INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

---

présenté pour obtenir le titre de

DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Option GÉNIE DES PROCÉDÉS

présentée par

Francisco Cardigos dos Reis

Intitulée

ACRUN - Contribution de la Technologie Orientée Objet à l'Intégration Logicielle de l'Acquisition et de la Commande

---

Soutenue publiquement le

devant la commission d'examen

Rapporteurs	P. BOURSEAU
	X. JOULIA
Examinateurs	J.P. CORRIOU
	J. LOURENÇO
	A. NAPOLI

ACRUN - Contribution de la Technologie Orientée Objet à l'Intégration Logicielle de l'Acquisition et de la Commande

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION

L'objectif du contrôle de procédé est l'automatisation des procédés industriels. Il fournit des bénéfices significatifs aux industries, tels qu'une productivité accrue, une amélioration de qualité de produit et une supervision plus facile des opérations.

Un système de contrôle de procédé est souvent général et peut s'appliquer à de nombreux domaines différents: surveillance de procédés chimiques, biologiques, agroalimentaires, médicaux, ... ; il concerne aussi bien la commande PID classique qu'une commande plus récente basée sur des réseaux neuronaux ou la logique floue; il concerne aussi l'informatique du matériel au logiciel. Pourtant, la réalisation d'un système de commande de procédé exige habituellement une longue période incluant des études théoriques, une expérimentation, une mise au point sur place et l'opération. Raccourcir cette période est toujours le but de promoteurs de systèmes de contrôle de procédé.

Pendant longtemps, dans le domaine du contrôle de procédé, le concept de bloc a été utilisé. L'avantage de bâtir des systèmes complexes en connectant des blocs de fonction plus simples par des raccordements visuels (parfois cité comme programmation graphique) est de simuler la façon dont les gens modélisent les problèmes [STANLEY 91]. Aussi, dans cette industrie, une grande variété de fonctionnalités doit coexister, allant du graphisme aux compteurs, de la gestion de données aux échanges de données en réseaux, du contrôle d'accès d'utilisateur aux procédures de sécurité des alarmes. Dans tous ces domaines, l'analyse, la conception, la construction de logiciel orientée objet peuvent amener des contributions importantes telles que fiabilité, facilité d'entretien, extensibilité et adaptabilité [DEDENE 94].

Aujourd'hui, des techniques orientées objet sont employées pour développer des applications dans une grande variété de domaines. Dans ce travail, la contribution principale des techniques orientées objet que l'on explore est de réaliser l'intégration dans le logiciel de contrôle de procédé. Dans l'Intégration de Présentation, l'Intégration de Procédé, l'Intégration de Données ou l'Intégration de Contrôle, on

étudiera comment un programme comme ACRUN peut être performant en traitant un grand nombre de facettes du développement de la commande et de mise en oeuvre de procédé. Ce dernier aspect d'intégration est crucial parce que la simulation et le contrôle temps réel devraient être réalisables dans un seul environnement, requérant seulement des changements mineurs pour se déplacer de l'un à l'autre [MCCROSKEY 91].

Afin de réaliser le niveau maximum d'intégration, il est nécessaire d'employer des normes pour des systèmes de communication et logiciel [MESSINA 94], donc on expliquera le choix du système opératif graphique Windows comme plate-forme de développement. L'emploi de mises en oeuvres différentes de Windows et leur conformation à l'opération temps réel, à l'échange de données, aux normes des objets, et au travail en réseaux sont discutées. La différenciation des systèmes traditionnels des systèmes temps réel [BETES 94] et la manière dont ACRUN traitera avec eux est aussi un sujet de ce travail. Des exemples de l'emploi du programme seront exposés notamment pour le développement avancé de contrôle et pour la surveillance (le procédé d'extraction et de collecte d'information en ce qui concerne le comportement d'un système particulier [SCHMID 94]) dans les usines.

Les avancées récentes dans l'équipement informatique de même que le développement de nouvelles méthodes de logiciel (comme la programmation orientée objet) ont ouvert de nouvelles perspectives pour concevoir de meilleures interfaces d'utilisateur [PANGALOS 93]. Ainsi, en s'appuyant sur les normes de l'interface utilisateur de Windows, de grandes ameliorations ont été faites pour obtenir un meilleur emploi de l'écran. Cela avait un but commun centré sur l'orientation objet qui était entreprise: créer une meilleure interface utilisateur pour gérer et intégrer différentes sortes d'objets (fonctionnalité).

Les domaines tels que la recherche dans la commande (développement d'algorithmes), l'ingénierie de procédé (mise en oeuvre d'usine), les systèmes d'ordinateur (utilisant les ports d'ordinateur et bus d'expansion), l'ingénierie de logiciel (programmation orientée objet) et la science cognitive (interface homme -machine) sont abordés dans cette thèse. Dans chacun de ces domaines, ce n'est pas une étude profonde et exhaustive qui est

visée, il s'agit seulement d'exposer ses caractéristiques principales et de démontrer comment ils sont intégrés l'un à l'autre.

Une certaine vision de l'avenir du contrôle de procédé est également présentée parce que, dans les années à venir, une plus grande intégration de systèmes d'information et de systèmes de contrôle de procédé aura lieu. Le même ordinateur sera employé pour la modélisation, la conception, le contrôle, l'opération et la sécurité [BENSON 92]. Cela deviendra possible par la puissance croissante des ordinateurs, par les nouvelles pratiques de développement logiciel, par les réseaux plus rapides et les communications globales, et par les nouvelles interfaces d'utilisateur et possibilités multimédia. Ce travail tente de mettre en valeur l'intérêt des quatre thèmes principaux défendus par [HSU 94] pour des avances majeures au cours de la prochaine décennie: 1) gestion de systèmes multiples qui opèrent concurremment sur un réseau distribué; 2) architecture des systèmes à même de loger des systèmes hérités et d'ajouter de nouveaux systèmes; 3) exploitation de la technologie orientée objet dans des systèmes de production; 4) boucle de rétroaction de la production à la conception.

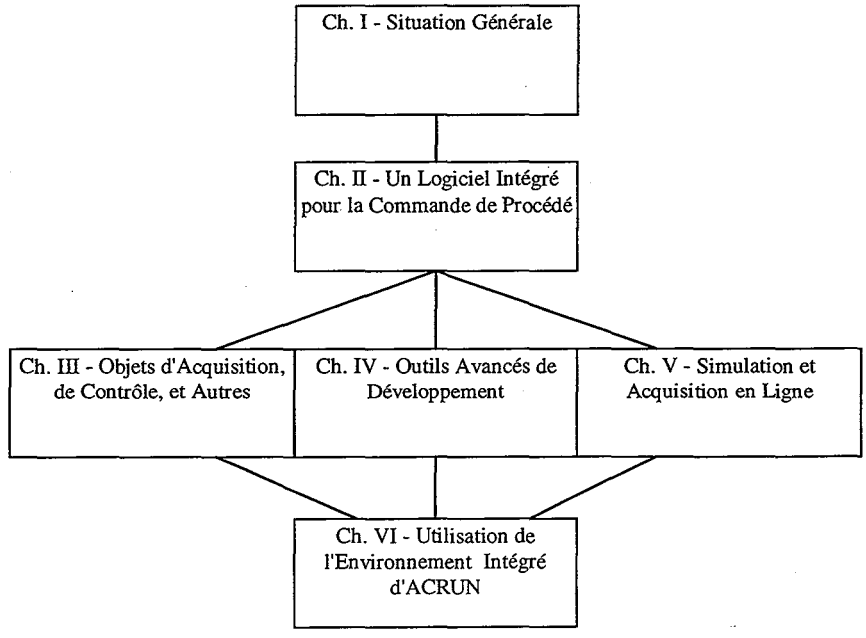


Figure 1 - Structure des Chapitres

Dans le Chapitre I, les besoins pour la mise en oeuvre et pour le développement de contrôle seront décrits ainsi que les types d'outils qui sont disponibles. Les pratiques actuelles d'ingénierie de logiciel et les tendances seront présentées avec une attention spéciale aux concepts orientés objets.

L'importance de l'intégration dans ses différents aspects est décrite dans le Chapitre II et une description des objectifs du projet ACRUN est présentée: accomplir simultanément les besoins de développement et de mise en oeuvre dans un environnement intégré. Cela sera important pour juger les réussites et échecs de ce travail et tirer des conclusions pour les améliorations futures et les corrections qui peuvent être faites.

Le chapitre III introduira les caractéristiques et attributs généraux des objets de ACRUN et leur mode de communication. Une description des objets disponibles dans le programme est faite de même que la manière dont ils sont intégrés les uns avec les autres. Cela non seulement montrera les caractéristiques disponibles, mais aidera aussi à illustrer comment les techniques orientées objet pourront intégrer des systèmes de contrôle et des systèmes d'information.

Les outils avancés sont décrits dans le chapitre IV. Ce sont des possibilités habituellement absentes dans les autres logiciels, ou des caractéristiques témoignant d'une nouvelle approche d'ACRUN. Ces outils visent essentiellement le développement de contrôle de procédé, mais les résultats obtenus sont aussi employés dans la mise en oeuvre des systèmes opérationnels.

Les différences sur l'emploi d'ACRUN si le travail de développement est réalisé hors-ligne ou une opération est effectuée en-ligne sont décrites dans le chapitre V. La nécessité de systèmes multitâches et de l'opération temps réel est discutée ainsi que les paramètres de taux d'échantillonnage soutenus par le programme. La synchronisation de données est aussi discutée dans l'environnement d'un seul ordinateur ou d'ordinateurs multiples, durant la simulation, l'acquisition ou la supervision.

Enfin, dans le chapitre VI, trois utilisations distinctes de ACRUN sont décrites: pour un étudiant dans un cours préliminaire de commande; pour un chercheur développant des algorithmes élaborés de commande; pour une usine désirant la surveillance de l'instrumentation distribuée dans l'unité de fabrication. Une comparaison brève avec d'autres produits est effectuée et l'utilité du projet entier en tant que mémoire et vitrine d'un groupe de recherche de contrôle est mise en relief.

La coopération université - industrie pour l'innovation des produits et des procédés [SUH 90], où chercheurs doivent être pertinents dans le transfert technologique vers l'industrie [BLAKELY 94], est importante. La citation ci-après de [GUY 94] constitue un bon exemple de la nouvelle mentalité nécessaire pour que cette coopération puisse avoir lieu:

> "Beaucoup de chercheurs paraissent considérer le développement d'outils industriels comme inintéressant et comme travail de développement de bas niveau. Les chercheurs qui ont contribué à la commercialisation de tels outils peuvent confirmer que les défis impliqués en faisant cela sont à la fois réels et intéressants".

Chapitre I - Situation Générale

I.0/ Introduction

Auparavant, beaucoup de procédés ayant des comportements dynamiques complexes étaient simulés par des ordinateurs puissants. Au cours de ces dernières années, grâce au développement rapide du logiciel et des équipements d'ordinateur, certaines stratégies de contrôle avancé sont maintenant adaptées à l'exécution en ligne et beaucoup d'outils de mise en oeuvre et de développement deviennent disponibles sur le marché. Par la suite, un grand nombre de systèmes de contrôle de procédé basés sur des ordinateurs et des micro-ordinateurs sont apparus.

