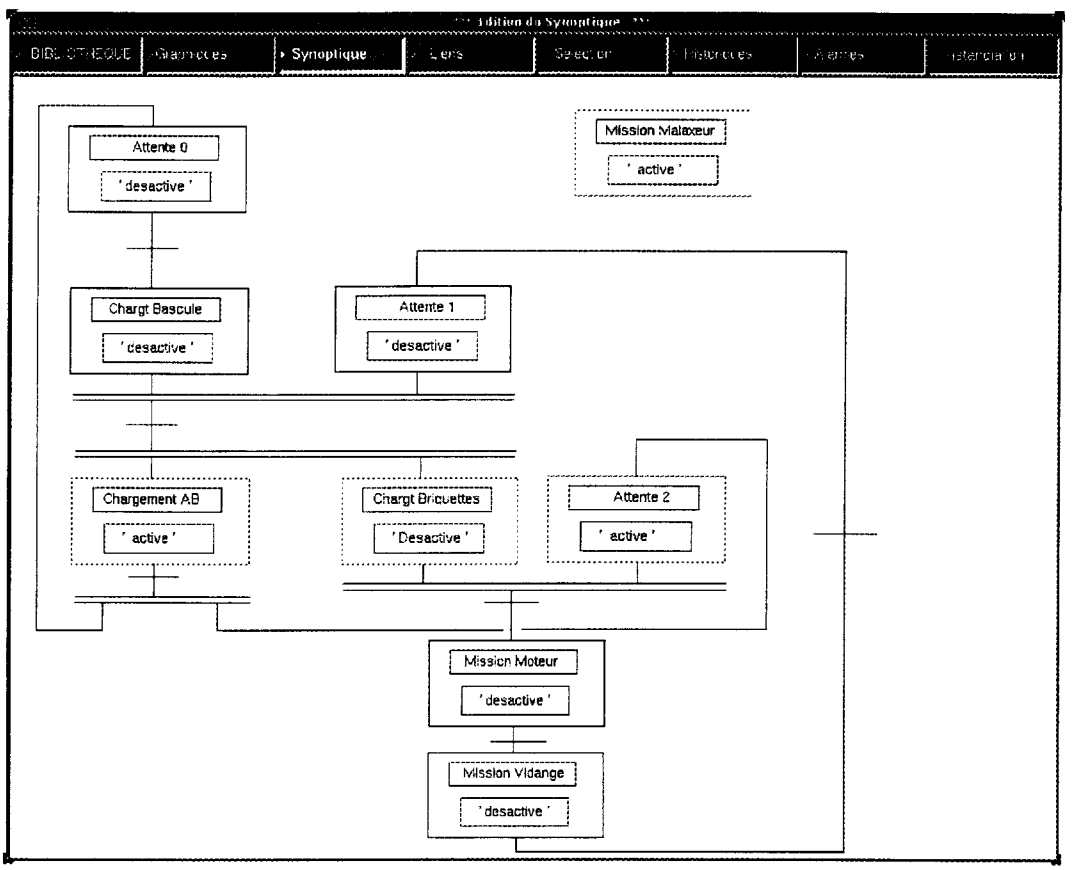


Figure V.5 : partie commande du système malaxeur

V.3/ ETUDE D'UN SYSTEME D'IRRIGATION

Cette application est issue d'un accord programme entre l'I.N.E.S d'électronique de Setif et l'U.S.T.L de Lille. Ce programme a pour objet la conception et la réalisation d'un prototype de système d'irrigation, entièrement automatique, adapté aux hautes plaines Algériennes.

V.3.1/ Vue générale d'un système d'irrigation automatique

La tendance actuelle est orientée vers la distribution géographique et fonctionnelle du système de contrôle-commande du système d'irrigation, entre différents modules intelligents et autonomes. Chaque module, disposant d'un matériel et d'un logiciel spécifiques, réalise des fonctions en réponse à des objectifs transmis par des modules d'un niveau supérieur.

Ce type de structure est parfaitement adapté au pilotage automatique d'un processus d'irrigation. Quatre types de stations intelligentes et autonomes seraient réparties sur le site et chargées de missions particulières et complémentaires :

SPA : la station de pompage pour la régulation de la pression d'eau et la commande des pompes (Figure V.10),

SAA : les stations parcellaires d'arrosage pour la saisie de mesures sur le site et la commande des électrovannes d'arrosage (Figure V.11),

SMA : la station de météo automatique pour l'acquisition des données bioclimatiques (Figure V.11).

SPC : la station principale pour la conduite, la surveillance et le dialogue opérateur (Figure V.12),

V.3.2/ Les matériels utilisés dans un système d'irrigation

Quel que soit la méthode d'irrigation utilisée, dans tout réseau d'irrigation sous pression (réseau où l'adduction d'eau se fait par conduites et non par canaux à l'air libre), on trouve généralement les éléments matériels suivants : pompes, vannes, conduites, réservoirs, capteurs et actionneurs.

Dans ce qui suit, nous allons présenter la définition et la modélisation des éléments nécessaires pour la construction du système d'irrigation. Il faut souligner que les modèles que nous avons adoptés sont des modèles simplifiés correspondant à un fonctionnement normal des objets physiques. Une modélisation plus réaliste du système, qui sort un peu du cadre de ce mémoire, est envisagée dans la continuité des travaux de recherche du Centre d'Automatique de Lille.

V.3.2.1/ Les vannes

On en trouve, en particulier, à chaque borne d'irrigation. On distingue trois familles de vannes d'irrigation suivant le type de leurs actionneurs : les vannes manuelles, les vannes hydrauliques et les électrovannes.

Quel que soit son type, la vanne est modélisée par les trois variables de fonctionnement suivantes : *ouverture*, *section* et *débit*. Son animation est identique à celle vue dans l'application du malaxeur.

V.3.2.2/ Les pompes

Elles équipent les stations de pompage des réseaux d'irrigation sous pression. C'est elles qui assurent le débit d'eau nécessaire dans les canalisations. Comme pour les vannes, les pompes peuvent être actionnées par différents types de moteurs : à énergie fossile, électriques ou même solaires. L'assemblage moteur plus pompe constitue un groupe motopompe. On peut distinguer :

♦ Les motopompes à vitesse fixe dont le débit ne peut varier que dans une plage relativement étroite, ce qui nécessite l'utilisation d'un réservoir tampon chargé d'absorber les excédents de débit.

♦ Les motopompes à vitesse variable, chers mais d'un fonctionnement souple et économe en énergie, la plage de débit couverte est beaucoup plus vaste et le rôle du réservoir tampon bien réduit.

A vrai dire, dans le cadre de la simulation, le problème de choix ne se pose pas. En effet, on peut envisager l'utilisation des motopompes à vitesse variable dont les paramètres de fonctionnement sont : *marche*, *vitesse*, *débit*, *niveauSource* et *section*.

Lorsque l'objet de simulation *pompe* est mis en marche (*marche* = true), il vérifie les deux contraintes suivantes :

♦ si le niveau du réservoir source est suffisant (*niveauSource* ~= 0),

♦ si la section de sortie n'est pas fermée (*section* ~= 0),

Dans le cas de satisfaction de ces contraintes, il affecte à la variable *débit* une valeur proportionnelle à la vitesse du moteur. L'animation peut se réduire au simple changement de couleur d'un composant graphique en fonction de la variable *marche*.

V.3.2.3/ Les conduites

Les conduites peuvent être divisées en deux catégories : conduites primaires et secondaires. On désigne par conduites primaires ou principales, les canalisations allant de la retenue d'eau jusqu'aux bornes d'irrigation, en passant par la station de pompage. Les conduites secondaires assurent l'alimentation en eau des arroseurs à partir des bornes d'irrigation.

La modélisation des canalisations n'est indispensable que lorsqu'on se trouve dans l'un des deux cas suivants :

1) une canalisation complexe ayant plus d'une entrée ou d'une sortie, dans ce cas, son comportement sert à gérer les débits de sorties en fonction des débits d'entrées et des différentes sections des extrémités. En effet, dans le cas d'une canalisation simple, il suffit de la remplacer par deux simples objets liens permettant de partager les sections et les débits.

Prenons l'exemple d'une canalisation ayant une entrée et deux sorties. Les variables caractéristiques sont : *débitE* (le débit d'entrée), *débitS1* (le débit de la

première sortie), *débitS2* (le débit de la deuxième sortie), *sectionE* (la section d'entrée), *sectionS1* (la section de la première sortie) et *sectionS2* (la section de la deuxième sortie). Les contraintes à vérifier sont :

- si les sections des sorties sont fermées (*sectionS1* = *sectionS2* = 0), alors fermer la section d'entrée (*sectionE* := 0),
- si la sortie 1 est fermée et la sortie 2 est ouverte, alors *débitS1* := 0 et *débitS2* := *débitE*,
- si la sortie 1 est ouverte et la sortie 2 est fermée, alors *débitS1* := *débitE* et *débitS2* := 0,
- si les deux sections sont ouvertes, alors affecter les deux débits de sorties : *débitS1* := *débitE* / 2 et *débitS2* := *débitE* / 2. Cette répartition des débits est valable pour des sections de sorties identiques.

2) Le deuxième cas où la modélisation d'une canalisation est indispensable apparait lors de l'introduction des fuites. Dans ce cas, la fuite est considérée comme une sortie supplémentaire qui sera prise en compte dans le calcul des débits des sorties.

L'animation des canalisations n'est pas prévue dans les méthodes de traduction et a été remplacée par des capteurs de débits placés en entrée et en sortie. La description de ces capteurs sera détaillée dans les paragraphes suivants.

V.3.2.4/ Les arroseurs

Le choix des arroseurs est fonction des considérations techniques suivantes :

- le débit et la pression d'arrosage (à calculer suivant le site à irriguer) : chaque arroseur doit supporter la pression d'eau en tête de rampe,
- l'écartement et la pluviométrie souhaités : chaque type d'arroseur assure une portée (fixant sa position par rapport aux autres) et une pluviométrie précises si la pression qui lui est appliquée est conforme,
- les types de sol et de culture : suivant les cultures, les arroseurs doivent être placés à une certaine hauteur sur des cannes.

La modélisation d'un arroseur est ramenée à celle d'une conduite simple qui termine sur une vanne dont l'ouverture est conditionnée par une pression de service minimale. Les paramètres de fonctionnement sont : *débit*, *section*, *pressionService*.

L'animation dans le cas de l'arroseur se fait par la rotation d'une hélice (2 arcs) et le changement de couleur de quatre cercles de manière séquentielle pour représenter le jet d'eau sur la surface irriguée (figure V.6).

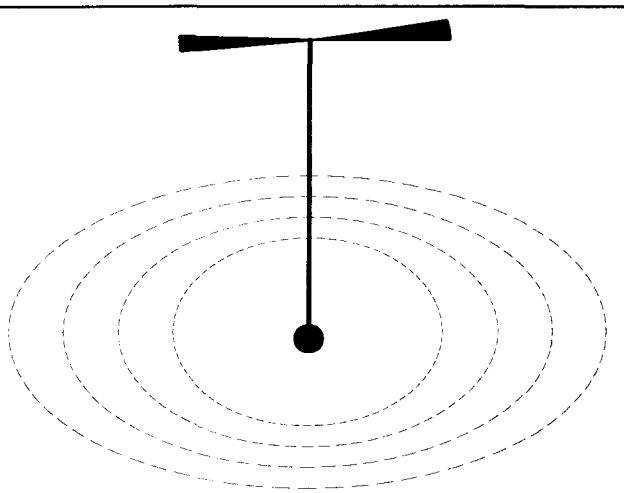


Figure V.6 : présentation d'un arroseur

V.3.2.5/ Les réservoirs

Deux types de réservoirs sont considérés :

- Le réservoir source représente la retenue d'eau et alimente la station de pompage.
- Le réservoir tampon est un réservoir fermé sous pression. Il se situe entre les pompes et les bornes d'irrigation et assure un débit constant en sortie. Sa particularité, par rapport au réservoir source, est la définition d'une variable supplémentaire (*débitService*) et l'introduction d'une régulation élémentaire de bas niveau. Cette régulation permet d'agir sur la pompe en la désactivant lorsque la pression interne du réservoir dépasse un certain seuil correspondant

à son remplissage. L'animation des réservoirs a été étudiée dans le cas du système malaxeur.

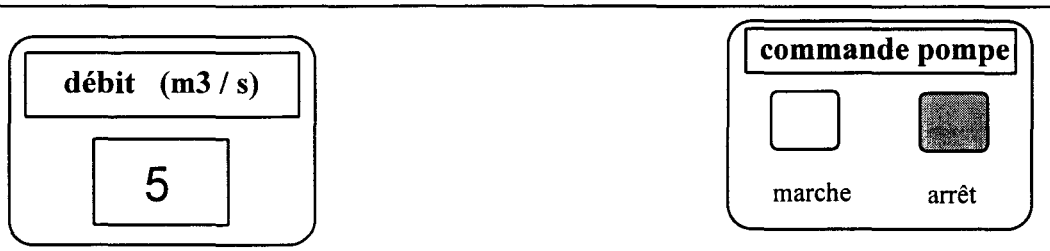
V.3.2.6/ Les capteurs et les actionneurs

Pour assurer l'alimentation en données des équipements de commande, un certain nombre de mesures doivent être effectuées, à l'aide de capteurs, sur le site à irriguer. C'est à ces mêmes équipements (de commande) de les prendre en charge et d'activer les actionneurs nécessaires qui leur sont connectés.