Comme des ordinateurs d'usage général devenaient disponibles dans les universités, laboratoires et instituts de recherche, les promoteurs de systèmes de contrôle (utilisant jusqu'alors de puissants ordinateurs centraux) commencent à utiliser des stations de travail meilleur marché et plus accessibles. Les logiciels spécifiques de contrôle, utilisant essentiellement des compilateurs de langage, font rapidement leur entrée dans ce nouveau marché en expansion.

Longtemps appréciés au bureau, les ordinateurs d'usage général devaient affronter une grande réticence des industries requérant la commande de procédé, d'une part à cause du manque de périphériques de contrôle et d'acquisition spécifique, mais essentiellement parce que cela impliquait de changer de vieilles habitudes en innovant sur la mentalité d'usine très conservatrice. Une importante diminution du rapport prix/performance et une large gamme de solutions basée sur des ordinateurs d'usage général ouvrait enfin à ces plates-formes une part significative du marché. Les équipements d'acquisition et de contrôle sont maintenant relativement standardisés et le choix des outils logiciels appropriés à la mise en oeuvre des procédés est la tâche la plus importante et la plus difficile.

Deux domaines distincts étaient définis dans les marchés globaux de systèmes de contrôle de procédé implanté sur les logiciels d'ordinateur d'usage général: le premier,

le domaine du logiciel de développement de contrôle, visait à développer le contrôle de procédé pour une mise en oeuvre finale sur toutes sortes d'installations de contrôle de procédé; le second, le domaine de mise en oeuvre de contrôle de procédé, devait faire fonctionner le logiciel de contrôle accouplant l'ordinateur au procédé par l'équipement de contrôle et d'acquisition.

Des logiciels comme Matlab apparaissaient sur le marché du développement, et des logiciels comme Labtech/Control sur le marché de mise en oeuvre. Malgré tout, pour aller du développement à la mise en oeuvre, même en utilisant le même ordinateur, un manque de support évident apparaissait. Deux types différents de logiciels devaient être employés, l'un pour le développement et l'autre pour la mise en oeuvre, ou bien un compilateur de langage général était la solution ultime. Le logiciel pour le développement de contrôle (visant une large gamme de mise en oeuvre de procédé) a essayer de comblé cette lacune en ajoutant des produits spécifiques pour certaines mises en oeuvre. Du coté du logiciel de mise en oeuvre de contrôle, alors que certains outils de développement de la commande étaient ajoutés à ces produits, aucune solution complete de mise en oeuvre et de développement n'existe sur le marché [BENSON 92].

Pour comble de malheur, souvent l'ordinateur le mieux adapté au développement n'est pas celui employé pour le contrôle de procédé. Les deux ordinateurs étant d'usage général, est-ce qu'il ne devrait pas exister une voie facile qui relierait le développement de contrôle à la mise en oeuvre de procédé?

C'est ce que le projet ACRUN prétend explorer, répondant simultanément aux besoins de mise en oeuvre et de développement, s'appuyant sur un système d'exploitation tel que Windows NT qui tourne sur de multiples plates-formes, et éliminant chaque duplication possible de travail dans le cycle développement - mise en oeuvre. Diminuer l' " écart évident de transfert de technologie entre (...) les enthousiastes du développement et les utilisateurs finaux (...) de l'usine de fabrication" [PUIGJANER 91] est l'objectif principal.

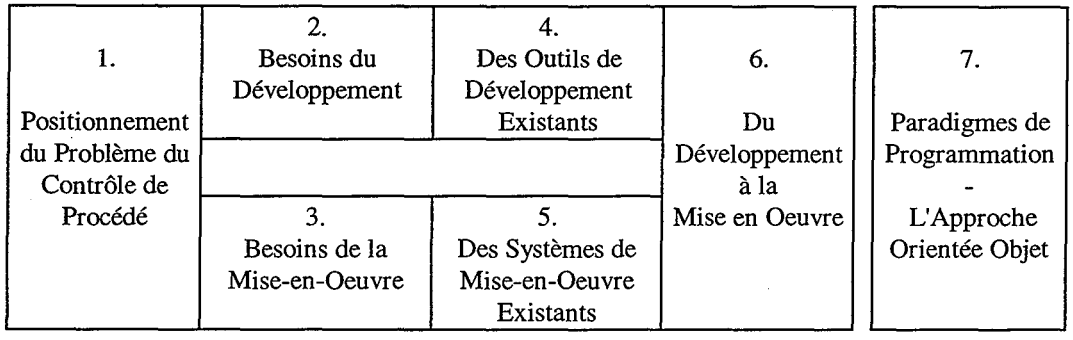


Figure 2 - Des Besoins de Contrôle de Procédé aux Outils Logiciels Disponibles

Ce chapitre montre les difficultés et mérites d'un logiciel intégré capable, pour des systèmes de contrôle de procédé, de couvrir les étapes de développement et de mise en oeuvre. Dans la section 1, le problème du contrôle de procédé est introduit, suivi par une description des besoins du développement (section 2) et de la mise en oeuvre (section 3) du point de vue du contrôle. Dans la section 4 et la section 5, la situation générale des outils de développement et des systèmes de mise en oeuvre est discutée. L'analyse de la manière possible de supprimer l'écart entre le développement de système de contrôle et la mise en oeuvre est le sujet principal de la section 6. Enfin, un aperçu des tendances actuelles dans l'ingénierie logicielle est donné en mettant l'accent sur les différences entre le paradigme fonctionnel et le paradigme objet (section 7).

I.1/ Positionnement du Problème du Contrôle de Procédé

Le contrôle de procédé traite des problèmes d'opérations automatiques dans l'ingénierie de procédé, tels que les procédés des industries pétrochimiques, biologiques et chimiques. Au cours des trois dernières décennies avec le développement des systèmes d'ordinateurs flexibles et puissants, de plus en plus de systèmes de contrôle par ordinateur ont été utilisés dans ces domaines [BENSON 89].

La performance du contrôle, la fiabilité des systèmes et les moyens de développement ont été beaucoup améliorés. La réalisation d'un système de contrôle de procédé prend

généralement beaucoup de temps, de l'étude théorique et expérimentale à la conception du contrôleur, du développement des logiciels à la mise en oeuvre en ligne , etc..

Commençons par analyser la structure générale d'un système de contrôle de procédé.

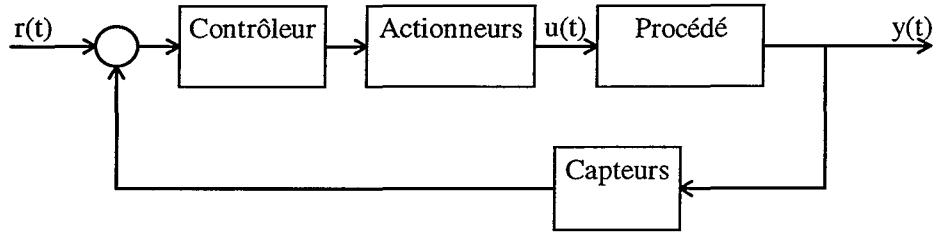


Figure 3 - Système de Contrôle d'un Procédé

Un système typique de contrôle de procédé peut être séparé en quatre parties principales (Figure 3): le procédé lui-même, les capteurs, les actionneurs et le contrôleur [LEVESON 94]. Le procédé peut être un réacteur chimique (discontinu, semi-continu ou continu), une colonne de distillation, un échangeur de chaleur, etc. . Le contrôleur peut être réalisé avec de l'équipement spécifique de contrôle, avec un système spécialisé d'ordinateur optimisé pour le contrôle de procédé, ou par un ordinateur d'usage général tel qu'un compatible IBM PC avec des accessoires pour l'acquisition et contrôle. L'interface relie les signaux entre le contrôleur et le procédé en vue de compléter le système de contrôle, réalisant essentiellement des conversions analogique-numériques et numérique-analogiques et le conditionnement des signaux.

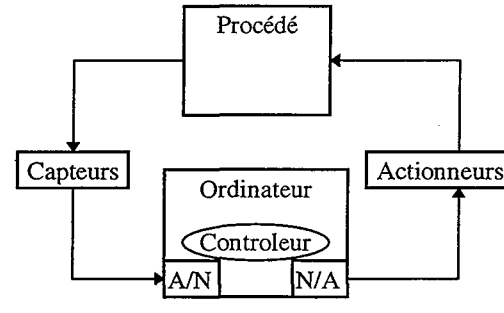


Figure 4 - Interface Ordinateur - Procédé

Un système complexe de commande de procédé est habituellement réalisé, afin d'assurer son succès, en deux étapes interconnectées: le développement et la mise en oeuvre. Par développement, il faut entendre identification du modèle représentatif du procédé, conception du contrôleur et simulation des systèmes de contrôle par

ordinateur. Par mise en oeuvre, il faut entendre l'application du contrôleur développé à un procédé réel et acquérant les données requises dans l'étape de développement.

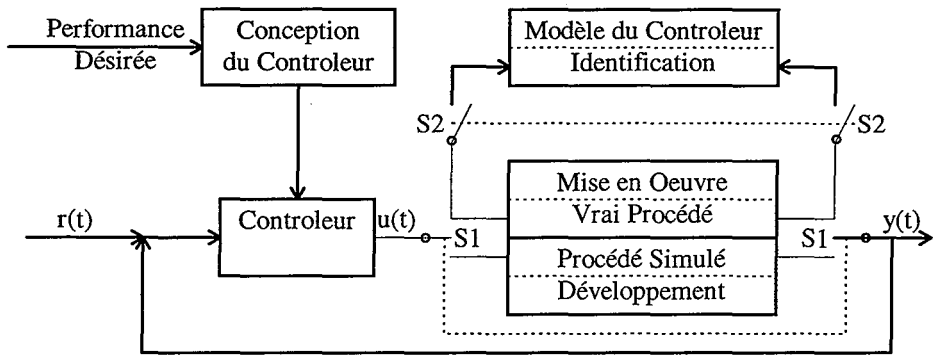


Figure 5 - Schéma de Développement et de Mise en oeuvre

Cette procédure est illustrée dans la Figure 5. Lors du développement, les commutateurs de contrôle S1 sont en position basse correspondant à l'étape de simulation utilisant le modèle du procédé fonctionnant avec un contrôleur en test. Quand l'algorithme de contrôle est obtenu, il sera alors appliqué au procédé réel, en plaçant les commutateurs S1 en position haute pour que les valeurs de commande soient appliquées au procédé (par des actionneurs) et que les variables du procédé soient lues (par des capteurs). Pour construire le modèle du procédé, des couples entrée-sorties du procédé sont requis, donc le commutateur S2 doit être fermé.

Pour exécuter ces procédures, certains besoins fondamentaux sont rencontrés qui peuvent aussi être partagés en deux aspects distincts: les besoins du développement et les besoins de la mise en oeuvre.

I.2/ Besoins du Développement.

Le développement en contrôle est une étape fondamentale très importante avant la mise en oeuvre. Il est concerné par les connaissances dans les domaines de l'ingénierie chimique, de l'informatique et du contrôle automatique. L'objectif n'est

pas de faire une description exhaustive de tous les besoins du développement, mais seulement de ses aspects importants et fondamentaux du point de vue du contrôle.

1. Le Traitement du Signal : cela inclut les filtres, les calculs statistiques, et aussi les opérations dans le domaine de la fréquence et du temps, afin de filtrer des signaux, d'analyser le comportement des procédés, de faire de l'identification, etc. . Pour valider le modèle de contrôle obtenu par l'identification paramétrique, certains calculs statistiques sont requis, tels que la variance et les autocorrélations des erreurs de prédiction.

2. L'identification : en contrôle, un modèle de procédé est requis, soit de connaissance, soit de comportement (le plus souvent) qui peut être obtenu par l'identification paramétrique (moindres carrés récursifs, méthode d'erreur de prédiction, maximum de vraisemblance, ...). Les séquences d'entrées/sorties du procédé sont préparées avant l'identification. Les données peuvent être enregistrées au moyen d'instruments d'enregistrement en ligne ou mesurées directement par un ordinateur. Le type de modèle, qu'il soit SISO (Single Input and Single Output) ou MIMO (Multiple Inputs and Multiple Outputs), dépend de la stratégie de contrôle, et est généralement exprimé sous forme discrète par rapport au temps. Une séquence d'excitation, par exemple binaire pseudo-aléatoire (SBPA), peut être appliquée en entrée pour obtenir des données d'entrées/sorties suffisamment riches. La SBPA peut être produite facilement par un ordinateur et est alors convertie en un signal analogique u(t) qui est l'entrée pour le procédé. En même temps, les sorties du procédé sont mesurées, en convertissant les signaux analogiques en numérique. Ensuite, les algorithmes conçus pour l'identification sont employés pour identifier la fonction de transfert discrète du procédé. Cela peut aussi être réalisé en ligne.

3. Le Contrôleur : c'est de plus en plus souvent un algorithme exprimant des lois de contrôle réalisées sur un ordinateur et qui doit garantir le comportement du procédé par rapport aux performances désirées. Il est possible d'expérimenter des contrôleurs différents (comme le PID, la commande par modèle interne, la commande adaptative prédictive, ou même la commande non-linéaire, etc.) sur le même procédé afin

d'obtenir une performance satisfaisante tout en assurant une stratégie de contrôle robuste, ce qui est essentiel sur le plan industriel. En général, les stratégies plus avancées peuvent conduire à une meilleure performance de contrôle, voire de robustesse, mais leurs algorithmes seront plus complexes et plus exigeants en puissance de calcul.

4. La Simulation de Procédé : avant la mise en oeuvre, une simulation par ordinateur du système de contrôle est requise. Pour faire cela, on doit simuler le comportement dynamique du procédé. Habituellement, cela correspond à la résolution numérique d'un système d'équations différentielles ordinaires et d'équations algébriques, éventuellement d'équations aux différences en temps discret ou même d' équations aux dérivées partielles qui seront discrétisées en général, soumises à des entrées.
Les modèles sont en général basés sur des bilans de matières, des bilans énergétiques, et des bilans de quantités de mouvement. L' ensemble résultant d'équations différentielles ordinaires et d'équations algébriques doit être résolu par des méthodes numériques adaptées. La simulation du système contrôlé peut être exécutée et le comportement du système de contrôle peut être examiné, comme par exemple la stabilité en boucle ouverte et fermée, la performance de contrôle, la robustesse, et ainsi de suite. Cela fournira la base pour la mise en oeuvre ultérieure du système de contrôle.

5. L'Affichage de Données : tables et graphiques sont requis pour exposer les résultats des opérations. En effet, la visualisation peut être très importante pour le développement et la détection d'erreurs quand on utilise des logiciels scientifique [HATTON 94].

I.3/ Besoins de la Mise en oeuvre

De nombreux besoins de la mise en oeuvre sont aussi communs aux besoins du développement, tel que le traitement du signal, la conception du contrôleur, et l'affichage de données (la représentation graphique de l'histoire de la dynamique des tendances fournit un moyen direct pour les opérateurs de procédé de prendre des décisions basées sur le statut opérationnel du procédé [WHITELY 92]). Outre ceux-ci, il y a encore quatre besoins importants:

1. L'acquisition de signal et le contrôle (interface des Entrées/Sorties-E/S). Dans des systèmes de contrôle, le signal peut être numérique (N) ou analogique (A), donc des convertisseurs A/N et N/A sont requis. La plupart des systèmes de contrôle de procédé travaillent en boucle fermée, donc, à chaque moment d'échantillonnage, le contrôleur impose sa sortie à l'actionneur après calcul de la loi de commande liée aux erreurs entre les consignes et les sorties et à différentes mesures sur le procédé. Donc, le contrôle basé sur un ordinateur nécessite des signaux d'entrée et de sortie. Cette tâche sera exécutée par l'interface E/S reliant l'ordinateur aux capteurs et aux actionneurs; les données peuvent être gardées en mémoire vive (RAM) ou enregistrées directement sur le disque dur. Bien sûr, l'interface E/S peut être employée pour obtenir les séquences d'entrée-sorties requises pour l'identification durant l'étape de développement.

2. L'interface homme-machine. Durant la mise en oeuvre des systèmes de contrôle de procédé, les conditions d'opération doivent souvent être contrôlées et parfois changées.
Ainsi, l'affichage en temps réel des données et la possibilité d'effectuer des modifications sont requis. Souvent, certaines conditions initiales et certains paramètres du contrôleur (par exemple l'ordre du modèle de fonction de transfert discrète, le retard entrée-sortie et les filtres) doivent être donnés ou modifiés pour améliorer la performance de contrôle. Une meilleure interface homme-machine se traduit par une opération plus facile et un système moins sujet aux erreurs de

l'utilisateur . La coopération entre l'opérateur humain et le système est cruciale pour la réussite [RUSSEL 94].

3 . Les Alarmes et la Sécurité. En réalité, des situations de procédé peuvent survenir qui sont anormales et qui doivent être traitées par l'opérateur ou par le logiciel de contrôle. Pour surveiller le procédé et détecter ces situations, des alarmes doivent être réalisées qui soient sensibles aux valeurs que certaines variables prennent et qui, en opération normale, ne doivent pas enfreindre certaines limites prédéfinies. Tout cela a pour but d'assurer la performance ou la sécurité. La sécurité est aussi souvent demandée dans des systèmes de mise en oeuvre pour empêcher l'accès au système ou refuser certaines actions à des utilisateurs non autorisés. Elle seule peut permettre, par exemple, au maître d'oeuvre de changer la température de référence que le procédé doit suivre. Les problèmes de protection et de sécurité sont bien développés dans [BURNS 92].

4. Système d'exploitation temps réel. Sa fonction est d'assurer les fonctions fondamentales de E/S (telles que l'échantillonnage) avec la précision désirée et de garantir les priorités d'évènements différents (tels que les alarmes). Les sorties y(t) du procédé doivent être acquises à des intervalles de temps précis et immédiatement les nouvelles sorties u(t) du contrôleur doivent être calculées et envoyées au procédé. Cela ne peut pas être interrompu ou retardé par d'autres événements tels que l'affichage de données, par le traitement d'autres programmes travaillant concurremment ou par des procédures de sauvegarde ou d'entretien de l'ordinateur.

L'interface homme-machine et l'affichage de données sont des problèmes complémentaires mais différents, mais ce sont les deux éléments essentiels pour l'application satisfaisante d'un procédé chimique [BACKSH 94]. Sa sensibilité est directement associée aux caractéristiques temps réel du logiciel et de l'équipement choisi. Ces deux facteurs, l'interface et le temps réel, sont les points critiques dans le système réalisé.

I.4/ Des Outils de Développement Existants.

Les facteurs principaux qu'un développeur de système de contrôle doit considérer quand il choisit le système à utiliser seront au nombre de deux: les aptitudes de l'équipement de l'ordinateur (puissance de calcul et interface de E/S) et les outils disponibles sur cet ordinateur (systèmes d'exploitation et logiciels).

Sachant, par exemple, que la simulation de procédé nécessitera le traitement de plusieurs matrices de grande dimension, si plusieurs machines sont disponibles avec des vitesses de calcul différentes, naturellement la machine la plus rapide est préférable. Si toutes les machines emploient le même système d'exploitation avec le même logiciel, alors le choix sera basé seulement sur la vitesse (de calcul et/ou d'affichage) et sur le prix. Si pour des ordinateurs différents, on dispose aussi d'outils différents, alors ceux-ci doivent être pris en compte.

Les outils logiciels pour le développement du contrôle peuvent être partagés en deux grandes catégories: les compilateurs de langage généraux et les logiciels spécifiques pour les systèmes de contrôle. Quand on travaille avec des compilateurs, il n'y a pratiquement pas de limites pour le travail fait et tout ordinateur possédera au moins un compilateur (habituellement Fortran, C, Pascal ou Basic). L'inconvénient de cette approche est d'avoir toujours à repartir d'éléments très basiques si des bibliothèques statistiques, mathématiques et graphiques adéquates ne sont pas disponibles.

La deuxième approche fait appel à des logiciels spécifiques de système de contrôle. Ils peuvent être des compilateurs de langages spécialisés adaptés aux signaux, au contrôle et à l'analyse de systèmes. Ils peuvent inclure une interface spéciale qui allège le développement de systèmes complexes employés en contrôle. Quand des interfaces graphiques sont réalisées, habituellement elles emploient le concept de bloc qui consiste à interconnecter des blocs avec des fonctions simples pour construire des systèmes complexes. La puissance et l'efficacité de ces solutions consiste à accélérer le travail du développeur pour des opérations communément employées: par exemple, afficher le graphique de la sortie 'S' d'un procédé simulé peut être aussi facile qu'écrire

'PLOT S' ou entrer dans un menu, sélectionnant l'option PLOT, et choisir alors 'S' des signaux disponibles. Cette approche n'est pas aussi flexible qu'écrire son propre code en utilisant FORTRAN ou C parce qu'il y a la restriction liée aux caractéristiques spécifiques du logiciel.

Enfin, on ne peut pas terminer ce sujet sans se référer à la tendance principale des outils de développement de systèmes de contrôle: obtenir la meilleure des deux solutions. Un logiciel de système de contrôle doit posséder la possibilité de relier des procédures écrites par l'utilisateur dans un langage général. Les avantages sont directs: accélérer le travail sur des points fondamentaux, permettre la réalisation de stratégies de contrôle parfois assez avancées, tout en utilisant les aptitudes intégrées au logiciel. Dans le cas où l'utilisateur a besoin de programmer des procédures uniques (par exemple, le modèle de connaissance du procédé chimique étudié), le logiciel doit les intégrer avec le reste des fonctions disponibles.

Après avoir analysé tout l'équipement (l'ordinateur de travail) et le logiciel (compilateurs de langages et systèmes spécifiques) disponibles sur le site de travail ou sur le marché, alors une décision fondamentale doit être prise concernant la ou les plates-formes qui seront employées.

Exemple d'un Outil de Développement pour les Ordinateurs d'Usage Général

Matlab (Matrix Laboratory) se décrit lui-même comme étant "L'environnement de calcul technique pour la haute performance de traitement numérique et visualisation" [MATLAB 92a]. La base fondamentale de Matlab est l'algèbre linéaire qui lui permet de "résoudre les problèmes avec des formulations matricielles qui interviennent dans des disciplines telles que la théorie de la commande".

Une caractéristique importante de Matlab est le support des "boîtes à outils" qui sont des bibliothèques de fonctions de Matlab qui prolongent sa fonctionnalité standard. Il existe ainsi des boîtes à outils pour le traitement du signal, pour la conception de

systèmes de contrôle [MATLAB 92b], l'identification de systèmes linéaires, l'optimisation, entre autres.