Les capteurs

Tous les capteurs simples sont modélisés de la même manière par un objet de simulation ayant une variable d'instance *valeur*. Cette variable sera liée, par l'intermédiaire d'un lien, au paramètre à mesurer. Un débitmètre, par exemple, est obtenu en utilisant un capteur simple dont la variable *valeur* est liée à la variable *débit* de la canalisation sur laquelle est effectuée la mesure.

La présentation du capteur contient essentiellement une information sur la valeur sous forme alphanumérique (composant graphique texte) et éventuellement des commentaires supplémentaires indiquant, par exemple, le type du capteur (figure V.7).



a) présentation du capteur

b) présentation de l'actionneur

Figure V.7 : présentations du capteur et de l'actionneur

De plus, des comportements spécifiques peuvent être intégrés. On parle dans ce cas de capteurs sophistiqués. En effet, on peut imaginer un capteur permettant d'indiquer un dépassement d'un seuil maximum ou d'un seuil minimum. Au niveau synoptique, ce comportement peut se traduire sous forme d'un changement de couleur

d'un composant graphique quelconque. Le choix des couleurs est très important et impose au concepteur d'obéir aux règles ergonomiques des synoptiques.

Les actionneurs

Dans l'environnement de simulation, les actionneurs se situent entre les équipements à commander et les organes de commande. Ces organes peuvent être matériels (missions) ou immatériels (utilisateurs).

Dans le premier cas, les actionneurs servent plutôt d'adaptateurs entre les équipements qui sont, par nature, différents dans leur fonctionnement et dans leur modélisation. La modélisation de ces actionneurs est basée sur un principe unique. La différence entre un actionneur et un autre dépend directement de la modélisation des équipements sur lesquels agissent ces actionneurs. Mais, d'une manière générale, un objet de simulation actionneur possède une ou plusieurs variables d'entrée, liées aux variables de l'objet commande, et une ou plusieurs variables de sortie, liées aux paramètres de mise en marche ou en arrêt de l'équipement commandé. Les contraintes à définir servent à maintenir une cohérence entre les variables d'entrée et celles de sortie.

Dans le cas ou l'actionneur est piloté directement par l'utilisateur, la construction de la présentation est basée sur des principes ergonomiques permettant un pilotage manuel confortable de l'utilisateur. Prenons l'exemple d'un actionneur du type marche/arrêt pour la commande d'une pompe (figure V.7). La présentation est caractérisée par deux composants graphiques simples dont les identificateurs sont respectivement *'marche'* et *'arrêt'*. L'utilisateur pointe par la souris l'un des composants pour exprimer son besoin d'activer ou de désactiver la pompe. En fonction du bouton sélectionné, l'objet de simulation actionneur lance l'une ou l'autre des deux actions correspondant aux états *marche* et *arrêt*. Ces actions sont définies lors de la phase de modélisation.

V.3.3/ Présentation du système d'irrigation sur écran

L'organisation des objets du système d'irrigation sous forme hiérarchique est basée sur le même principe que celui du système malaxeur. On se trouve alors avec une hiérarchie à trois niveaux cette fois ci. Sa structure est montrée en figure V.8. Le niveau 1 est toujours réservé à une vue d'ensemble du système. Cette vue regroupe l'ensemble des stations sous forme iconique non détaillée.

Dans le niveau 2, l'opérateur a une vue éclatée de l'une des stations. Enfin, en ce qui concerne la station d'arrosage particulièrement et pour plus d'ergonomie dans la présentation des objets, on a prévu un troisième niveau pour représenter le détail des bornes d'irrigation. En effet, au niveau 2 de la station d'arrosage, l'utilisateur dispose d'une vue générale des bornes d'irrigation (type de culture, quantité d'eau consommée, date de semence, etc.). De plus, il peut sélectionner une des bornes et passer du deuxième au troisième niveau afin d'avoir plus d'informations sur telle ou telle culture

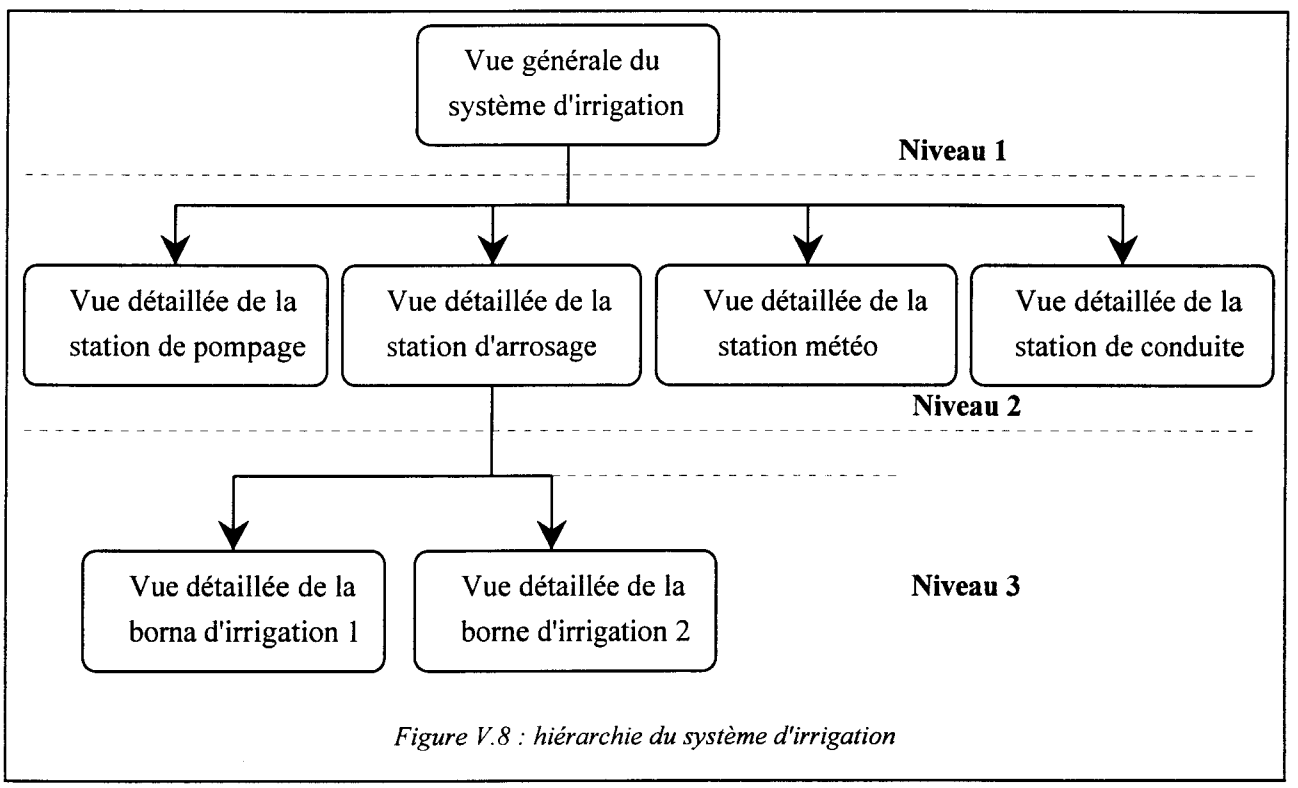


Figure V.8 : hiérarchie du système d'irrigation

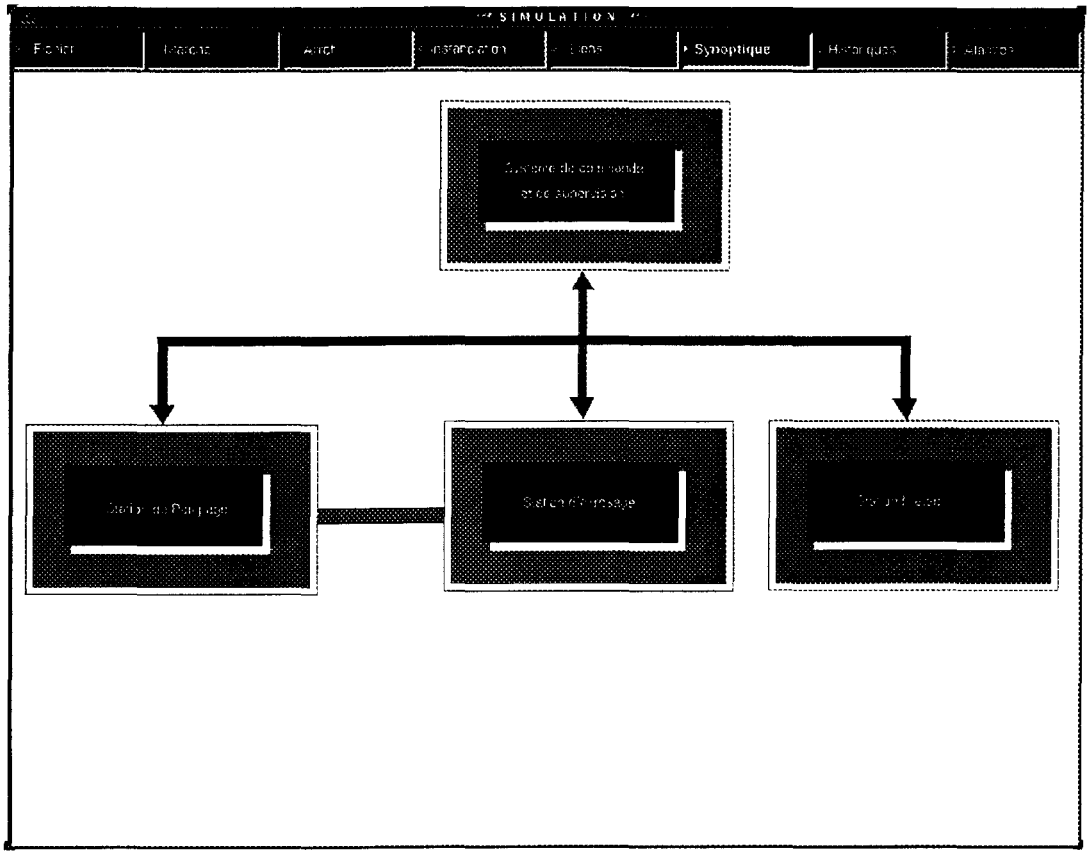


Figure V.9 : vue générale du système d'irrigation (niveau 1)

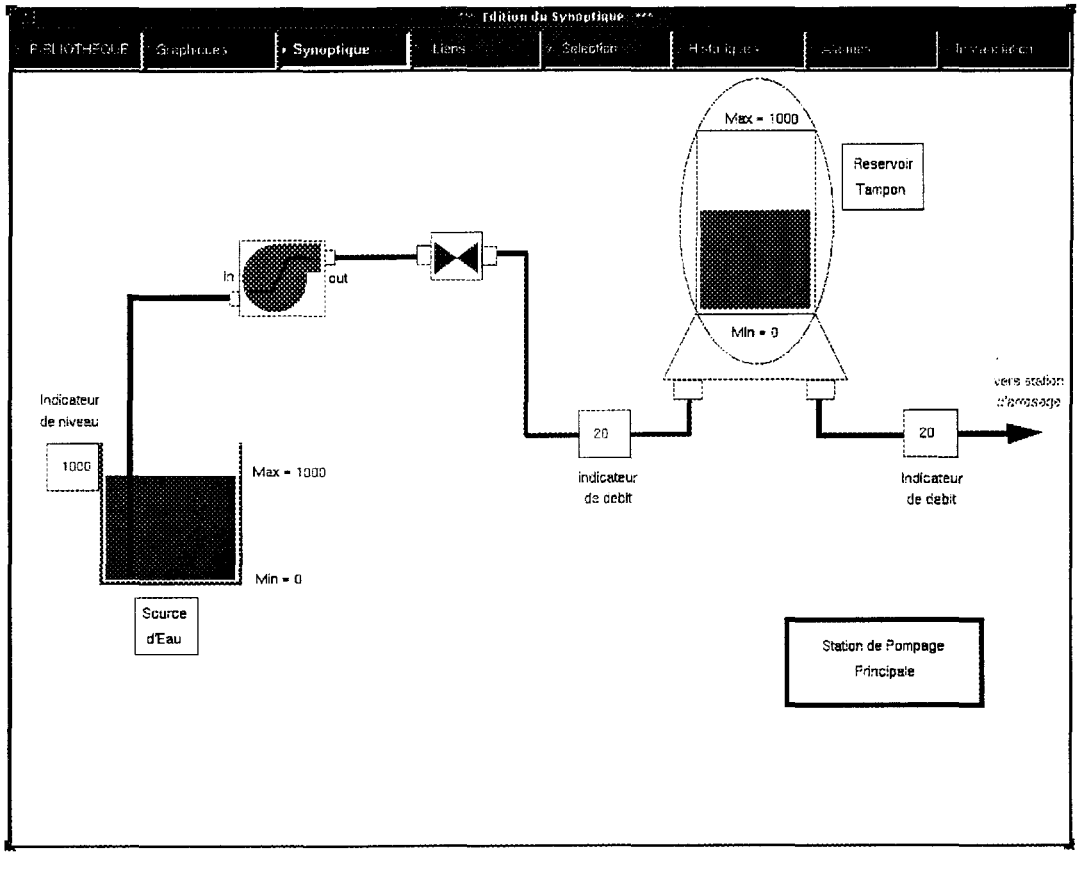


Figure V.10 : vue détaillée de la station de pompage (niveau 2)