Essentiellement, Matlab est un compilateur de langage spécial incluant des variables, des fonctions, des commandes de contrôle de flux, des commandes introduites par l'utilisateur, et même des possibilités de débogage. Ses fonctions mathématiques élémentaires et spécialisées, ses fonctions de matrices, l'analyse de données et les transformations de Fourier, ses fonctions polynomiales et d'interpolation qui sont très complètes, ses sous-programmes de résolution numérique performants, sa modularité, constituent des points forts de Matlab. Un autre point fort réside dans les graphiques à deux et trois dimensions avec modèles améliorés de couleur, contrôle d'éclairage et animation.

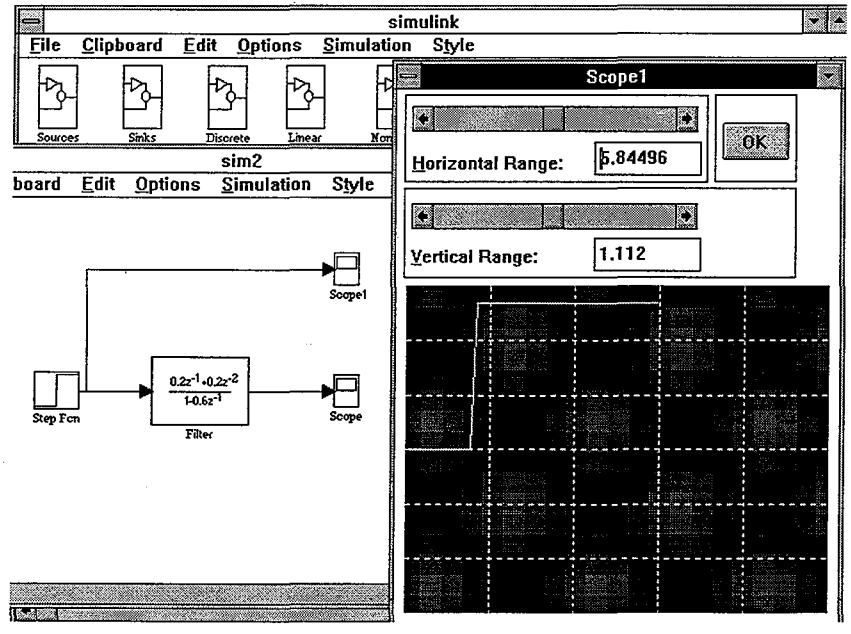


Figure 6 - Matlab et Simulink

Matlab peut être relié au programme Simulink qui réalise une interface graphique pour construire des systèmes utilisant le concept de bloc [SIMULINK 93]. Les graphismes de résultats (appelés 'Scopes') peuvent représenter un ou plusieurs signaux et être mis à jour pendant que les simulations progressent. Les fonctions externes écrites par

l'utilisateur peuvent être reliées au programme et être employées et référencées comme toute autre fonction integrée de Matlab.

I.5/ Systèmes de Mise en oeuvre Existants

Lorsque tous les aspects de contrôle ou surveillance du procédé ont été étudiés, leur application au procédé peut être envisagée.

Les Besoins d'Equipement - Réaliser un système de contrôle soit dans un petit laboratoire, soit dans une grande usine, implique:

- l'instrumentation d'acquisition et de contrôle (capteurs et actionneurs), la transmission et le conditionnement des signaux des ou vers les unités de traitement;
- l'unité de traitement où les données sont reçues, traitées et visualisées et, tôt ou tard, envoyées aux instruments de contrôle, aux unités de sauvegarde ou aux périphériques d'archivage.

Dans des procédés de grande taille où le traitement et la surveillance distribués sont employés, il n'y a aucune séparation physique claire entre l'instrumentation et le contrôle (par exemple, quand les micro-contrôleurs locaux sont intégrés aux capteurs et aux actionneurs et leurs consignes reçues d'une unité centrale de commande). Quand c'est le cas, notre attention sera concentrée sur l'unité centrale de traitement dont les caractéristiques sont les facteurs importants quand on choisit la réalisation d'un contrôle de procédé.

Besoins en Logiciels - Si l'unité de traitement n'est pas programmable, alors cela implique le choix d'une stratégie de contrôle (par exemple, un PID numérique possède déjà un programme integré). Etant l'unité de traitement réalisée par l'ordinateur, alors le logiciel doit être choisi selon les besoins du contrôle.

Le choix de la mise en oeuvre d'un système de contrôle dépend de la complexité de l'équipement dont le procédé a besoin, le degré de développement postérieur que le contrôleur aura, et si le système de contrôle sera utilisé dans d'autres applications différentes à l'avenir. Quand une mise en oeuvre de contrôle doit être employée de manière définitive dans le même procédé et que des changements dans les stratégies de contrôle ne sont pas envisagés, alors un équipement avec logiciel permanent peut être adéquat. Ce peut être le cas d'une ligne de production d'usine, rarement celui d'une installation pilote ou de recherche.

Une caractéristique importante requise par le logiciel est son comportement temps réel, s'il peut satisfaire les besoins de mise en oeuvre décrits auparavant. De façon simplifiée, on peut dire que le temps réel est l'aptitude à exécuter certaines procédures critiques avec un retard maximum connu par rapport au temps désiré. Par exemple, la précision avec laquelle les données sont recueillies est cruciale pour une analyse correcte. Egalement, il doit y avoir une garantie d'un retard maximal entre la détection d'une situation d'alarme et les procédures d'urgence correspondantes. Le temps réel étant une exigence du logiciel dans le contrôle, il dépend directement de l'équipement utilisé.

Trois types d'installations de commande ou de surveillance peuvent être considérés:

1. Les Solutions Matérielles - Elles consistent habituellement en des composants industriels standards montés dans un panneau de contrôle près du procédé ou dans une salle de contrôle, possédant comme unité de traitement un ou plusieurs micro-contrôleurs. Cette sorte d'approche est employée essentiellement dans de petites installations, le PID étant le mode de contrôle habituellement appliqué. Le micro-contrôleur peut exécuter une boucle ordinaire de contrôle après introduction de certains paramètres, ou il peut réaliser un algorithme très spécifique. Ces paramètres, ou l'algorithme spécifique, proviennent du développement du contrôle qui

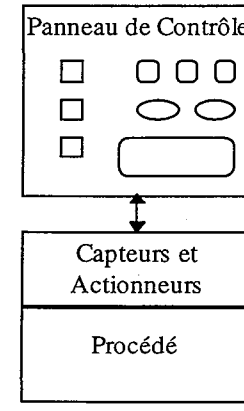


Figure 7 - Les Solutions Matérielles

est fait pour le procédé. Tout changement de l'algorithme de contrôle implique le transfert du programme vers le micro-contrôleur, le changement d'un composant d'équipement (par exemple,. une mémoire fixe type ROM) ou même le remplacement de la carte entière avec le micro-contrôleur. Les avantages de cette solution sont sa robustesse et son prix. Ces systèmes sont aussi appelés des systèmes embarqués quoique ce terme ne soit pas employé dans ce travail, car il peut avoir aussi d'autres interprétations [LEVESON 90].

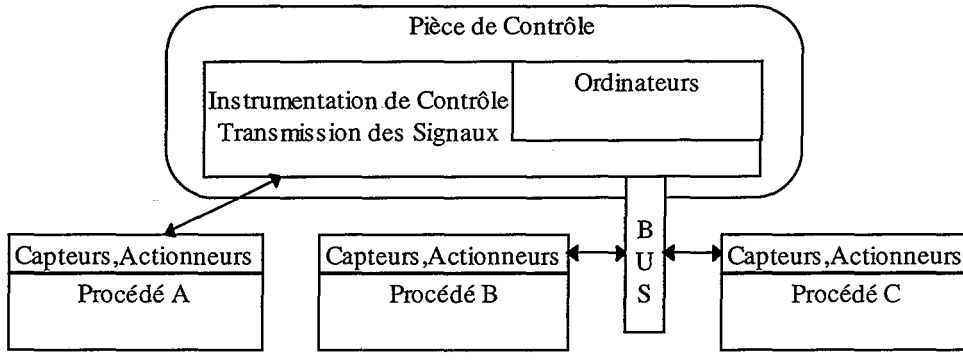


Figure 8 -Plate-Formes Spécialisées de Contrôle

2. Les Plate-Formes Spécialisées de Contrôle - Ce sont des ordinateurs spécifiques utilisant des logiciels propriétaires optimisés pour le contrôle et pour l'acquisition de données, intégrés avec de l'équipement spécialisé de contrôle et d'acquisition. Ils sont employés pour les procédés de moyenne et grande échelle où la tolérance aux pannes est le souci principal. Ces solutions peuvent contrôler beaucoup de procédés simultanément. A la différence de la Solution Matérielle, les utilisateurs de ces systèmes auront à acquérir une grande somme de préparation et de formation, parfois sur le site du fabricant. Ces systèmes sont très fermés dans le sens où tout le développement et la mise en oeuvre sont faits par une même compagnie pour leur client. Celui-ci a peu ou aucune connaissance du fonctionnement intérieur du système de commande et a des possibilités restreintes de changer les spécifications en dehors des accords contractuels. Les normes dans ce domaine concernent seulement l'interface opérateur (synoptiques, alarmes, etc.) tandis que le système d'exploitation d'ordinateur, son logiciel et même l'instrumentation et la transmission du signal peuvent être spécifiques au fabricant. Les prix de ces systèmes sont élevés et la mise

en oeuvre de nouvelles stratégies de commande est difficile (ils sont employés sur des procédés très stables essentiellement dans les lignes de production des usines et dans les procédés où la sécurité est critique tels que les centrales électriques).

3. Les Ordinateurs à Usage Général (OUG) - Ils représentent la solution la plus flexible et aussi la plus problématique. Faire appel en effet à l'une des deux solutions précédentes implique d'utiliser des outils de l'acquisition et du contrôle pour réaliser de l'acquisition et du contrôle. Dans le cas d'Ordinateurs à Usage Général, il faut ajouter à l'ordinateur des équipements pour lire les signaux provenant de l'instrumentation (en utilisant des convertisseurs analogique/numériques et des entrées numériques) et pour activer les actionneurs du procédé (en utilisant des convertisseurs numérique/analogiques et des sorties numériques). Il existe essentiellement deux manières de faire cela: en insérant une carte d'interface dans le bus interne de l'ordinateur ou en reliant un système de conversion aux possibilités d'entrées et de sorties de l'ordinateur (ports parallèles ou séries, réseaux d'ordinateur ou industrieles, etc.).

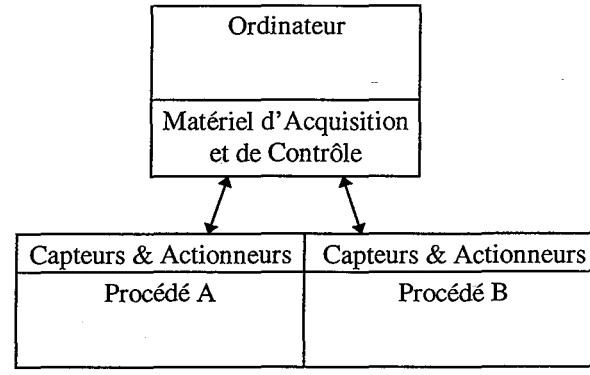


Figure 9 - Les Ordinateurs à Usage Général

Il faut aussi avoir la garantie que le logiciel peut correspondre aux exigences du contrôle de procédé travaillant essentiellement en temps réel. Ce n'est pas toujours facile à cause du système d'exploitation de l'ordinateur qui peut ne pas réaliser le temps réel ou à cause des caractéristiques du logiciel disponible ayant seulement certaines capacités de travailler en temps réel.

Le choix d'une solution OUG est essentiellement motivé par:

- un bon rapport performance/prix dû aux coûts peu élevés de ces ordinateurs,
- l'emploi des ordinateurs disponibles pour contrôler des procédés sur des périodes courtes,

- une grande disponibilité de logiciels notamment ceux pour le contrôle,
- une grande flexibilité pour le changement des algorithmes de commande,
- le fait de ne pas être tributaire d'un seul fournisseur d'équipement et de logiciel

Les inconvénients de l'utilisation des OUG sont:

- les problèmes pour harmoniser le fonctionnement de tous les composants,
- l'équipement n'offrant aucune protection de tolérance aux pannes spécifiques à l'acquisition et aux mises en oeuvre de contrôle,
- le système d'exploitation non optimisé ou non adapté au temps réel,
- le soin nécessité par le choix de l'équipement qui va travailler dans un environnement industriel (avec bruit électromagnétique, poussière, fuites, etc.)

On peut rencontrer des procédés où deux ou trois sortes de mises en oeuvre de contrôle sont présentes, par exemple, quand des stratégies distribuées et hiérarchiques sont réalisées ou quand la protection contre les pannes est exécutée en utilisant des technologies différentes (la duplication d'équipement est une autre solution commune mais moins fiable).

Dans les usines, des réseaux industriels sont souvent employés comme le FIP, le PROFIBUS et le SERCOS [CENA 95] qui ont des caractéristiques électriques et logiques permettant la communication entre toutes sortes d'instrumentation. Cela peut aller des capteurs et des actionneurs intelligents jusqu'aux ordinateurs. Cela rend plus facile la réalisation du contrôle distribué, la duplication d'instruments et d'autres caractéristiques avancées. Ces réseaux industriels sont assez distincts des réseaux d'ordinateur [THOMESSE 93] bien qu'ils puissent parfois aussi être employés tels quels.

Exemple d'Un Outil de Mise en oeuvre pour Ordinateurs d'Usage Général

LABTECH/Control est un produit de Laboratories Technologies pour réaliser le contrôle et l'acquisition de données sur des ordinateurs personnels. Une description de la version de Windows sera faite en prêtant une attention spéciale à ses caractéristiques telles qu'elles sont détaillées dans le manuel utilisateur [LABTECH 92].

Comme il est courant aujourd'hui, l'élément fondamental de Labtech/Control est le bloc. Celui ci peut être défini en utilisant des menus de texte ou des interactions graphiques (en utilisant ICONVIEW inclus).

Trois types de blocs sont disponibles: Numérique, Analogique, et Pulsation . Ils sont appelés 'Blocs de Sortie' parce qu'ils sont caractérisés par le mode de changement de leurs sorties selon le cas discret ou continu.

Les mises en oeuvre du contrôle de procédé sont séparées en Boucle Ouverte et en Boucle Fermée. Pour cette dernière, trois mises en oeuvre sont possibles consistant en PID classique et en des opérations d'Alarmes On/Off assez spécifiques.

Des fonctions contrôlant le procédé sont disponibles pour lire et représenter tous les signaux typiques d'un procédé comme voltage, courant, température (incluant les thermocouples et RTDs), et aussi des fonctions plus spécifiques pour la contrainte et le déplacement. Les variables peuvent être créées par des blocs de fonctions et avoir des entrées et des sorties, par exemple, un voltage lu sur une carte d'acquisition peut être traduit en la valeur physique qu'il représente.

L'affichage temps réel des données peut être créé et affiché sur des écrans différents. Ces affichages peuvent consister en graphiques, diagrammes de procédé ou images qui peuvent être importés de fichiers d'images au format BMP (ces deux derniers sont répertoriés comme les synoptiques de procédé). Chaque écran peut être amené à

représenter un sous-procédé ou un groupe spécial d'objets tels que, par exemple, un panneau des alarmes de contrôle.

La fréquence d'échantillonnage peut être aussi bien élevée de l'ordre de l'heure que aussi courte que 'la vitesse déterminée par l'équipement'. Cela peut sembler une pauvre caractérisation, mais elle est assez réaliste pour traduire l'importance qu'ont l'ordinateur et les cartes ou périphériques d'acquisition et de contrôle pour la performance du système.

Pour sauvegarder les données sur disque, trois méthodes peuvent être employées: écrire continûment les données sur le disque; écrire sur le disque quand une alarme ou un événement spécifique est déclenché; ou écrire les données choisies sur le disque sous l'ordre de l'opérateur. Dans le premier cas, toutes les données sont enregistrées avec l'inconvénient d'occuper beaucoup d'espace sur le disque. Dans les deux autres cas, seules les données importantes (ou intéressantes) sont gardées.

Un outil de développement pour intégrer les fonctions de l'utilisateur existe pour Labtech/Control, C-Icon, qui permet d'inclure dans un bloc des sous-programmes en langage C. De cette façon, si une fonction spécifique n'est pas disponible, elle peut être développée extérieurement et être intégrée dans l'environnement de Labtech/Control.

Deux programmes Labtech/Control peuvent s'exécuter simultanément dans deux ordinateurs différents échangeant/partageant des données à travers le réseau. Une supervision classique peut donc être faite avec un ordinateur exposant les données acquises sur l'autre. Cette aptitude à fonctionner en réseau n'est pas intégrée dans le logiciel Labtech/Control, mais doit être achetée séparément.

Enfin, un 'Driver-Toolkit' (Boite d'Outils des Pilotes) peut aussi être acquis pour développer la communication avec l'équipement de contrôle et d'acquisition. Le développement des pilotes est une tâche complexe non seulement à cause des problèmes intrinsèques liés au maniement des registres de l'équipement (ni langage

spécifique, ni outils de débogage ne sont disponibles), mais aussi à cause de la manière dont le lien avec Labtech/Control est assuré.

I.6/ Du Développement à la Mise en oeuvre

> "Je ne connais aucun système commercial de CAD pour les procédés (qui puisse être employé) du concept initial à l'opération finale" [BENSON 92].

Après avoir examiné les types d'outils de développement disponibles et aussi les options de mise en oeuvre qui nous sont accessibles, on va établir un pont entre les deux. Bien que dans certains cas il s'agisse d'un processus en deux étapes, d'abord le développement et ensuite la mise en oeuvre, ces deux points sont indissociables. Dans la section I.1/ , cela apparaît clairement lors de l'acquisition des données des entrées et des sorties du procédé pour réaliser l'identification. Si possible, les entrées du procédé sont générées par l'ordinateur, par exemple une SBPA, donc les sorties des actionneurs vers le procédé doivent aussi être réalisées. Une chose est certaine: quand le développement prend fin, la stratégie de commande résultante doit être réalisée sur l'équipement de contrôle utilisant les capteurs et les actionneurs qui sont reliés au procédé.

Les problèmes de transition de la théorie (développement) à la pratique industrielle (mise en oeuvre) sont essentiellement technologiques à cause du manque de systèmes et d'outils intégrés. Ils sont aussi dûs à d'autres facteurs comme la difficulté d'intégrer la robustesse dans les méthodes modernes d'automatique (dans l'opposition au traditionnel PID et aux systèmes analogiques) [LARMINAT 91].

Solutions Matérielles

N'étant pas basée sur des ordinateurs, la réalisation de la commande par un matériel est considérée comme la voie la plus difficile pour réaliser une commande avancée.

En effet, l'équipement et le logiciel sont intégrés et font habituellement partie d'un plus grand système. Ils ont comme premier but de fournir au moins le contrôle partiel du système ou du procédé auquel il est lié, surtout lorsque la commande se fait en temps réel [LEVESON 94]. Leur développement ne peut pas être réalisé sur l'équipement ciblé, mais doit être développé sur un ordinateur et transféré ensuite, par exemple, à l'EPROM d'un micro-contrôleur.

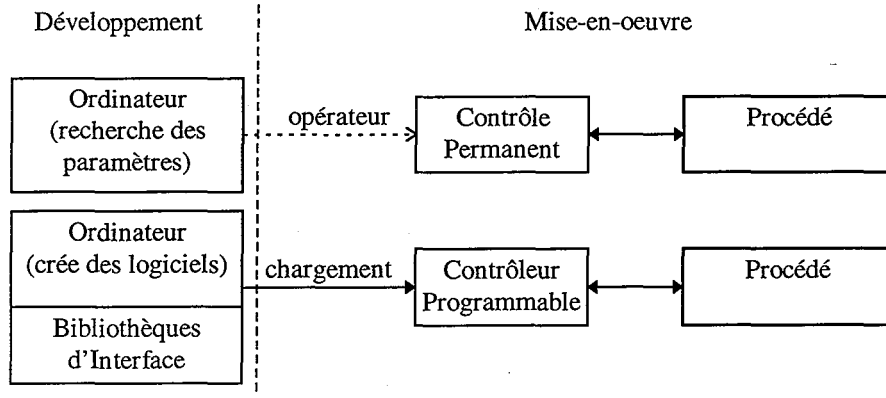


Figure 10 - Du Développement à la Mise en oeuvre (Solutions Matérielles)

Si le développement matériel est basé sur un logiciel spécifique pour le contrôle, alors trois possibilités existent:

1. Le logiciel supporte la solution d'équipement retenue, alors sa mise en oeuvre est aisée. Dès le début, la configuration équipement - logiciel - développement va à la rencontre des besoins exacts pour le procédé particulier.

2. Le logiciel peut produire le code source du contrôleur développé dans un langage pour lequel un compilateur existe pour l'équipement. L'utilisateur devra écrire tout le code requis pour employer les caractéristiques temps réel disponibles et relier les variables dans le programme aux ports entrées/sorties de l'équipement. Des bibliothèques sont habituellement disponibles chez le fabricant d'équipement pour faciliter cette tâche.

3. L'écriture de tout le code est nécessaire pour réaliser la stratégie de contrôle développée, en utilisant les langages de haut ou de bas niveau disponibles

éventuellement avec des bibliothèques de fonctions. C'est la voie la plus difficile pour mettre en oeuvre un système de commande.

En réalité, le plus souvent, l'équipement inclut déjà des types de commande et le processus de développement se borne à trouver les meilleurs paramètres de contrôle optimisés pour le procédé. Réaliser et essayer une commande avancée (par exemple, adaptative robuste) requiert une approche plus flexible et pratique.

Si le développement est fait en utilisant seulement un compilateur de langage général, alors le point 2 précédent est applicable. Même si un compilateur de même langage n'est pas disponible pour l'équipement de mise en oeuvre, il est beaucoup plus facile de réécrire le code avec un des compilateurs disponibles que d'employer un logiciel spécifique de commande n'ayant aucune compatibilité avec l'équipement.