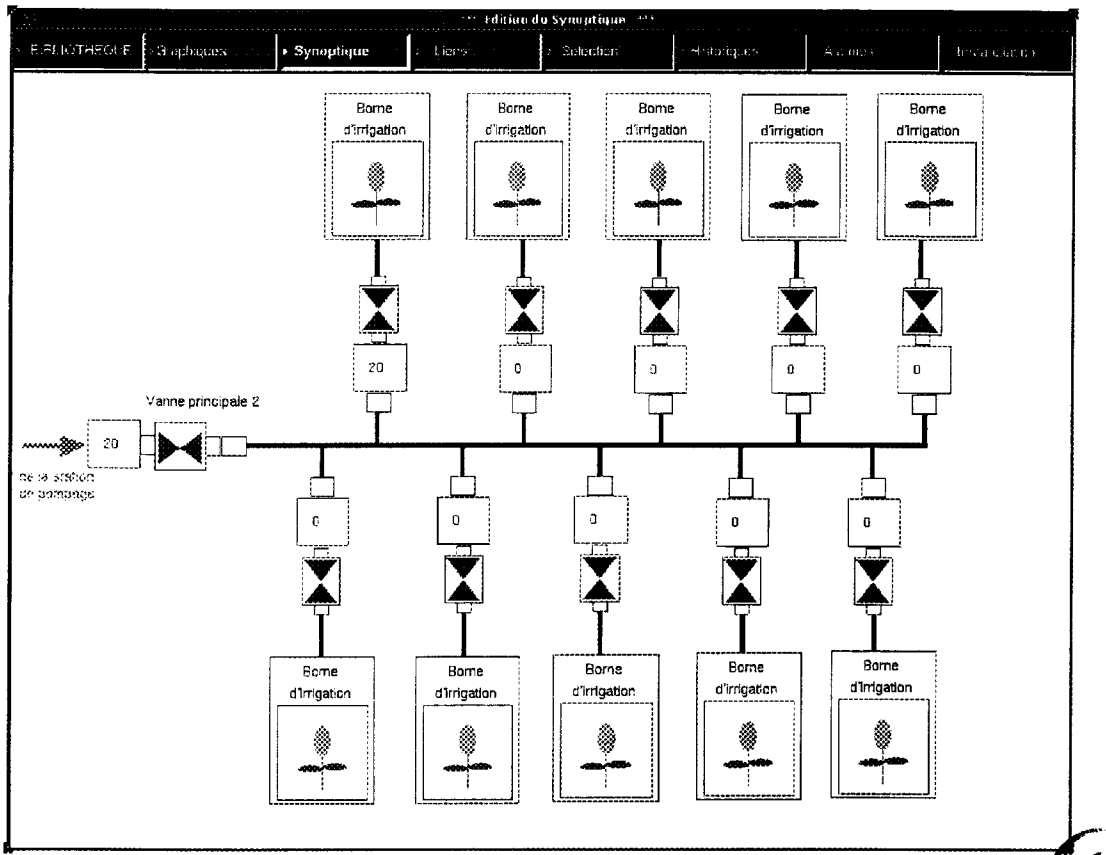


Figure V.11 : vue détaillée de la station d'arrosage (niveau 2)

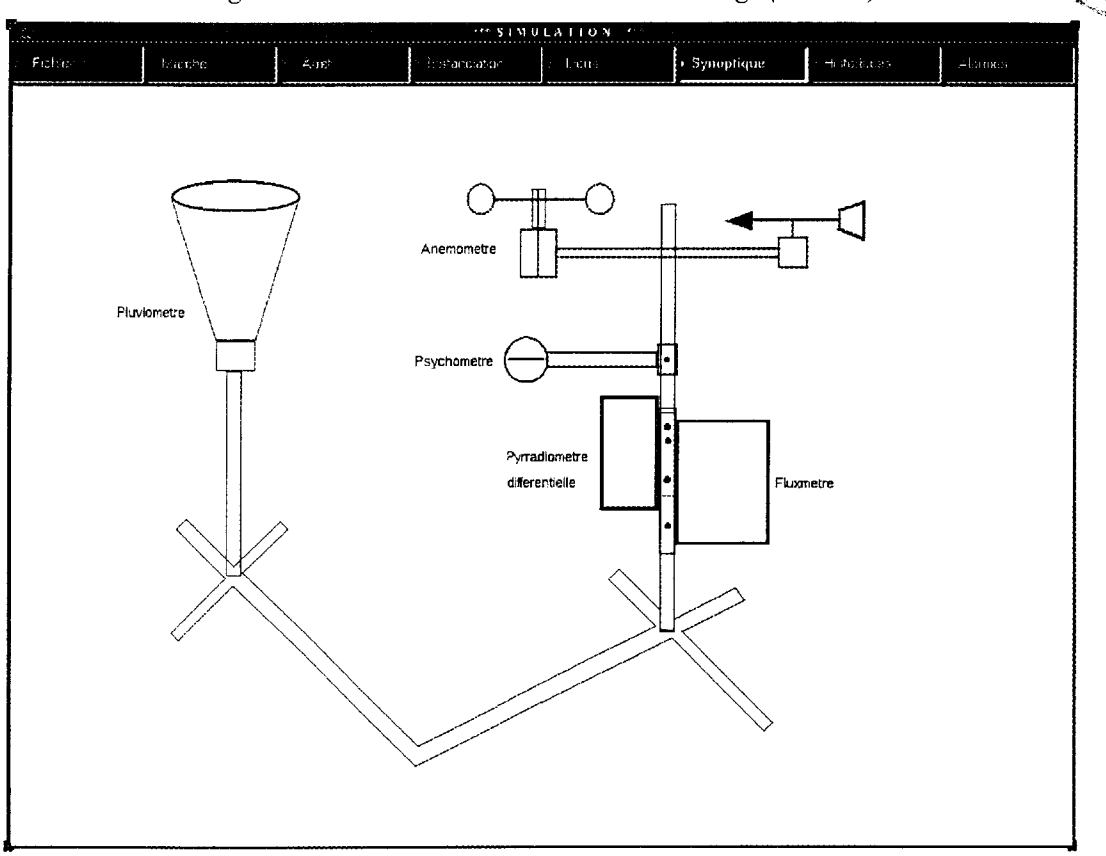


Figure V.12 : vue détaillée de la station météo (niveau 2)

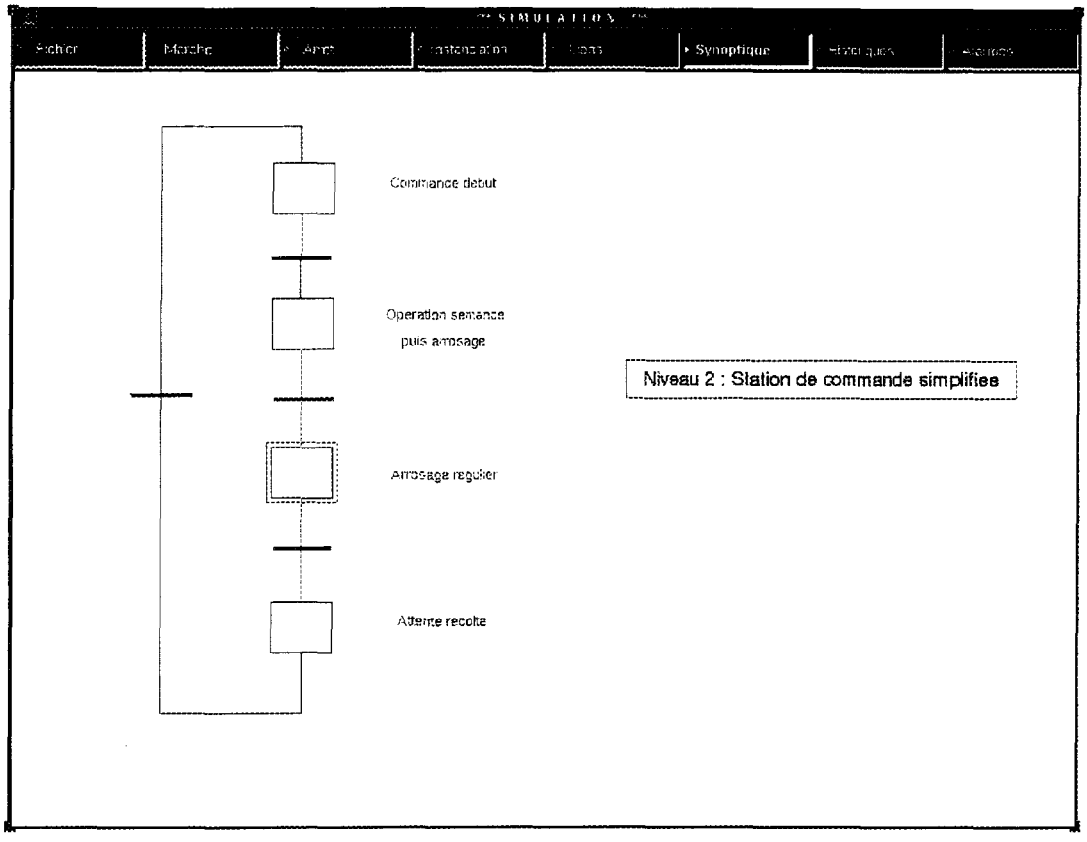


Figure V.12 : vue détaillée de la station de conduite (niveau 2)

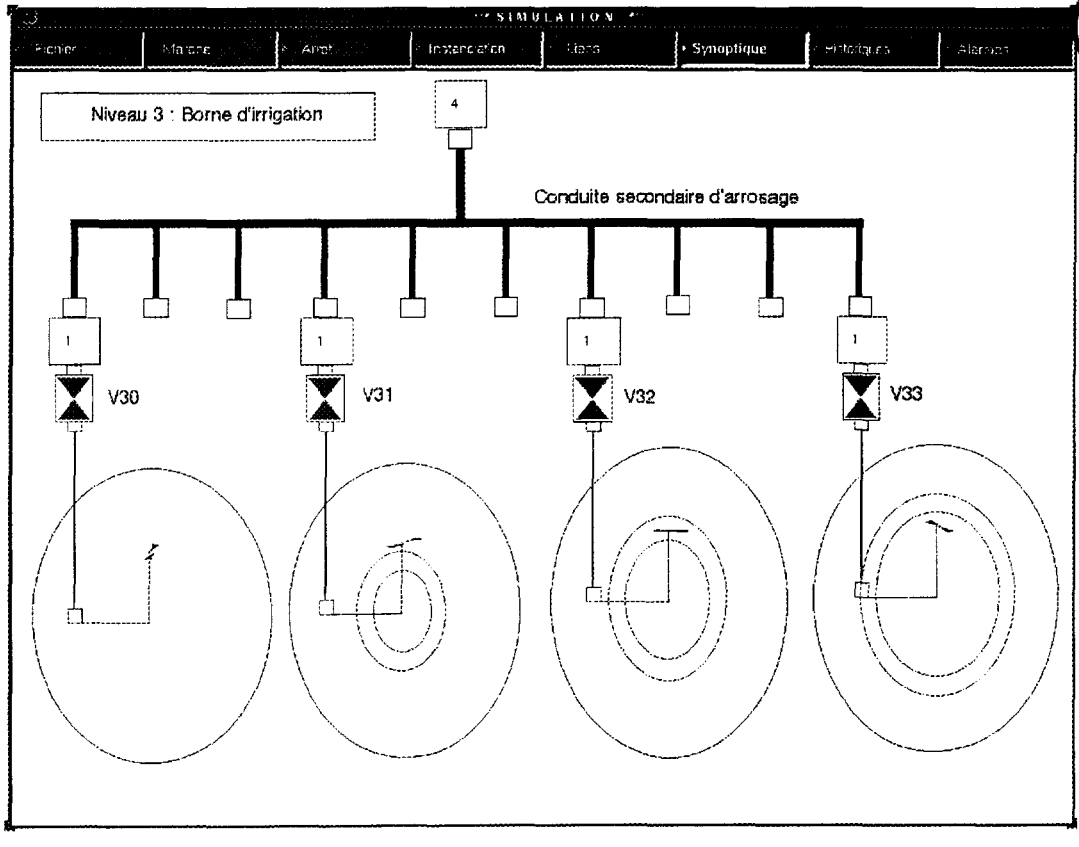


Figure V.12 : vue détaillée de la borne d'irrigation (niveau 3)

V.4/ LES SYSTEMES DE TRANSPORT AUTOMATISES

Le système étudié, inspiré du métro automatique de Lille (V.A.L.), est composé de plusieurs Rames. Chaque rame a une certaine capacité (nombre de voyageurs) et certaines performances cinématique.

Les rames circulent sur un Réseau composé de Voies sur lesquelles sont disposées des stations. La connexion entre ces voies est effectuée à l'aide d'Aiguillages.

A un instant donné, les paramètres cinématique d'une rame (position, vitesse, vitesse de consigne, accélération, etc.) ont chacun une certaine valeur et la rame contient un certain nombre de voyageurs.

V.4.1/ Décomposition en objets de simulation

La description du système étudié nous conduit à identifier les objets de simulation suivants : Rame, Voie, Station, Aiguillage. Il convient de construire pour chaque objet un prototype.

V.4.1.1/ Prototype de la Rame

L'objet Rame définit les variables suivantes :

- *position* : la position de la Rame dans le Réseau,
- *positionCible* : la position destination de la Rame,
- *vitesse* : la vitesse de la Rame,
- *accélération* : l'accélération de la Rame,
- *élémentDessous* : l'élément sur lequel se trouve la Rame (Voie, Station, etc.),

Le comportement de la Rame consiste à se déplacer de la position initiale avec les paramètres vitesse et accélération. Ceci se traduit par la contrainte :

position := *position* + *vitesse***pas* + 1/2**accélération***pas***pas*;

" pas " représente le pas de simulation.