Des Plates-Formes Spécialisées de Commande

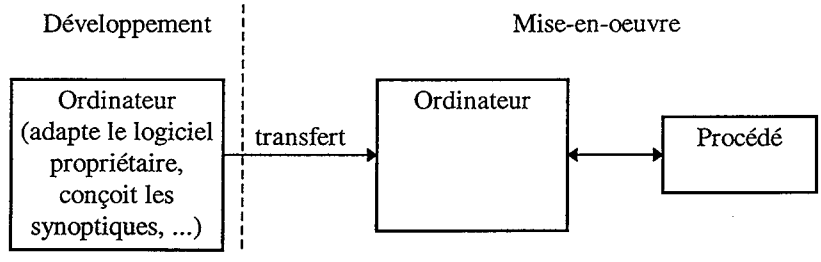


Figure 11 - Du Développement à la Mise en oeuvre (Plates-Formes Spécialisées)

L'équipement et le logiciel étant étroitement liés, si de plus tout le développement et la mise en oeuvre sont la responsabilité d'une même compagnie spécialisée, des outils existent parfaitement adaptés pour assister dans ce travail. Quand une compagnie pénètre dans ce marché, il y a deux soucis principaux. Le premier, c'est de garantir que l'équipement et le logiciel qu'elle offrira seront capables de répondre aux besoins des utilisateurs dans le segment de marché choisi (souvent un petit secteur du marché global de commande ou de surveillance de procédés). Le second, c'est d'avoir une bonne équipe responsable de faire le pont entre le client et le développement. Cette équipe, composée surtout d'ingénieurs et de technico-commerciaux, a la responsabilité

de relier les besoins de l'utilisateur aux solutions que l'entreprise offre. Les problèmes sérieux, lorsqu'ils se présentent, concernent surtout les spécifications de commande du procédé et les aspects d'ingénierie d'implémentation, rarement le cycle développement/mise en oeuvre. Si les spécifications ne peuvent pas être respectées par les systèmes de commande de la compagnie, la collaboration n'est simplement pas possible (à moins que la compagnie ne soit prête à augmenter sa part de marché en améliorant ou agrandissant ses offres de solutions d'équipement - logiciel).

Ordinateurs d'Usage Général (OUG)

La disponibilité de ces ordinateurs dans les universités, dans les laboratoires, dans les instituts de recherche et même chez les particuliers, liée à leur performances croissantes, fait que les OUG sont les plates-formes les plus employées pour l'enseignement et le développement de la commande. Les stations de travail Sun, Apollo et IBM ainsi que les ordinateurs Apple Macintosh et IBM PC et ses compatibles sont des exemples de OUG pour lesquels des logiciels de commande sont disponibles.

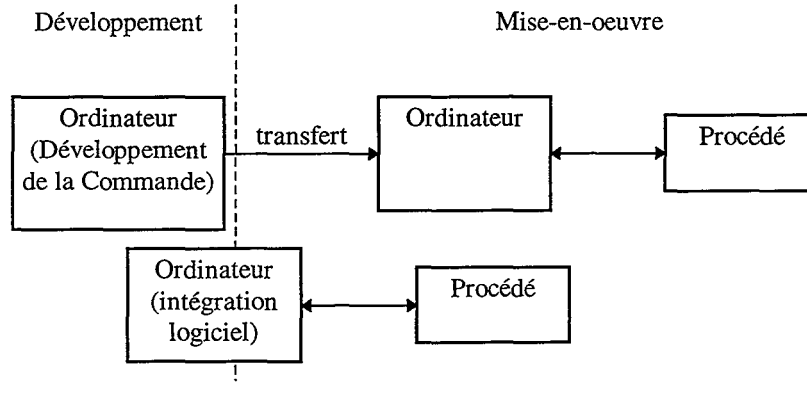


Figure 12 - Du Développement à la Mise en oeuvre (Ordinateurs d'Usage Général)

Tous ceux-ci, spécialement les PCs, ont certaines possibilités d'entrées - sorties que plusieurs fabricants d'équipement ont utilisées pour offrir des périphériques de contrôle et d'acquisition. De cette façon, beaucoup de plates-formes d'acquisition et de contrôle sont disponibles sur le marché.

Il peut paraître facile, avec un tel choix, d'aller du développement à la mise en oeuvre avec ces ordinateurs d'usage général. Hélas, en général, les choses ne se passent pas de cette manière. En choisissant l'équipement de commande et d'acquisition, un logiciel spécial doit être employé pour y intégrer la stratégie de contrôle, l'ordinateur utilisé et les périphériques de commande et d'acquisition.

Ce logiciel devra satisfaire un certain nombre d'exigences que nous allons examiner.

- les stratégies de commande nécessaires
- les possibilités temps réel requises
- la connexion à l'équipement de contrôle et d'acquisition disponible
- l'interface entre l'ordinateur et l'utilisateur

Par stratégies de commande, on entend non seulement les boucles de contrôle mais aussi les alarmes qui peuvent exister, la sauvegarde de données sur les disques de l'ordinateur, le tout associé à la surveillance et au contrôle du procédé.

Pour réaliser les possibilités temps réel, il existe trois niveaux principaux:

1) le niveau du logiciel où ses propres procédures, tirant parti des aptitudes du système d'exploitation, réalisent le temps réel;

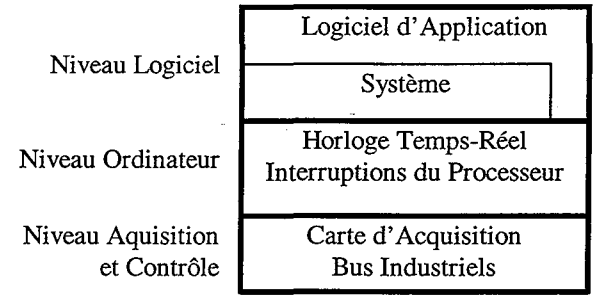


Figure 13 - Les Niveaux de Temps Réel

2) le niveau d'équipement de l'ordinateur quand le système d'exploitation n'en fait pas un emploi efficace ou complet (par exemple, le système d'horloge temps réel);

3) le niveau d'équipement de contrôle et d'acquisition, en effet, seul équipement spécifique de contrôle du système.

Parfois toutes les caractéristiques temps réel du système sont limitées au logiciel de commande temps réel et aux périphériques d'acquisition et de contrôle, ne donnant le système d'exploitation et l'ordinateur aucune contribution significative.

La connexion de l'équipement de contrôle et d'acquisition (par exemple, initier une conversion A/N ou N/A) est une tâche complexe. L'approche classique c'est de réaliser une couche logiciel, appelé un 'pilote' (en anglais *driver*), reliant le logiciel à l'équipement. Cela a impliqué, pendant de nombreuses années, l'écriture pour chaque logiciel d'un pilote spécifique pour chacune des cartes d'acquisition et de contrôle. Une des caractéristiques des logiciels de mise en oeuvre concerne les pilotes fournis (ou en option) et la facilité, si elle existe, pour le technicien d'écrire son propre pilote. Cela signifiait que les choix de l'équipement d'acquisition et de contrôle et du logiciel de mise en oeuvre devaient être fait simultanément. Si le pilote requis n'était pas disponible, l'équipement (parfois réalisé "maison") ne pouvait pas être employé. Aujourd'hui des normes émergent concernant l'interface des logiciels de contrôle et les pilotes dits universels. Le but est que chaque fabricant d'équipement d'acquisition et de contrôle développe le pilote standard auquel tout logiciel respectant cette norme pourrait se connecter (comme les pilotes d'imprimantes dans Windows).

Bien qu'il y ait des exigences communes dues aux normes de l'industrie, l'interface d'opérateur est encore un des domaines les plus ouverts à l'innovation et à la différenciation des produits. Les graphiques, les synoptiques et la visualisation des alarmes sont habituellement présents, mais le degré de facilité et la courbe d'apprentissage de l'opérateur sont fondamentaux. Au début, dans les OUG, ainsi que dans les plates-formes Spécialisées de Commande, l'objectif était de reproduire toutes les Solutions Matérielles où les panneaux des instruments étaient déjà standardisés. Cela diminuerait la résistance des opérateurs à substituer en usine des équipements traditionnels de contrôle par des systèmes basés sur des ordinateurs. Mais de l'approche de contrôle et surveillance du type 'un capteur, un indicateur', aux interfaces comme le modèle de flux multi-niveaux [BUIËL 94], l'évolution a été rapide.

Que se passe t'il avec le développement logiciel en OUG ? Comme on l'a vu dans "Besoins de Développement", les mathématiques et la visualisation des données sont les soucis principaux quand on choisit les outils de logiciel. Pour réaliser les deux, les différentes approches possibles sont décrites dans "Outils de Développement". La

discussion des problèmes impliqués sera faite maintenant et va se centrer sur le passage de l'étape de développement à celle de la mise en oeuvre. Dans la Figure 5, le commutateur S1 montre que tandis que le développement emploie un procédé simulé, tôt ou tard la commande est appliquée au procédé réel. Modifier la position de ce commutateur signifie presque toujours un changement complet non seulement sur le logiciel employé mais même sur l'ordinateur. Pourquoi cela se passe-t'il?

Les deux raisons principales sont:

- le logiciel le mieux adapté aux besoins du développement n'a pas toutes les caractéristiques d'un logiciel de mise en oeuvre (par exemple, le temps réel);
- l'ordinateur choisi pour le développement est incompatible avec celui installé pour contrôler le procédé ou avec les périphériques d'acquisition et de contrôle employés.

Cette dernière raison peut être définitive à moins que le système d'exploitation des deux ordinateurs ne soit le même ou qu'une version compatible du logiciel existe pour les deux plateformes. La voie la plus facile pour affronter ce changement d'environnement d'ordinateur, c'est d'employer des compilateurs de langages universels comme Fortran ou C, comme cela a été évoqué dans "Solutions Matérielles", parce qu'ils sont disponibles pour une grande variété de plates-formes.

Notre souci principal sera de trouver pourquoi le développement et la réalisation d'un système de contrôle de procédé sur un même type d'ordinateur peuvent être une tâche aussi compliquée. Est-ce que le logiciel pour le développement de la commande a des caractéristiques si différentes des logiciels de contrôle temps réel? Quelles sont les caractéristiques communes? Où divergent elles?

En utilisant à nouveau la Figure 5, nous pouvons examiner les changements quand le commutateur S1 remplace le procédé simulé: les signaux du procédé doivent être fournis au logiciel et la sortie du contrôleur doit être appliquée au procédé, tout cela avec des chronométrages précis. Donc, le logiciel de contrôle doit avoir des caractéristiques temps réel ainsi que l'interface pour l'équipement d'acquisition et de contrôle connecté que le logiciel de développement ne nécessite pas.

Par contre, le logiciel de développement doit être capable de réaliser des sous-programmes de validation, de conduire l'optimisation du contrôle (comparer la performance des différents modèles de contrôle vus dans "Besoins de Développement") et de faire de la simulation de procédé. Ces opérations ne sont pas normalement requises dans le logiciel de mise en oeuvre (certaines stratégies de commande, comme la commande par modèle interne, ont aussi besoin de la simulation du procédé pendant la mise en oeuvre).

Alors que l'interface ordinateur - opérateur ressemble typiquement aux panneaux de contrôle des usines et que le nombre de données, de boutons, de synoptiques et de graphiques exposés sont fixes et préalablement établis, ce n'est pas le cas avec l'interface ordinateur - chercheur de commande. Pendant qu'il améliore, compare et essaie des stratégies de contrôle, le nombre et le type de données importantes varient beaucoup. Davantage de graphiques doivent être exposés simultanément sur l'écran, ainsi que des tables où des données (paramètres) peuvent être examinées et changées. L'interface ordinateur - opérateur a retenu particulièrement l'attention des programmeurs de logiciel, parce que les opérateurs sont moins qualifiés et aussi pour des raisons commerciales (lors de la vente d'une solution spécifique de contrôle, c'est ce que le client verra et emploiera). En effet, les interfaces ordinateur - développeur sont celles qui ont les plus grandes exigences et où des améliorations peuvent amener à des gains plus substantiels de productivité.

I.7/ Paradigmes de Programmation - L'Approche Orientée Objet

Le génie logiciel est la science qui traite des objectifs, des moyens, et des problèmes rencontrés en développant des logiciels. Elle a évolué en créant des nouveaux paradigmes de programmation et en conséquence de nouveaux outils.

Un paradigme de programmation, dans le sens employé dans le génie logiciel, est la philosophie appliquée aux problèmes réels dans le monde afin de développer la

solution logicielle adéquate [BUDD 91]. Le paradigme peut donner plus ou moins d'importance à certains buts de développement des logiciels [BELLE 92], tels que:

- recherche des besoins des utilisateurs,
- bas coût de production,
- haute performance,
- transportabilité,
- bas coût d'entretien,
- haute fiabilité,
- ponctualité de livraison.

Pour répondre à ces problèmes, il existe beaucoup de techniques qui traitent du cycle de vie du développement logiciel, établissant des exigences et des spécifications, concevant, réalisant, mettant en oeuvre et réalisant la maintenance. Le paradigme de programmation est important au vu de ces problèmes parce qu'il fait le pont entre le cycle de développement du logiciel et la résolution des problèmes réels.

Programmation Fonctionnelle

Paradigme de programmation le plus répandu, la programmation fonctionnelle est basée sur la division du problème qu'on traite en plus petits problèmes plus abordables. A chacun de ces petits problèmes, une fonction peut être réalisée, ou davantage de subdivisions peuvent être faites en problèmes encore plus petits. En définitive, chaque aspect de la tâche originale est séparé en fonctions ayant une structure hiérarchique (Figure 14).

La mise en oeuvre des fonctions dépend du langage de programmation employé.

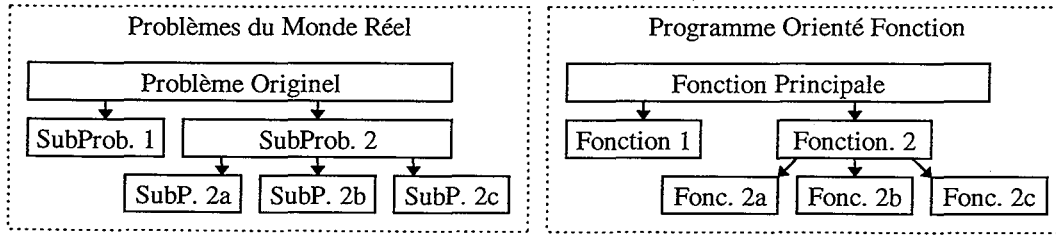


Figure 14 - La Programmation Basée sur le Paradigme Fonctionnel

Dans des langages procéduriels comme Pascal, Fortran et C, des fonctions (ou des procédures) sont réalisées comme une séquence de pas dans le but de calculer un résultat. Ces fonctions peuvent avoir des paramètres qui garantissent le flux de données dans le programme (les flux de données peuvent être vus comme la liaison de plusieurs fonctions afin d'obtenir un certain résultat)

Le développement des fonctions peut être isolé séparément. Les données et leur format en entrée et sortie d'une fonction doivent être définis et les algorithmes pour résoudre le problème respectif peuvent alors être programmés.

Le contrôle du programme est de la responsabilité des programmeurs ainsi que le partage de données entre les fonctions. Les seules tâches que le compilateur de langage doit faire sont de relier ensemble les fonctions entre elles et de réserver de l'espace pour les variables.

Les langages procéduriels, comme le C, peuvent être employés pour réaliser d'autres paradigmes de programmation. Cela n'est pas une bonne pratique parce que la vérification de la syntaxe et des types sont étroitement liés à une philosophie spécifique de programmation.

Programmation Logique

Tandis que les langages fonctionnels imposent une approche algorithmique à la construction des programmes, la programmation logique traite seulement de la description du problème. Autrement dit, au lieu de programmer la solution du problème, seules sont énoncées les règles qui s'appliquent au problème. Alors, c'est l'interprète de langage qui contrôle l'exécution du programme appliquant la logique (sous la forme de règles) pour résoudre les problèmes spécifiques.

Cette séparation entre logique et contrôle est la force principale de la programmation logique. Le programmeur doit seulement traduire en règles les lois (ou les contraintes) liées au problème et, en cas de mauvais résultats, seule la logique doit

être vérifiée. La performance, en appliquant la logique (les règles), est entièrement de la responsabilité du compilateur et si les règles sont correctes, la solution sera aussi correcte.

Les règles peuvent être les suivantes:

Règle 1: La factorielle de 0 est 1
Règle 2: La factorielle de N est N * factorielle de (N-1)
Règle 3: La somme de N et M est N+M

La solution pour trouver la factorielle d'un nombre consiste en l'application successive des règles 1 à 3 jusqu'à ce qu'une solution puisse être trouvée (plus aucune autre règle ne pourra être appliquée qui puisse changer le résultat).

Supposons qu'on veuille trouver la factorielle de N=2:
la règle 1 ne peut pas être appliquée; la règle 2 est appliquée et le résultat est que la factorielle de 2 est 2 * (factorielle de 1); la règle 3 ne peut pas être appliquée; la règle 1 ne peut pas être appliquée; la règle 2 est appliquée et le résultat est que la factorielle de 2 c'est 2 * (1 * factorielle de 0); la règle 3 ne peut pas être appliquée; la règle 1 est appliquée et le résultat c'est que la factorielle de 2 est 2 * 1 * 1; aucune autre règle peut être appliquée donc le résultat de factorielle de 2 est 2 (le résultat de 2 * 1 * 1).

Comme on peut le vérifier par cet exemple, l'application des règles n'est pas de la responsabilité du programmeur. Créer la base de données des règles est la seule tâche pour résoudre un problème spécifique.

Programmation Orientée Objet (POO)

Le paradigme objet est né quand des objets conceptuels ou physiques du monde réel ont été métaphoriquement traduits en objets dans le domaine de la programmation. Les objets peuvent être des pièces et des outils dans une usine ou des clients dans une

entreprise. Le fait de donner aux objets dans le programme les mêmes caractéristiques et aptitudes que leurs contreparties du monde réel résulte en un logiciel beaucoup plus facile à comprendre et à employer [BELL 92].

Les caractéristiques principales des langages de POO:

1. **calcul en passant des messages** - les objets communiquent en envoyant des messages l'un à l'autre. Quand deux objets sont reliés, il doit s'établir une "conversation" entre eux pour voir si la fonctionnalité de chacun peut être utilisée par l'autre. Malgré la terminologie, envoyer un message à un objet peut être assimilé à un appel traditionnel de fonction. Les fonctions (messages) soutenues par les objets sont habituellement appelées des 'méthodes', synonyme de 'opérations'.

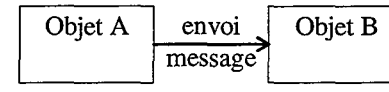


Figure 15 - Envoi de Messages

2. **l'Encapsulation des données** - l'abstraction de données est réalisée dans la POO pour que les données internes d'un objet puissent être indépendantes du comportement de l'objet. De cette façon, les données d'un objet sont seulement accessibles indirectement aux autres objets en utilisant des méthodes (messages). Faire la séparation de la fonction d'un objet de son état interne constitue une approche très puissante. Si nécessaire, les données qui structurent un objet peuvent être complètement modifiées sans avoir d'implications sur les interactions de ses méthodes (bien sûr, le code interne des méthodes de l'objet doit aussi être modifié).

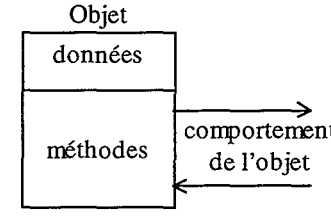


Figure 16 - Encapsulation de Données

3. **Le Polymorphisme** - une des caractéristiques cruciales de la POO est que l'interprétation du message dépend entièrement du compte de l'objet qui le reçoit. L'objet envoyant le message n'a pas besoin de connaître l'objet le recevant, ce qui rend le travail de développement

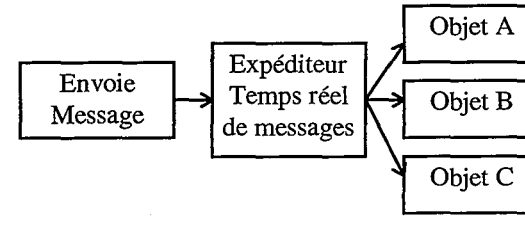


Figure 17 - Polymorphisme

plus facile. Cela tient au fait que seul un objet abstrait est employé en invoquant des objets, alors que en réalité nous appelons beaucoup de sortes d'objets. Cela résulte aussi en des programmes exécutables plus petits à cause de la liaison dynamique du sous-programme appelé (voir discussion sur ce sujet dans l'Annexe D - Programmation Orientée Objet).

4. **La Classification** - beaucoup de sortes d'objets différents ont des caractéristiques similaires (comme les armoires rouges et les armoires vertes) avec peu de différences entre eux. On dit que ces objets similaires appartiennent à une même Classe avec des variables (données) communes et des opérations (méthodes) communes. Les valeurs différentes des variables différencient les objets de la même classe. Quand un objet appartient à une Classe, il est dit être une instanciation de cette Classe. Dans le monde réel, la classification est aussi très commune en tant que manière de grouper des objets similaires.

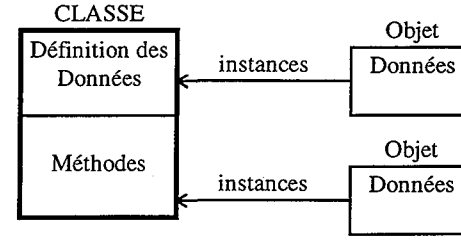


Figure 18 - Classification

5. **L'Héritage** - un objet peut avoir certains attributs d'une classe spécifique mais néanmoins avoir certaines caractéristiques uniques qui lui sont propres. Créer une toute nouvelle classe (avec des données et méthodes) pour cet objet impliquerait la duplication d'une grande part du travail déjà fait. Pour éviter cela, les Classes peuvent hériter de toutes les données et méthodes d'une Classe déjà existante, en ajoutant les nouvelles données et les nouvelles méthodes appropriées. L'Héritage possède la propriété qu'une Classe Enfant se comporte juste comme sa Classe Parent sauf dans certains cas.

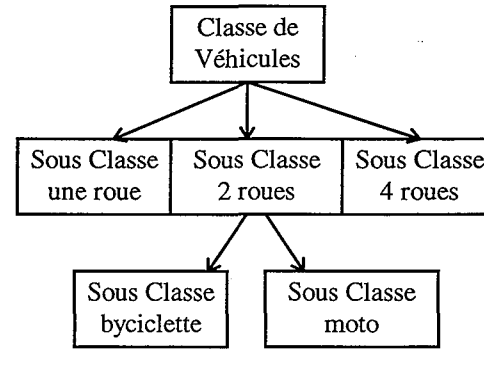


Figure 19 - Exemple d'Héritage

Tous les compilateurs de langages objet ne réalisent pas l'héritage de la même manière. Certains langages réalisent l'héritage unique (Smalltalk, Simula, Borland Pascal) tandis que d'autres réalisent l'héritage multiple (Clos, Eiffel, C++), ce qui veut dire qu'ils permettent à une classe d'hériter de une ou de plusieurs classes. Certains compilateurs de langage ont beaucoup de classes d'objets (aussi appelés 'Frameworks' en anglais) déjà développés que le programmeur peut employer en développant ses objets (en les faisant hériter de ces classes d'objet).