L'animation dans le cas de la Rame, est donnée par un déplacement de la présentation en fonction de l'évolution de sa position. La méthode de traduction se résume donc à mettre à jour la position de la présentation en question.

V.4.1.2/ Prototype Voie

Un objet Voie est un élément du réseau. Lors de sa modélisation, il convient de distinguer deux types de voies : les voies à sens unique et les voies à double sens. Les voies à double sens autorisent, comme leur nom l'indique, la circulation des rames dans les deux sens et font intervenir les problèmes classiques de gestion des ressources non partageables et l'utilisation des techniques temps réel que nous verrons dans les paragraphes suivants. Les variables associées à un objet Voie à sens unique sont :

- *positionE* : la position d'entrée à la Voie,
- *positionS* : la position de sortie de la Voie,
- positions : la liste des positions indiquant l'itinéraire de la rame à l'intérieur de la voie,
- *rameE* : la rame entrante,
- *rameS* : la rame sortante,
- *listeRames* : la liste des rames qui se trouve à un moment donné à l'intérieur de la Voie.

Un objet Voie peut être connecté à d'autres éléments du réseau par l'intermédiaire des variables d'entrée *rameE* et de sortie *rameS*.

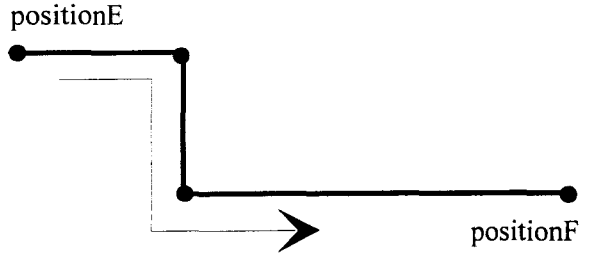


Figure V.15 : présentation de la Voie

A chaque fois qu'une rame entre dans la voie par la variable *rameE*, elle est dirigée vers la position d'entrée *positionE* et est rajoutée à la liste des rames *listeRames*.

Son comportement se résume dans sa capacité d'informer les rames sur leurs positions cibles. En effet, à chaque fois qu'une rame atteint sa position cible, elle demande à l'élément de voie qui la contient de lui renvoyer une nouvelle position cible. Lorsque la rame arrive à la position de sortie *positionS*, la rame est enlevée de la liste et quitte la voie par l'intermédiaire de la variable *rameS* pour aller sur un autre élément de voie et ainsi de suite. Ce type d'objet ne nécessite pas d'animation particulière et donc pas de méthodes de traduction.

V.4.1.3/ Prototype Station

L'objet de simulation Station est un élément du réseau qui contient deux voies parallèles (figure V.16), avec :

- *pE1* : la position d'entrée gauche de la station,
- *pE2* : la position d'entrée droite de la station,
- *pS1* : la position de sortie droite de la station,
- *pS2* : la position de sortie gauche de la station,
- *pA1* : la position d'arrêt de la première rame,
- *pA1* : la position d'arrêt de la deuxième rame,
- *rameE1* : la rame entrante par la gauche,
- *rameS1* : la rame sortante par la droite,
- *rameE2* : la rame entrante par la droite,
- *rameS2* : la rame sortante par la gauche,

La fonctionnalité de la station inclue celle de la voie, qui se charge d'informer la rame des positions cibles, mais en plus, de contrôler le temps d'arrêt de cette rame à la position *pA1* ou *pA2* pendant un laps de temps fixé avant de sortir par la variable *rameS1* ou *rameS2*.

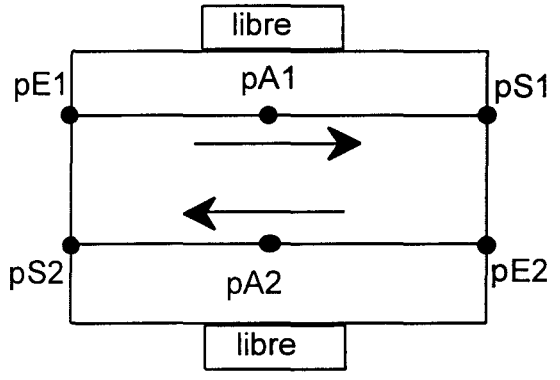


Figure V.16 : présentation de la station

V.4.1.4/ Prototype Aiguillage

L'objet aiguillage est l'élément du réseau qui permet aux rames de passer d'une voie à une autre avec un changement de sens de circulation. Les aiguillages sont situés dans chaque extrémité du réseau. Contrairement aux voies, un aiguillage ne peut supporter qu'une seule rame à la fois. Les variables d'instances dans ce cas sont :

- *pE* : la position d'entrée à l'aiguillage,
- *pS* : la position de sortie de l'aiguillage,
- *rameE* : la rame entrante,
- *rameS* : la rame sortante.

La présentation graphique d'un objet aiguillage est donnée par le schéma de la figure V.17. Les deux types de présentations correspondent aux cas ou l'aiguillage se trouve à gauche ou à droite du réseau.

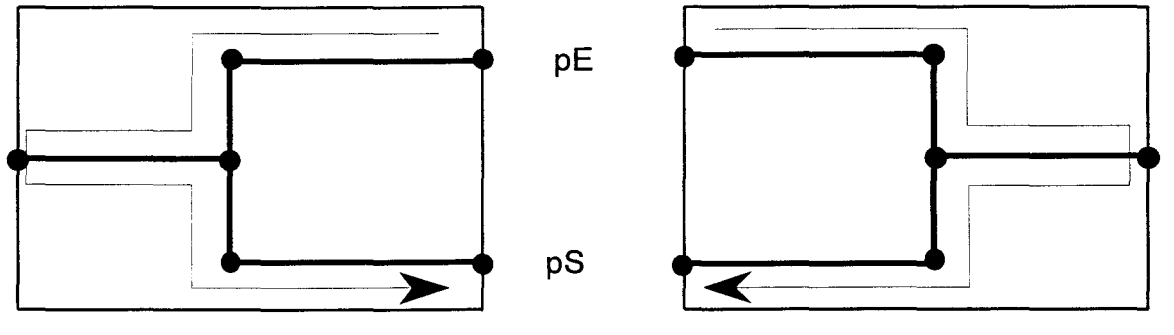


Figure V.17 : présentation d'un aiguillage

V.4.2/ Intégration du mécanisme anti-collision

Le problème de collision entre les rames a été résolu par l'introduction de la notion de canton dans les contraintes des voies. Une voie est en fait décomposée en plusieurs cantons. Chaque canton ne doit contenir qu'une seule rame à la fois. La répartition en cantons correspond en réalité à la liste des positions cibles qui représentent la trajectoire de la voie.

A chaque fois qu'une rame atteint une position cible et demande à la voie une nouvelle position, pour passer du canton actuel au canton suivant, la voie vérifie si cette nouvelle position cible appartient à l'une des rames de la liste. Dans ce cas, la voie installe un sémaphore de synchronisation dans les deux rames concernées (une rame bloquée et une rame bloquante). Ainsi, dés que la rame bloquante libère le canton, elle agit sur le sémaphore et autorise la rame bloquée à demander de nouveau la nouvelle position cible qui lui sera automatiquement accordée.

Ce problème de collision apparaît lorsqu'une rame quitte une voie pour rentrer dans un autre élément du réseau (voie, station ou aiguillage). C'est pourquoi un deuxième niveau de vérification est géré par la voie lorsqu'une rame atteint la position de sortie. Cette vérification permet de détecter un éventuel blocage dans l'élément du réseau en aval. Dans ce cas, la voie procède de la même manière et installe un sémaphore entre les deux rames concernées par la collision, c'est à dire entre la rame bloquée en sortie et la rame qui tente de sortir.

V.4.3/ Adaptation du modèle de simulation au mécanisme anti-collision

La structure de la rame a été sophistiquée pour avoir un comportement adapté au mécanisme anti-collision. Trois variables d'instances supplémentaires ont été rajoutées:

drapeau : un booléen qui autorise ou non l'objet de simulation à résoudre ses contraintes,

sémaphoreS : le sémaphore qui met la rame dans un état bloquant,

sémaphoreW : le sémaphore qui met la rame dans un état bloqué.

Lorsque la voie détecte deux rames en collision, elle crée un sémaphore de synchronisation et l'affecte à la fois au *sémaphoreS* de la rame bloquante et au *sémaphoreW* de la rame bloquée.

En conséquence, la rame bloquée lance un processus d'attente définit comme suit:

processusAttente

drapeau := false.

sémaphoreW wait.

drapeau := true

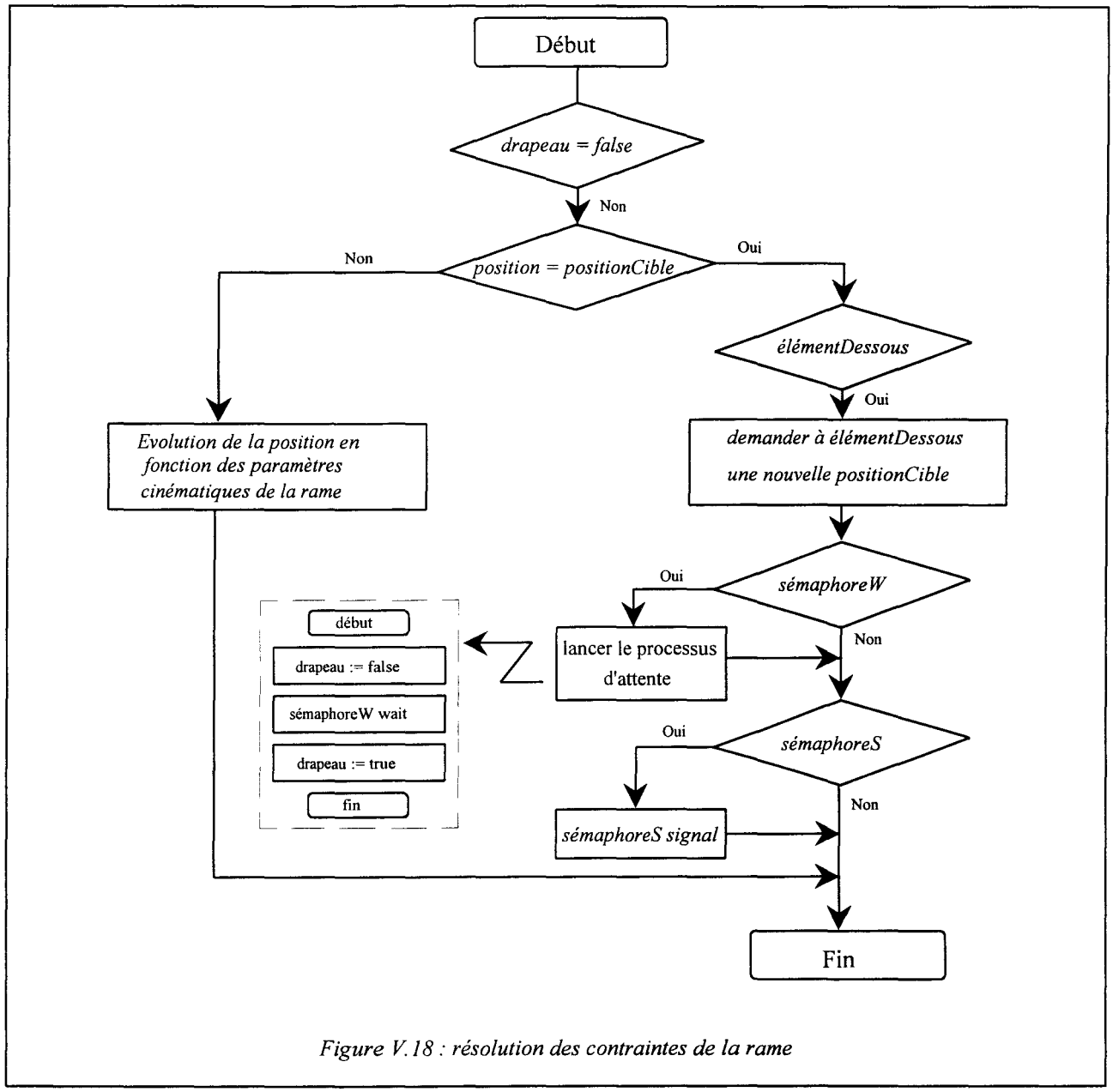


Figure V.18 : résolution des contraintes de la rame

Ce processus sera donc bloqué au niveau de la deuxième action tant que le *sémaphoreW* n'aura pas été signalé (réception du message *signal*). Le *drapeau* étant à l'état *false*, l'objet de simulation rame en question n'est plus autorisé à résoudre ses contraintes.

Par ailleurs, la rame bloquante évolue normalement jusqu'à la position cible. Et aussitôt qu'elle obtient une nouvelle position cible, elle envoie au sémaphore de synchronisation le message *signal*.

L'organigramme de la figure V.18 résume le comportement de la rame lors de la résolution des contraintes.

V.4.4/ Gestion des ressources non partageables

Tous les éléments du réseau qui n'accepte qu'une rame à la fois sont considérés comme des ressources non partageables. C'est le cas des stations, des aiguillages et des voies à double sens qu'on va décrire dans ce paragraphe. Dans ce cas, une variable caractéristique appelées *drapeau* sert d'indicateur de présence d'une rame et le blocage des rames désirant accéder à la ressource. A noter que dans le cas des stations, deux indicateurs sont nécessaires : *drapeau1* et *drapeau2* puisqu'elles autorisent l'accès simultané de deux rames à la fois.

Lorsqu'une rame accède à une ressource, l'indicateur est positionné (*occupé*) et ne sera libéré que si la rame quitte cette ressource. C'est sur ce principe qu'a été construit l'élément voie à double sens dont les variables d'instances sont les suivantes :

- *pE1* : la position d'entrée 1,
- *pE2* : la position d'entrée 2,
- *pS1* : la position de sortie 1,
- *pS2* : la position de sortie 2,
- *rameE1* : la rame entrante N°1,
- *rameE2* : la rame entrante N°2,
- *positions1* : l'itinéraire de la rame N°1,
- *positions2* : l'itinéraire de la rame N°2,
- *drapeau* : indicateur de présence de rame.

La figure V.19 montre comment on peut utiliser une voie à double sens " *voieD* " dans un circuit qui met en évidence le problème de section critique. Dans cet exemple, deux types de rames sont considérés : les rames qui viennent de *voieE1,* qui se dirigent vers *voieS1* en empruntant *voieD,* et les rames en provenance de *voieE2,* qui se dirigent vers *voieS2* en empruntant la même *voieD.*

A l'état initial, la *voieD* est libre. Les rames peuvent y accéder par l'intermédiaire des variables *rameE1* et/ou *rameE2.* Ces deux variables sont liées par des liens aux variables *rameS* des éléments du réseau situés en amont (figure V.19). Elles pointent sur les objets rames qui tentent d'accéder à la *voieD* et servent à informer cette dernière sur son état, lors de la résolution de ses contraintes, et lui permettre ainsi de gérer cette section critique.

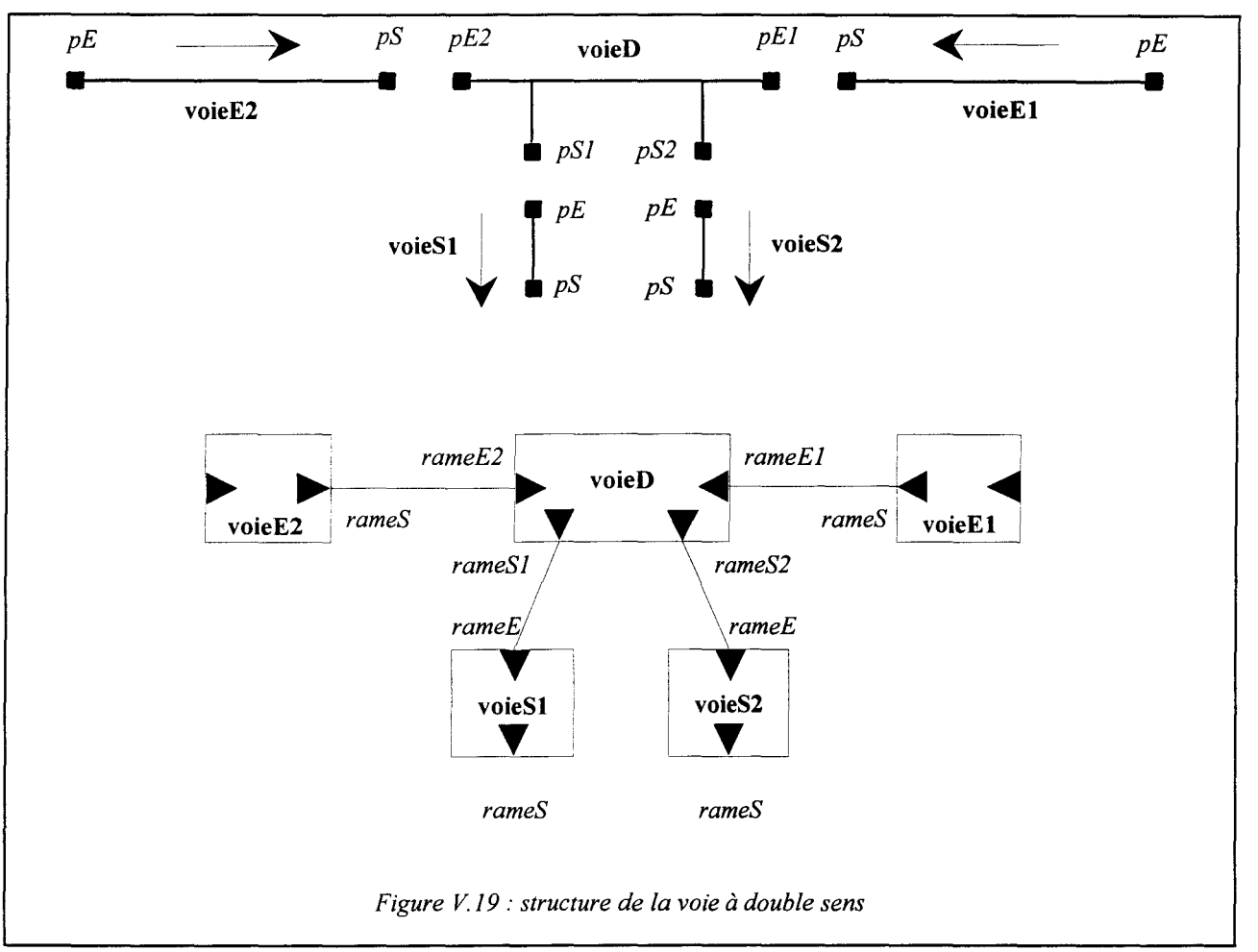


Figure V.19 : structure de la voie à double sens

La première rame qui rentre provoque la levée du *drapeau* et le blocage des rames au niveau des deux entrées. Lorsque cette rame quitte la voie par l'une des

variables *rameS1* ou *rameS2* et se trouve dans un état non bloqué, le drapeau est baissé et les rames sont autorisées à rentrer.

Nous avons donc défini deux types de contraintes à vérifier. La première permet l'introduction d'une rame à l'intérieur de la voie et la deuxième se charge de son évacuation lorsqu'elle atteint la position de sortie *pS1* ou *pS2*. La figure V.20 illustre ces deux contraintes sous forme d'organigrammes.

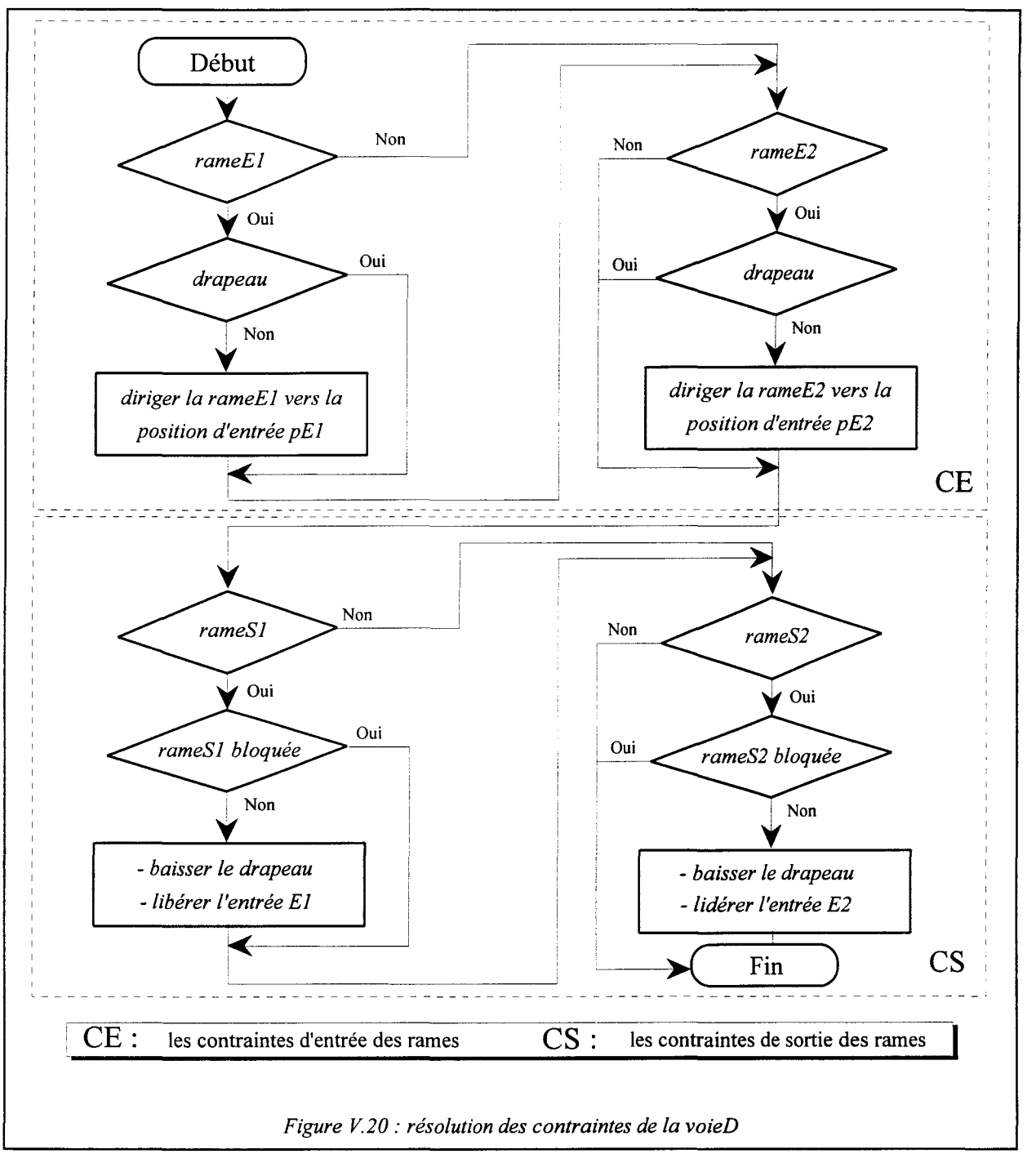


Figure V.20 : résolution des contraintes de la voieD

V.4.5/ exemples d'applications

Dans le cadre de la simulation des systèmes de transports automatisés, nous avons étudié deux applications différentes. La première reprend d'une certaine manière le métro automatique de Lille en utilisant des voies à sens unique, des stations et des éléments d'aiguillage aux extrémités.

La figure V.21 représente l'architecture informatique de l'application structurée en deux schémas séparés. Dans le premier schéma, la trajectoire des rames est représentée par les différentes positions d'entrée et de sortie de chacun des éléments du réseau alors que le deuxième schéma montre comment sont liés ces éléments par des objets liens.

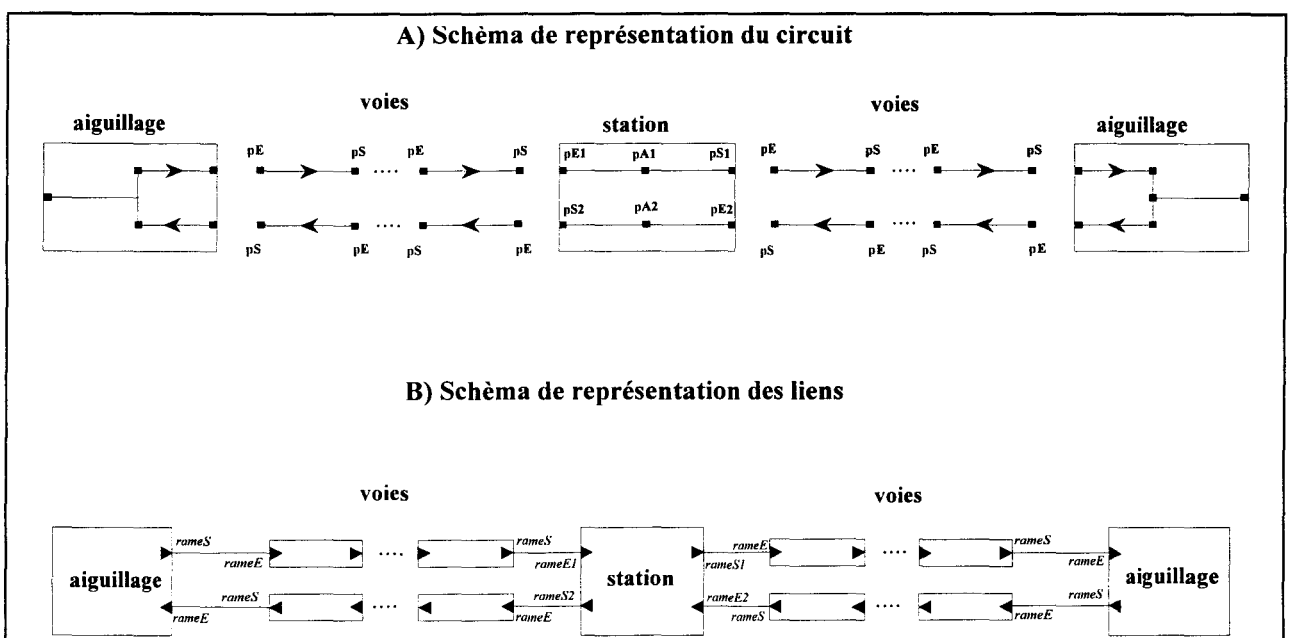
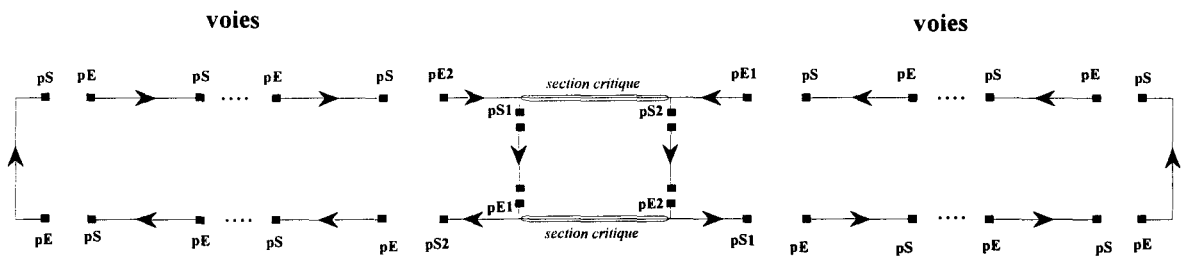


Figure V.21 : structure informatique de l'exemple1

Le deuxième exemple décrit un autre type de circuit utilisant des éléments voies à double sens de manière à avoir deux sections critiques et deux types de rames circulant chacun avec un sens pré-défini. La structure informatique correspondante est montrée en figure V.22.

Ces deux exemples montrent effectivement l'efficacité du mécanisme de l'anti-collision et des possibilités d'intégration des contraintes dans la modélisation des systèmes de transports automatisés.

A) Schèma de représentation du circuit



B) Schèma de représentation des liens

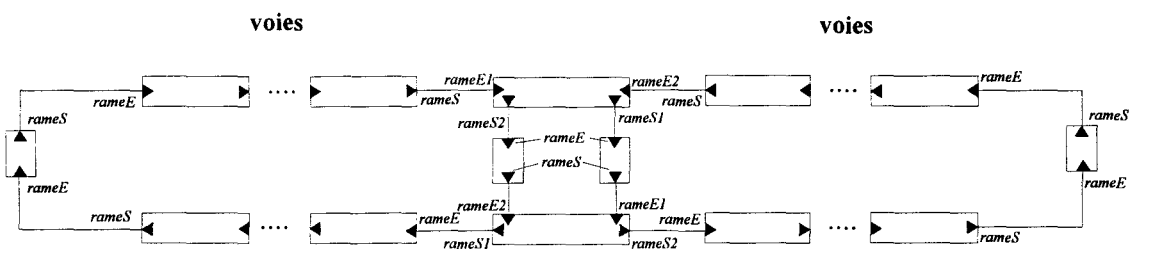


Figure V.22 : structure informatique de l'exemple1

V.4.6/ Présentation des circuits sur écran

Les figures V.23 et V.24 montrent les synoptiques choisis pour les circuits étudiés précédemment. Les rames sont représentées par des carrés pleins.

Il est important de signaler que le lecteur de ce mémoire appréciera mieux la présentation et la démonstration sur machine. En effet, il nous est difficile, si ce n'est impossible, de reproduire certains paramètres ergonomiques comme l'animation.

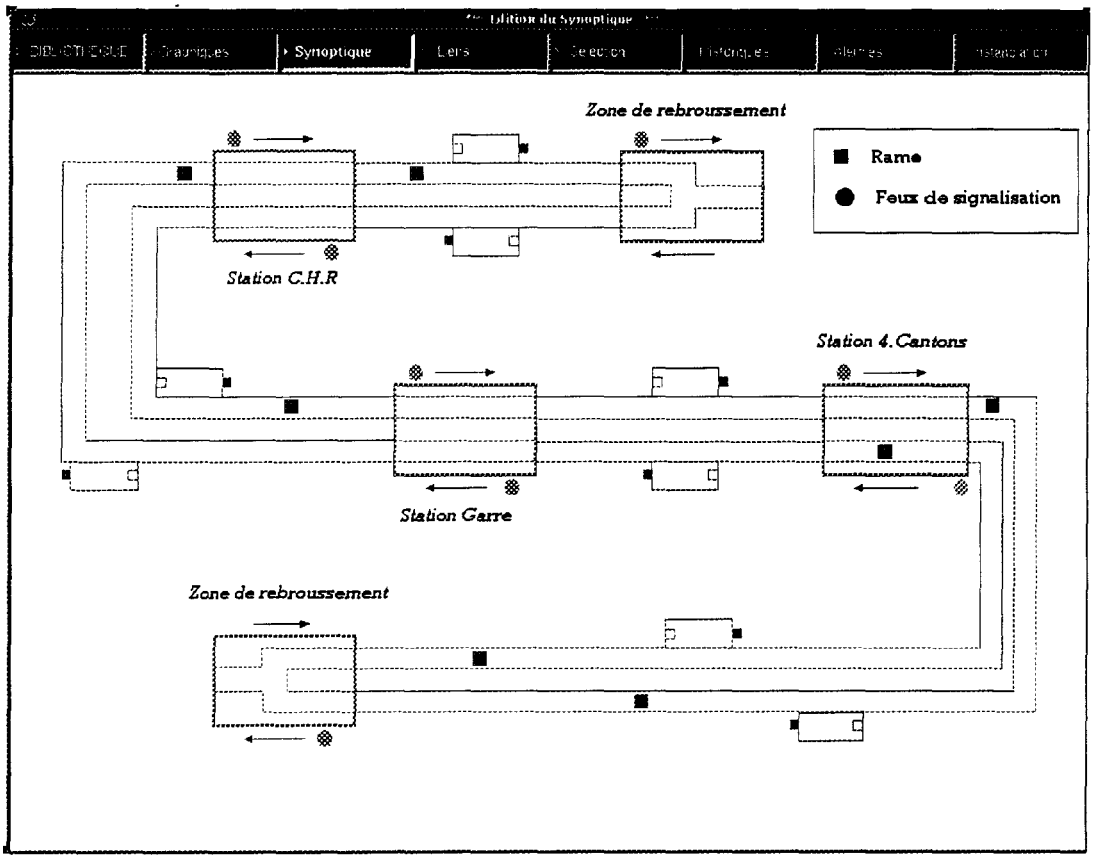


Figure V.23 : prototype de simulation du métro de Lille (VAL)

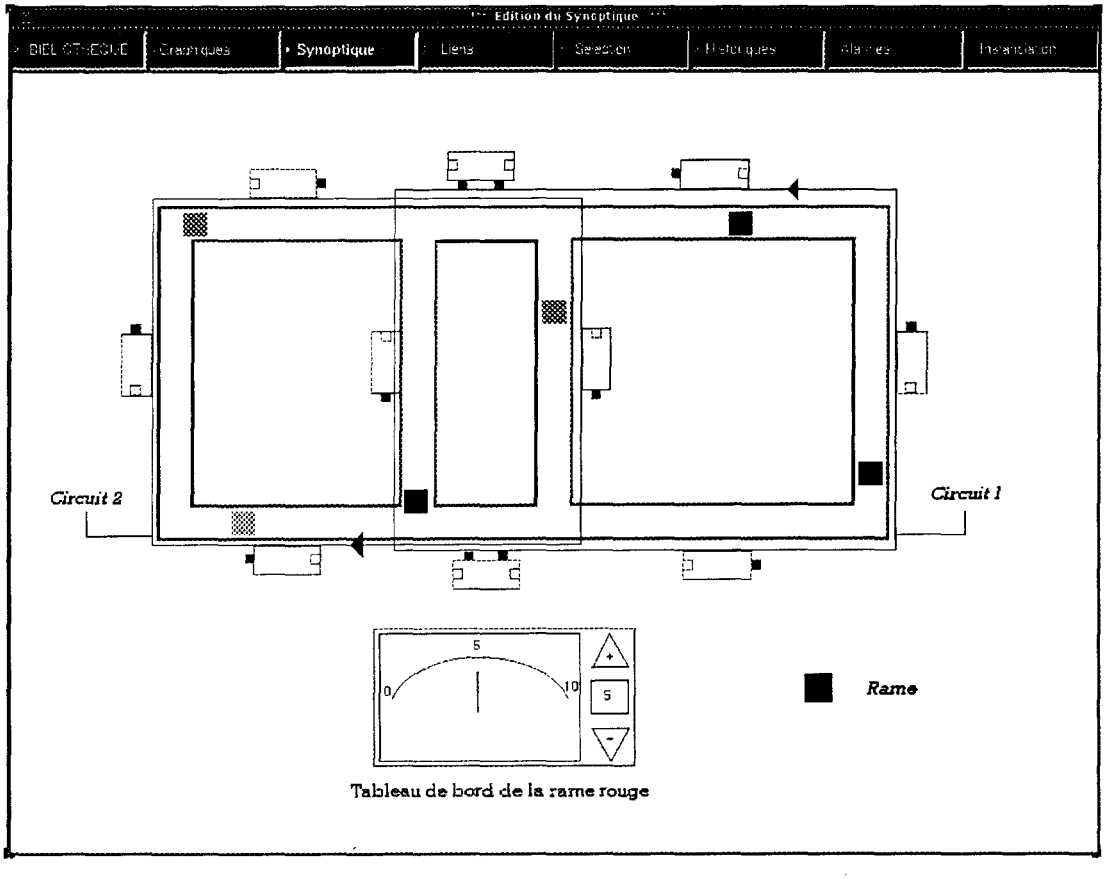


Figure V.24 : simulation d'un circuit avec deux sections critiques

V.5/ CONCLUSION

Ce chapitre illustre l'utilisation du modèle P.Os.T avec la description concrète de quatre types de systèmes interactifs. L'intérêt de cette présentation est de montrer l'adaptation du modèle P.Os.T à la diversité des applications et des utilisateurs.

Vis-à-vis des applications, les objets de simulation écrits à base de contraintes traduisent de manière naturelle le comportement des objets physiques qu'ils représentent. En effet, quel que soit le type de l'application, le principe de modélisation reste le même. L'utilisation des objets liens leur permet de résoudre les problèmes les plus courants rencontrés dans les applications temps réel (synchronisation, exclusion mutuelle, etc.).

Vis-à-vis de l'utilisateur, la séparation entre l'application et l'interface permet d'associer à un même modèle de simulation de synoptiques différents. Ainsi, lors de la construction de l'interface, l'utilisateur exploite ses connaissances antérieures pour s'exprimer graphiquement, en se servant de l'éditeur d'image, et construire son propre synoptique. Ce service est très utile et autorise une ergonomie adaptée.

De plus, la création automatique de l'animation permet une interprétation des résultats plus efficace et un pilotage plus confortable.

Conclusion générale

Conclusion générale

Notre étude s'inscrit dans le cadre de la modélisation et la simulation de systèmes industriels de production. L'accent à été mis sur la définition d'un modèle d'architecture informatique pour la conception de plates-formes de simulation interactive, basée sur une approche totalement orientée objet en séparant l'interface de l'application.

Ce modèle est structuré en trois éléments : la présentation, l'objet de simulation et le traducteur.

- La présentation définit l'image du système à simuler, c'est à dire son comportement en entrée comme en sortie vis à vis de l'utilisateur.
- L'objet de simulation désigne les concepts et les fonctions du modèle de simulation.
- Le traducteur maintient une cohérence des domaines abstrait (objet de simulation) et graphique (présentation).

Lors de sa modélisation, un système est décomposé en plusieurs éléments matériels ou immatériels réunis grâce à un mécanisme de communication leur permettant d'être en interaction en exerçant une influence les uns sur les autres.

Chaque élément du système est représenté par un objet de simulation à contraintes. La communication entre les objets est assurée par l'introduction des objets liens. Le rôle des objets liens est de maintenir une cohérence entre un certain nombre de variables appartenant à des objets de simulation différents. Cette cohérence permet de garantir la propagation d'un état quelconque dans le réseau du système et un comportement global adéquat.

De l'autre côté de la conception, on trouve la partie visuelle du système qu'est le synoptique. Ce dernier fait partie intégrante de l'interface utilisateur.

Nous avons vu que la conception d'une interface homme-machine est complexe et nécessite l'utilisation d'un modèle d'architecture informatique et nous avons distingué deux catégories de modèles :

- Les modèles classiques bâtis sur la grammaire d'interaction, ou sur des niveaux d'abstraction successifs. Ces modèles conduisent à une complexité du dialogue homme-machine.
- Les modèles multiagent (à objet) ou le dialogue est réparti sur une multitude d'objets ou agents de présentation et s'inscrit dans le cadre de l'approche objet-action. Les événements, actions de l'utilisateur sur des objets, se traduisent en messages adressés à ces objets. Ces messages sélectionnent les méthodes responsables des comportements internes et externes des objets. Les interfaces basées sur ce modèle sont appelées interfaces événementielles et s'adaptent mieux avec l'utilisateur créatif et non programmé.

L'interface de simulation interactive repose sur la combinaison des modèles MVC de Smalltalk et le modèle P.Os.T que nous avons développé spécialement pour représenter les objets du modèle de simulation. En tirant profit des avantages du paradigme objet, le synoptique est structuré sous forme hiérarchique en plusieurs niveaux. Chaque niveau regroupe un ensemble de présentations et chaque présentation est liée à l'objet de simulation représenté grâce à un objet traducteur.

Une plate-forme de simulation interactive est donc définie comme un ensemble de modèles P.Os.T encapsulés dans une fenêtre principale MVC. Le dialogue est réparti sur l'ensemble des présentations et le contrôleur de la fenêtre principale. Ce dernier, appelé contrôleur graphique s'adapte selon le contexte et gère les requêtes utilisateur concernant l'environnement de simulation. Les autres requêtes sont confiées aux présentations elles mêmes.

La création d'une application interactive de simulation passe par les phases suivantes : prototypage des modèles P.Os.T, construction du synoptique et instanciation du modèle de simulation. La mise en oeuvre de la simulation permet à l'utilisateur de tester la validité de son modèle.

Le modèle P.Os.T a été testé dans des domaines très variés : les processus par lots, les systèmes d'irrigation automatiques et les systèmes de transport automatisés, ...etc.

Les points forts du modèle P.Os.T sont particulièrement son adaptation aux différentes applications et la création automatique de l'animation.

Par ailleurs, il est intéressant de noter l'état actuel de la plate-forme et les quelques perspectives essentielles :

Développement d'un outil de modélisation plus naturel, exploitable par un utilisateur naïf. En effet dans l'état actuel des choses, la modélisation est réservée aux seuls connaisseurs de la programmation à objet et de la notion de contrainte qui n'est pas si facile à implémenter dans les objets afin d'avoir un comportement adéquat.

Des travaux orientés dans ce sens ont déjà commencé et concernent notamment la modélisation de la partie commande. En effet, telle qu'on l'a présentée dans ce mémoire, la commande est vue comme un ensemble d'objets de simulation particuliers appelés objets missions. Ainsi, spécifier la commande d'un procédé consiste:

- à identifier l'ensemble des missions qu'il doit effectuer.
- ensuite à écrire chaque mission différente sous forme de classe.

Chaque mission possède une tâche particulière et utilise un certain nombre de variables appartenant à la partie opérative. De ce fait, la classe à laquelle appartient la mission doit définir autant de variables d'instances. Ces variables seront liés par l'intermédiaire des objets lien aux variables de la partie opérative. De plus, l'écriture des contraintes de la mission consiste à exprimer des relations entre ces variables d'instances. Le modèle de l'objet mission se trouve figé par sa structure.

Afin de surmonter ces limitations, en particulier la généricité, il convient d'adjoindre un véritable module de saisie de la commande et intégrer les concepts importants qu'on retrouve dans tout système de commande : le diagnostic et le traitement de défaillances.

Par ailleurs, d'autres travaux s'inscrivent dans le projet d'extension de la plate-forme de simulation afin de tester l'apport des techniques "systèmes experts". Ces

travaux concernent l'ajout de l'expertise multiparadigme, d'en concevoir les conséquences et les perspectives aux simulations orientées objet, notamment les connaissances empiriques des experts humains et les connaissances propres à la simulation. Ces connaissances interviennent pour la détection et le diagnostic de défaillance, la marche dégradée, la reprise après anomalie, la décision après détection de défaillance, l'optimisation, et d'un certain point de vue à la formation de pilotes de processus. La coopération avec la partie commande permettra une modification de l'objet *mission* et une démarche beaucoup plus souple à la conception et aux modifications de la simulation.

Par ailleurs, du coté de l'interface homme-machine, notre outil offre un moyen efficace aux utilisateurs leur permettant de personnaliser leur synoptique et surtout leurs animations. La personnalisation de l'interface est obtenue grâce à l'éditeur d'images et donne des résultats très satisfaisants, par contre la personnalisation des animations est bien sûr limitée à ce que peut offrir la bibliothèque des méthodes de traduction. Il serait nécessaire d'approfondir ce point, par l'enrichissement de la bibliothèque, afin de proposer des choix qui couvrent pratiquement tous les besoins des utilisateurs.

Bibliographie

Bibliographie

Les Interfaces Homme-Machine

[**ABI 86**] S. Salim ABI _EEZZI, and Albert J. BUNSHAFT.
'An implementation of PHIGS: Programmer's Hierarchical Interactive Graphics System'.
IEEE CG & A, February 1986, pp 13 - 23.

[**AFN 90**] AFNOR.
'Controle de process: Symboles graphiques et codes d'identification'.
Mesures, fiche N° 30, 25 juin 1990.

[**BAR 87**] H.A. BARKER, M. CHEN , C. P. JOPLING and P. TOWNSEND.
'Interactive Graphics for the Computer Aided Design of Dynamic Systems'.
IEEE Control System Magazine, june 1987.

[**BAR 89**] H. A. BARKER, P. W. GRANT, C. P. JOPLING, J. SONG and P. TOWNSEND.
'A Graphical Man-Machine Interface for Discrete-Event Dynamic Systems'.
ESC 89, Proceeding of the 3rd European Simulation Congress,
Sept, 5-8, 1989 Edinburgh, Scotland,

[**BAR 89**] H.A. BARKER, M. CHEN and P.W.GRANT, C.P. JOPLING and P. TOWNSEND.
'A Man-Machine Interface for Computer-Aided Design and Simulation of Control Systems'.
Automatica vol. 25, N° 2, pp 311-316, 1989.

[**BAR 89**] H.A.BARKER, T. Huynh. QUOC and P. TOWNSEND.
'A Graphical Preprocessor for Continuous System Simulators'. ESC 89, Proceeding of the 3rd European Simulation Congress, Sept, 5-8, 1989 Edinburgh, Scotland, pp 349-355.

[BIR 85] Graham BIRTWISTLE.
'AI, Graphics and Simulation'.
The Society for Computer Simulation, Copyright 1985, Simulation Council, INC.

[BOD 89] Robert BODU.
'L'opérateur et la conduite informatisée des procédés'.
Industrie Minérale, Mines & Carrières, vol 71, Novembre 89.

[BOR 81] Alain BORNING.
'The programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory'. ACM Transaction on Programming Languages and Systems,
vol 3, N° 4, October 1981, pp 353-387.

[BOR 86 A] Alain BORNING and Robert DUISBERG.
'Constraint Based Tools for Building User Interfaces'.
ACM Transaction on Graphics, vol 5, N° 4, October 1986, pp 345-374.

[BOR 86 B] Alain BORNING.
'Defining Contraints Graphically'.
CHI'86 Proceedings, ACM, April 1986, pp 137-143.

[BRA 90] Giorgio BRAJNIK, Giovanni GUIDA and Carlo TASSO.
'User Modelling in Expert Man-Machine Interfaces: A Case Study in Intelligent Information Retrieval'. IEEE Transactions on Systems, Man and Cybernetics,
Vol 20, N° 1, January/February 1990.

[BRU 88] S. BRUNEAUX.
'Etude d'un système d'animation en temps réel'.
Proceeding for MICAD 88.

[BUT 89] Berta BUTTRAZZI.
'Advanced Graphics Tools For Simulation Programs in a Sun Workstation Environment',
ESC 89, Proceeding of the 3rd European Simulation Congress,
Sept, 5-8, 1989 Edimburgh, Scotland, pp 378-381.

[CAC 90] P.C. CACCIABUE, G. MANCINI, and U. BERSINI.
'A Model of Operator Behavior for Man-Machine System Simulation'.
Automatica vol. 26, N° 6, pp 1025-1034, 1990.

[CLA 86] A. A. CLARKE.
'A three level Human-Computer Interface Model'.
International journal of Man-Machine Studies, 1986, vol 24, pp 503 - 517.

[CLA 89] J.A. CLARKE and J.H. RUTHERFORD.
'An Intelligent Fron-End for Computer-Aided Building Design'.
ESC 89, Proceeding of the 3rd European Simulation Congress,
Sept, 5-8, 1989 Edinburgh, Scotland.

[CLA 91] Mark A. CLARKSON.
'An Easier Interface'.
Byte February 1991.

[COH 86] Ellis S. COHEN, Edward T. SMITH, and Lee A. IVERSON.
'Constraint-Based Tiled Windows'.
IEEE CG & A, May 1986, pp 35-45.

[CON 87] J.CONKLIN.
'HyperText, An Introduction and Survey'.
IEEE Computer, vol 20, 9 sept 1987, pp 17-41.

[COS 88] COSTEA.
'Graphically Animated Simulation for Manufacturing'.
Proceeding for MICAD 88.

[COU 85] J. COUTAZ et V. JOLOBOFF .
'Abstractions for User Interface Design'.
IEEE Computer, vol 18, 9 september 1985, pp 21-34.

[COU 88] J. COUTAZ et V. JOLOBOFF .
'Méthodes et Outils pour l'Implementation des Interfaces Modernes'.
Cours INRIA-CERICS, Interfaces Utilisateurs Evoluées, 1988.

[COU 88] J.COUTAZ.
'Interface Homme Ordinateur,'.
Thèse de Doctorat d'état, Université Joseph Fourier de Grenoble, decembre 1988.

[COU 90] J. COUTAZ .
'Interfaces Homme - Ordinateur, Conception et Réalisation'.
Dunod Informatique, Bordas, Paris, 1990.

[CZE 90] Bogdan CZEDJO, Ramez ELMASRI, and Marek RUSINKIEWICZ.
'A Graphical Data Manipulation Language for an Extended Entity Relation -Ship Model'.
IEEE Computer, 1990, pp 26-36.

[DEL 90] Ph DELABRE et B CAUSSADE.
'Evolution des Opérateurs face aux Changements Technologiques sur un Haut-Fourneau'.
Industrie Minérale, Mine & Carrière, Décembre 90, pp 34-48.

[DUI 86] Robert A.DUISBERG.
'Animated Graphical Interfaces Using Temporal Contraints'.
Computer Human Interaction Conference Proceeding, ACM,
New York, April 1986, pp 131-136.

[DUR 91] William K. DURFEE, Mattew B. WALL, Derek ROWELL and Freeland K. ABBOH.
'Interactive Software for Dynamic System Modelling Using Linear Graphs'.
IEEE Control Systems magazine, june 91, pp 60-66.

[EGE 87] Raimund K.EGE, David Maier and Alan BORGING.
'The Filter Browser Defining Interfaces Graphically'. Proceeding of ECOOP'87, The European Conference on Object Oriented Programming, june 1987, pp 155-165.

[FAL 82] P.FALZON.
'Structure des dispositifs de présentation de l'information et compatibilité avec les représentations mentales'. Congrès SELF, PARIS, Octobre 1982.

[FIS 89] Gerard FISHER.
'Human-Computer Interaction Software: Lessons Learned, Challenges Ahead'.
IEEE Software, 1989, pp 44-52.

[GEN 86] Phillippe GENOUD et Jean Francois GRABOWIECKI.
'Vers un logiciel graphique interactif de haut niveau: CLOVIS'.
Proceeding of MICAD 86.

[GRA 86] Manual GRÄNA, Fernando FERRERES, Fransisco J.TORREALDEA
and Javier J DOLADO
'a GPSS like Hierarchical Language'. ESC 86, Proceeding of the 2nd European Simulation Congress, sept. 9-12, 1986, pp 275-281, Antwerp, Belgium.

[GRU 91] Jonathan GRUDIN, Aharhus University.
'Interactive Systems: Bridging the gaps between Developers and Users'.
IEEE Computers, April 1991.

[HAY 89] Alan J HAYWOOD and Stewart JOHNSON.
' Graphics Tools in Simulation '. ESC 89, Proceeding of the 3rd European Simulation Congress, Sept, 5-8, 1989 Edinburgh, Scotland.

[HOU 88] G. HOUBART.
'Des Objets et des Icônes pour l'Interface Utilisateur du Systeme de Conception Graphique COGITO '. Proceeding of MICAD 88.

[HUL 86] J. M. HULLOT.
'S O S Interface, un Générateur d'Interface Homme - Machine'. Actes des journées Afcet-Informatique sur les langages orientés objet, Bigre + Globule, vol 48, IRISA, Campus de BAULIEU, 35042 Rennes, janvier 1986, pp 69 - 78.

[HUL 86] J. M. HULOT.
'La construction d'Interfaces Homme-Machine'.
Journées AFCET, << Langages Orientés Objet >>, PARIS, 1986 .

[KAR 87] S.KARSENTY.
'Graffiti: un outil interactif pour la construction d'interfaces homme-machine adaptables'.
Thèse de Doctorat de troisième cycle Informatique, Université de Paris-Sud, centre d'Orsay, décembre 1987.

[LEH 86] A.LEHMANN, B.KNÖDLER, E.KWEE and H.SZCERBICKA.
'Interactive Modelling and Simulation in An Intelligent PC-Environment'.
ESC 86, Proceeding of the 2nd European Simulation Congress,
Antwerp, Belgium, sept. 9-12, 1986, pp 347-353.

[LIN 89] M.A.LINTON, J.M.VLISSIDE, P.R.CALDER.
'Composing User Interfaceswith InterViews'.
IEEE Computer, february 1989, pp 8-22.

[MAL 89] John H.MALONEY.
'Contraints Technology for User Interface Construction in ThingLab II'.
OOPSLA'89 Proceedings, October 1-6, 1989, pp 381-388.

[MAR 91] Aaron MARCUS and Associates Andries VAN DAM, Brown University.
'User Interface Developments for the Nineties'.
IEEE Computer, September 1991, pp 49-57.

[MEU 87] Pieter S.Van der MEULEN.
'INSIST: Interactive Simulation in Smalltalk'.
OOPSLA'87 Proceedings, October 4-8, 1987, pp 366-376.

[MIL 91] Frédéric MILLIOT et Stephane DESCLAUX.
'Windows, IA et POO'.
Micro-Systemes, Avril 1991, pp 181-182.

[MOR 81] Thomas P. MORAN.
'TheCommand Language Grammar: A Representaion For User Interface of Interactive ComputerSystems '. Internationnal journal. Man-Machine Studies, 1981, vol 15, pp 3-50.

[MOS 92] Mohammed MOSTEFAI.
'An Intelligent Man-Machine Interface for Interactive Simulation of Industrial Processes '.
Proceeding of the1992 European Simulation Symposium, pp 253-256.

[MOS 93] Mohammed MOSTEFAI.
'P.Os.T : An architectural model for Interactive Simulation Interfaces Design '. IEEE / SMC'93 Conference, Systems Engeneering in the service of the Humans, le Touquet-France, October 17-20, 1993, vol 2, pp 550-555.

[MUK 90] R.MUKUNDAN and S.S.SWAMY.
'A Procedure for Interactive Simulation Using Mnemonic Identifiers and Index Mapping Fonctions'. Simulation, volume 55: Number 1, july 1990, pp 39-46.

[NAK 89] Y.NAKAMORI.
'Development and Application of an Interactive Modelling Support System'.
Automatica vol. 25, N° 2, pp 185-206, 1989.

[PAY 86] Stephen J. PAYNE and T.R.G. GREEN.
'Task - Action Grammars: A Model of the Mental Representation of Task Languages'.
Human - Computer Interaction, 1986, Volume 2, pp 93-133.

[PRI 82] Lynne A. PRICE.
'THUMB: An Interactive Tool for Accessing and Maintaining Text'.
IEEE Transactions on Systems, Man and Cybernetics,
Vol. SMC-12, N°2 , March/April 1982.

[RAS 83] J. RASMUSSEN.
'Skills, Rules and Knowledges; Signals, Signs and Symbols, and other Distinctions in Human performances model'.
IEEE Transactions, Man and Cybernetics, SMC 13,3, May/June 1983.

[SCA 86] D. SCAPIN.
'Guide Ergonomique de Conception des Interfaces Homme - Machine'.
Rapport INRIA N° 77, Octobre 86.

[SEN 88] B. SENACH.
'Introduction à l'Ergonomie des Interfaces Homme - Machine'.
Cours INRIA / CERICS, Interfaces Utilisateurs Evoluées, 1988.

[SEN 90] B. SENACH.
'Evaluation Ergonomique des Interfaces Homme - Machine: une Revue de la littérature'.
Rapport de recherche INRIA, N° 1180, 1990.

[SHN 82] B.SCHNEIDERMAN.
'Multiparty Grammars and related Features for Defining Interactive Systems'.
IEEE Transaction on systems, Man and Cybernetics,
SMC-12, vol 2, 1982, pp 148-154.

[SHU 86] D.SHUEY, D.BAILEY, T.P.MORRISSEY.
'PHIGS: A standard, Dynamic,Interactive graphics Interface'.
IEEE Computer graphics and application, August 1986, pp 50-57.

[SZE 88] Pedro A.SZEKELEY and Brad A.MYERS.
'A User Interface Toolkits Based On Graphical Objetcs and Contraints'.
OOPSLA'88 Proceedings, ACM Conference on Object Oriented Programming
Systems and Applications, San Diego, California, sept, 25-30, 1988, pp 36-45.

[THA 91] Nadia Magnetat-THALMANN.
'Complex Models for Animating Syntetic Actors'.
IEEE CG & A, september 1991, pp 32 - 44.

[WAL 86] Franck J.WALES and Paul A.LUKER.
'Design of a Multi-Level User Interface for Descrete Event Simulation'.
ECS 86, Proceeding of the 2nd European Simulation Congress,
sept 9-12, Antwerp, Belgium, pp 361-366.

[YUN 89] S.K.YUNG and W. CLARKE.
'Local Sensor Validation'.
Measurment and Control, Volume 22, June 1989.

Simulation & Modélisation orientés objet.

[ADE 87] H.ADELSBERGER & F.BROECKX.
'Discret Event Simulation and Operations Research'.
Proceedings of the European Simulation MultiConference,
Wirtschaftsuniversität, VIENNA, AUSTRIA, july 7-10, 1987.

[ALT 89] J. L. ALTY and J. JOHANNSEN.
'Knowledge-Based Dialog for Dynamic Systems'.
Automatica vol. 25, N° 6, pp 829-840, 1989.

[CAN 88] E.CANTEGRIT.
' Modélisation et simulation orientéés objets des processus par lots '.
Thèse de Doctorat, Université des Sciences etTechnologies de Lille, 23 février 1988.

[BAN 86] Stephen P.BANK.
'Control System Engineering'.
1986 Prentice - Hall International (UK) Ltd.

[BAU 87] B.BAUDEL, E.CANTEGRIT, Professeur J.M.TOULOTTE.
' Utilisation de la notion d'objet pour la modélisation et la simulation des processus par lots '. Congrès I.A.S.T.E.D, Paris, june 87, pp 118-122.

[BEK 91] M.H.CHILOUP-BEKAERT.
'Utilisation de la notion d'objets avec contraintes pour la modélisation et la simulation des systèmes de production'. Thèse de Doctorat troisième cycle en Productique : Automatique et Informatique Industrielle, Université des Sciences et Technologies de Lille, décembre 91.

[BEL 85] G.BEL, D.Dubois.
'Modélisation et simulation de systèmes automatisés de production'.
APII-1985, vol.19, N° 1, pp 3-43.

[BEL 90] Gérard BEL, Jean Bernard CAVAILLE.
'Intégration de la simulation dans la conception de systèmes de production: Avantages et dangers de l'approche par langages à objets'.
Colloque international CIM 90, 12-14 juin 1990, Bordeaux.

[BEZ 84] Jean BEZIVIN.
'Simulation et Langages orientés objet'.
journées L.O.O de Brest, Bigres N° 41, Novembre 84.

[BEZ 87] Jean BEZIVIN.
'Some experiments in object-oriented simulation'.
OOPSLA '87 proceeding, October 4 - 8, 1987, pp 394-405.

[BEZ 90] Jean BEZIVIN, Bertrand MEYER and Jean Marc NERSON.
'Technology of Object Oriented Languages and Systems'. TOOLS2, Proceedings of the Second Internationnal Conferencs TOOLS, Paris 1990.

[CAN 90] Sylvie CANDELIER et Franck MATKOWSKA.
'Réalisation d'une plateforme de simulation pour les systemes de transports guidés en utilisant la programmation par objets'. Projet de fin d'etudes, ISEN, 1990.

[COD 90] J.F.COUDURIER.
'La Simulation des Flux de Production'.
Rapport d'Etude N° 105330, Centre Technique des Industries Mécaniques, 1990.

[COM] M. COMBACAU & M. COURVOISIER.
'A Hierarchical and Modular Structure for F.M.S Control and Monitoring'.
LAAS C.N.R.S.

[CRO 90] Société CROUZET - Valence.
'Contribution des langages orienté objet au prototypage d'organes de dialogue Homme - Machine en milieu embarqué', rapport d'activité, 1990.

[EUR 88] Euristic Systems.
'CHRONOS: l'intelligence du temps'.
Notice technique de présentation, Novembre 88.

[EUR 88] EURISTIC SYSTEMS.
'CHRONOS: outil de developpement de systèmes experts temps réel'.
Notice technique de présentation, Novembre 88.

[FAY 90] Beatrice FAYOLLE.
'Spécification et validation d'automatismes à haut niveau de sureté de fonctionnement appliqués aux transports terrestres automatisés'. These de Docteur Ingenieur, soutenue le 24 Avril 1990, Université des Sciences et Technologies de Lille.

[FAZ 90] Karl Heinz FAZOL & Georg Michael POHL.
'Simulation, Controler Design and Field Test for Hydropower - a case study'.
Automatica vol 26, N° 3, pp 475 - 485, 1990.

[FEN 90] S.J. FENVES, U.F. LEMMING, C. HEDRICKSON, M.L. MAHER, and G. SCHMITT.
'Integrated Software environment'. ButterWorth & Co(Publisher) Ltd, Computer Aided Design, vol 22, N°1, Jan/Feb 1990.

[FLA 89] Frédéric FLAMANT et Daniel HODOUIN.
'Simulation des procédés minéralurgiques'.
Industries Minérales; mines & carrières, Les techniques, pp 4-89.

[GAL 81] P.W. GALLIER, L.B. EVANS, H.I. BRITT, J.F. BOSTON.
'ASPEN : Advance System for Process Engeneering '.
Perspectives in Computing, 1(1), pp 43-49, 1981.

[GUA 90] Antonio GUASCH.
'Object Oriented Simulation'.
Proceedings of the MultiConference on Object Oriented Simulation 17-19 january, 1990, SanDiego, California.

[HUM 89] K. HUMPHREYS and A.FOSS.
'A Block-Based Graphical Approach for the description and Analysis of Dynamic Systems'
ESC 89, Proceeding of the 3rd European Simulation Congress,
Sept, 5-8, 1989 Edinburgh, Scotland.

[KET 86] Dirk L.KETTENIS.
'COSMOS: A member of a new generation simulation languages'.
ESC 86, Proceeding of the 2nd European Simulation Congress,
sept. 9-12, 1986, pp263-269, Antwerp, Belgium.

[KET 92] Dirk L.KETTENIS.
'COSMOS: Simulation Language for Continuous, Discrete and Combined models'.
Simulation, volume 58: Number 1, january 1992, pp 32-41.

[MAR 90] S. MARIE, A. Adam - NICHOLE, D. VILLEMIN.
'Utilisation des représentations par objets en chimie organique'.
Actes de la convention IA 90.

[MAR 91] Gérard MARTIN, Mohammed MOSTEFAI et Rachid IKNI.
'A Knowledge based Simulation System for Industrial Processes Control'.
Proceeding of COMMADEM 91 : The third Internationnal Congress on Condition Monitoring and Diagnostic Engineering Management Helt Southampton Institute, 2-4 July 1991, pp 60-64.

[MEG 91] Carlo MEGHINI, Fausto RABITTI, and Costantino THANOS.
'Conceptual Modeling of Multimedia Documents'.
IEEE Computer, October 1991 pp 23-30.

[MOS 91] Mohammed MOSTEFAI, Rachid IKNI.
'Modelling and Simulation of Irrigation Control System'.
Proceeding of the 1991 European Simulation Symposium, pp 206-211.

[OFT 89] O.F.T.A.
'Systemes Experts et Conduite de Processus'.
Observatoire Français Des Techniques Avancées, Masson, Paris, 1989.

[PRA 91] Herbert PRAHEOFER.
'Systems Theory Instrumented Modeling and Simulation Methodology'.
Hemisphere Publishing Corporation, Cybernetics and Systems:
An intrenational journal 22, pp 283-312, 1991

[RAM 92] Balshekar RAMCHANDRAN, James B. RIGGS, Hubert R. HEICHELHEIM, A. Franc SEIBERT and James R. FAIR.
'Dynamic Simulation of a Supercritical fluid Extraction Process'.
American Chemical Society, Ind. Eng. Chem. Res, 1992, pp 281-290.

[ROS 77] E.M. ROSEN, A.C. PAULS.
'Computer aided chemical process design : The Flowtran system '.
Computer & Chemical Engeneering, 1, pp 329-352, (1977).

[TSU 91] Kazuyuki TSUDA, Kensaku YAMATO, MASAHITO, KIRAKAWA, Minoru TANAKA, and Tadao ICHIKAWA.
'MORE: An object oriented data model with a facility for changing object structures'.
IEEE Transaction on knowledge and data engeneering, vol. N° 4, December 1991 .

[WAL 75] D.WALTZ.
'Understanding line drawings of scenes with shadows'.
P.H.Winston (Ed.), The Psychologie of Computer Vision, Mc Graw-Hill, New York, 1975.

[WOL 90] F. WOLINSKI.
'Représentation de systèmes robotiques en Smalltalk 80'.
Actes de la conférence européene sur les techniques et les applications de l'intelligence artificielle en milieu industriel et de service, janvier 90, Palais des Congrès.

Multimédia

[FOX 91] Edward A. FOX.
'Advances in Interactive Digital Multimedia Systems'.
IEEE Computer, October 1991 pp 9-21.

[HOD 89] Mattew E. HOFGES, Russell M. SASNET, and Marc S. ACKERMAN.
'A Construction Set for Multimedia Applications'.
IEEE Software, january 1989, pp 37-43.

[**MEG 91**] Carlo MEGHINI, Fausto RABITTI, and Costantino THANOS.
'Conceptual Modeling of Multimedia Documents'.
IEEE Computer, October 1991 pp 23-30.

[**SAN 92**] Didier SANZ et Laurent CLAUSE.
'Preparez vous au Multimedia'.
SVM, Mars 1992, pp 68-132.

[**THO 91**] THOMAS D. C Little, Arif GHAFOR.
'Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks'. IEEE Computer, October 1991 pp 42-50.

[**VIN 91**] Harrick M. VIN, P.T. ZELLWEGER, D.C. SWINEHART, and P.V.RANGAN.
'Multimedia conferencing in the Etherphone Environment'.
IEEE Computer, October 1991, pp 69-79.

Les Langages Orientés Objet

[**ALE 89**] Emmanuel ALEXANDRE.
'Des objets pour programmer'.
INFO PC N° 55, Octobre 1989, pp 177-183.

[**BEZ 86**] Jean BEZIVIN.
'Langages Objets et Prototypage'.
BIGRE + Globule N° 51, 1986, pp 23-32.

[**BEZ 88**] Jean BEZIVIN.
'Langages à objets et programmation concurrente: quelques expérimentations avec Smalltalk80'.
BIGRE + Globule N° 59, 1988, pp 176-187.

[**GOL 83**] A.GOLDBERG, D.ROBSON,
'Smalltalk 80 and it's implementation'.
Addison-Wesley Publishing Company, 1983.

[ROS 90] J. P. ROSEN.
'Object Oriented Design VS.Object Oriented Programming: a comparative assesment'.
ADALOG 1990, pp 1-9.

[THO 89] Dave THOMAS, Peter WEGNER, Mahesh H. DODANI, Charle H. HUGHES, J. Michael MOSHELL, and Tom THOMPSON.
'What's an Object, Learning the language, Separation of powers, The next step, Object oriented ressources'. BYTE 1989, pp 231-262.

[YOU 89] Douglas A. YOUNG.
'X Window, Programmation avec les Xt Intrinsics'.