Le Cycle de Vie du Développement Logiciel en Utilisant la POO

Voyons comment la programmation orientée objet peut aider dans diverses phases du cycle de développement:

- **Recherche des Besoins de Utilisateurs** - comme le paradigme d'objet est le reflet étroit du monde réel, il est plus facile de répondre aux besoins des utilisateurs. En premier lieu, l'utilisateur peut être conduit à exprimer ses besoins en utilisant des objets conceptuels ou physiques. En second lieu, ces objets peuvent être facilement traduits en tant qu'objets programmés. Enfin, l'utilisateur trouvera dans la mise en oeuvre finale une reproduction proche de la manière dont il a exprimé ses besoins. De cette façon, en établissant des exigences, en fixant les spécifications, en concevant et en réalisant les programmes, la même approche d'objet avec son vocabulaire peut être utilisée: les objets, ses données, et ses opérations.
- **Bas Coût de Production** - l'héritage des classes est une méthode puissante de réutilisation des logiciels. Au lieu de programmer un objet dès le début, il peut y avoir déjà une classe d'objets qui réalise une partie de la fonctionnalité requise. Des bibliothèques de classes, incluses souvent avec le compilateur de langage, peuvent être employées pour la réalisation de tâches communes comme la gestion de fichiers, la présentation à l'écran, etc.
- **Haute Performance** - comme le logiciel est réutilisé et que des bibliothèques de classe sont souvent employées, les classes principales ont habituellement leurs méthodes optimisées. La performance dépend beaucoup de l'efficacité du mécanisme d'envoi des messages et du compilateur employé (si le programme est

compilé ou interprété, s'il peut être optimisé pour la vitesse et/ou pour la taille, etc.).

- **Transportabilité** - ici encore le compilateur de langage et son respect pour les normes sont les grands responsables pour le transfert d'un programme sur une autre machine. Les bibliothèques de classes incluses, spécialement celles traitant de l'interface de l'utilisateur, peuvent être employées avec avantage en créant des programmes portables. Si tout le développement du programmeur est fait en héritant des classes principales, alors, pour réaliser la transportabilité, seules ces classes principales doivent êtres réécrites.
- **Bas Coût d'Entretien** - le débogage logiciel et les mises à jour sont les facteurs principaux du coût de l'entretien. La programmation orientée objet excelle dans ces deux domaines parce que le test et l'amélioration peuvent être faits objet par objet. Comme les données sont encapsulées et que ce sont les méthodes qui réalisent la fonctionnalité, le débogage d'une classe d'objets est limité à l'analyse de ses propres méthodes. Les changements dans les autres classes d'objets ne peuvent pas, si une programmation correcte orientée objet est réalisée, causer d'erreurs dans une classe d'objets. Dans le même esprit, la mise à jour d'une classe d'objets peut seulement causer des problèmes dans les sous-classes qui héritent de celle-ci. Enfin, une complète reprogrammation d'une classe n'implique pas de changer les autres classes.
- **Haute Fiabilité** - En raison de la facilité à déboguer et de l'isolement entre des classes d'objets qui communiquent par des messages, les logiciels orientés objet sont très fiables. Particulièrement dans la programmation fonctionnelle, certains petits changements de dernière heure d'un programme requéraient souvent des changements du code source dans beaucoup de fonctions (pouvant même résider dans des fichiers différents). En utilisant la programmation orientée objet, cela arrive seulement lors de restructurations profondes du programme, quand des changements dans les protocoles de message sont réalisés ou quand des structures fondamentales de données du noyau du programme sont modifiées.
- **Livraison Ponctuelle** - un débogage plus facile et une fiabilité accrue raccourcissent la durée de développement des logiciels. L'abstraction plus importante des données des objets permet une division naturelle du travail de

programmation en objets. Si le protocole de messagerie est bien défini, si la classe qui sert de base est déjà développée, alors la programmation de chaque nouvelle classe est à la fois indépendante des autres classes et efficace. Efficace parce que les structures de données peuvent être définies avec toute liberté de manière à mieux s'adapter à chaque classe particulière d'objets.

CHAPITRE II - Un Logiciel Intégré pour la Commande de Procédé

II.0/ Introduction

Le besoin d'un logiciel plus unifié à la fois pour la mise en oeuvre et le développement de la commande existe dans des groupes de recherche concernés par des installations pilote et d'automatique appliquée. La stratégie et le budget conduisent habituellement à un certain investissement dans le logiciel pour la commande de procédé et très peu dans l'équipement. Cela implique d'utiliser des ordinateurs d'usage général déjà disponibles, en les équipant de cartes d'acquisition et de contrôle et en trouvant le logiciel le plus adéquat à ces plates-formes et aux besoins de recherche.

Les chercheurs utilisent généralement des compilateurs de langages populaires (Fortran, C et Pascal) et des logiciels spécifiques comme Matlab pour le développement de la commande, tandis que pour la mise en oeuvre, ils emploient des compilateurs de langages et des logiciels spécialisés pour l'acquisition et le contrôle.

Les ordinateurs employés pour le développement sont habituellement des stations de travail basées sur Unix et des IBM PC compatibles, ces derniers étant largement utilisés aussi pour la mise en oeuvre.

En raison des problèmes évoqués dans le chapitre I, "Du Développement à la Mise en oeuvre" , il est clair que beaucoup de chercheurs préfèrent encore les compilateurs de langage pour leur développement de la commande. Souvent les sous-programmes de l'interface utilisateur sont écrits de manière à permettre l'emploi des compilateurs de langages pour réaliser la mise en oeuvre du contrôle de procédé. Il en résulte de longues périodes de développement et des interfaces machine-opérateur médiocres, mais une transition efficace (pas optimale) du développement à la mise en oeuvre. Les codes sources dispersés réalisant toutes sortes de modèles de commande, des calculs numériques et statistiques, des tableaux et des graphiques, des interfaces d'entrées/sorties et même des synoptiques de procédés sont produits de cette façon.

Le logiciel spécifique de développement de la commande est surtout employé pour enseigner et former tandis que les logiciels spécifiques de mise en oeuvre du contrôle sont essentiellement employés pour les mises en oeuvre finales. Ne serait-il pas possible, en utilisant le savoir-faire existant de développement et de mise en oeuvre, de fusionner une certaine partie de ce code et de développer un logiciel complet de développement et de mise en oeuvre? La réponse affirmative à cette question a conduit au projet "ACRUN - un logiciel pour l'Acquisition et Commande".

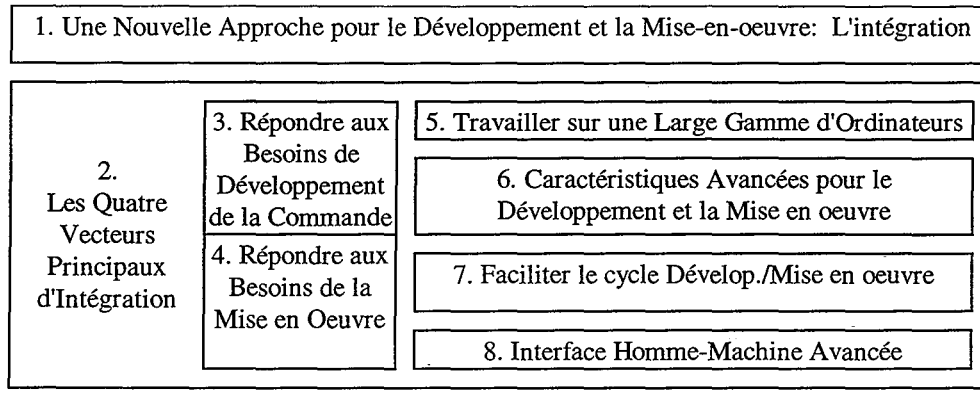


Figure 20 - Un Logiciel Intégré pour La Commande de Procédé

Les buts principaux du projet de logiciel ACRUN sont:

- réaliser l'intégration totale dans la commande de procédé,
- répondre aux besoins de développement de la commande,
- répondre aux besoins de mise en oeuvre de la commande,
- travailler sur la plus grande gamme possible d'ordinateurs,
- inclure des outils avancés de développement pour la commande,
- alléger le cycle développement/mise en oeuvre,
- avoir une interface homme - machine avancée et cohérente.

Pour le premier point, de loin le plus important, cela signifie que l'intégration totale entre les systèmes d'information et le contrôle de procédé est souhaitable (et dans l'optique de [BENSON 94], elle est aussi réalisable).

Il est important de rappeler que en tant que projet individuel de recherche, même si tous les buts devaient être atteints, il n'est pas possible, ni nécessaire de développer chaque caractéristique du logiciel jusqu'à ses limites. Il suffirait de donner un exemple de ce qui est fait pour atteindre chaque objectif particulier et comment à partir de là une autre fonctionnalité similaire peut être réalisée.

II.1/ Une Nouvelle Approche pour le Développement et la Mise en Oeuvre: L'intégration

> "Le développement des problèmes et débogage, l'entraînement, la formation, la simulation des usines, et la commande temps réel pourraient tous être inclus dans un système. Cela réduirait considérablement le temps et l'effort requis pour déplacer la technologie de l'étape de conception à sa mise en oeuvre finale dans l'usine" [MCCROSKEY 91]

Comme on l'a vu, beaucoup de logiciels actuels pour le développement de la commande sont basés sur le concept des blocs, où chaque bloc réalise une fonction spéciale et l'interconnexion de beaucoup de ces blocs peut être effectuée pour bâtir des fonctions complexes et simuler des systèmes avancés de commandes. C'est un concept très puissant et très similaire à la philosophie de programmation orientée objet.

Quand on développe un logiciel de commande et d'acquisition utilisant les pratiques classiques du génie logiciel, on traduit un monde d'objets (des blocs de calcul, des graphiques, des synoptiques, etc.) dans un langage d'ordinateur structuré séquentiel (fonctionnel). Même avec les meilleurs pratiques classiques de programmation, cela se transforme rapidement en un projet complexe, avec propension aux erreurs; il devient de plus en plus difficile de faire davantage d'améliorations ou modifications, et cela requiert une coordination et une gestion avancées s'il y a plus d'une personne développant l'application.

Tous ces blocs, graphiques et synoptiques doivent être présentés à l'utilisateur qui peut avoir besoin de les déplacer, de les redimensionner ou de les redessiner. C'est

l'exemple typique des avantages de la programmation orientée objet. Si chaque objet différent sait comment réagir aux ordres tels que 'redessine', 'redimensionne' ou 'efface', alors il n'y aura plus aucun besoin de s'inquiéter de quelle sorte d'objets existent sur l'écran. Il suffit de faire d'envoyer aux objets des ordres appropriés, sachant que chacun d'entre eux réagira correctement. Ajouter une nouvelle sorte d'objet requiert seulement l'apprentissage (programmation) de la manière dont cet objet doit réagir à certains ordres fondamentaux. Par exemple, le sous-programme pour redessiner l'écran ne sera pas changé parce qu'il continue à effacer l'arrière-plan et à envoyer à tous les objets l'ordre 'redessine'.

Le développement logiciel est intrinsèquement difficile [RUISSEAUX 87] parce qu'il n'y a pas encore de solution parfaite pour des problèmes aussi fondamentaux tel le respect des emplois du temps. L'utilisation des techniques orientées objet peut amener beaucoup d'avantages dans le développement d'un logiciel de commande et d'acquisition:

1. les outils orientés objet permettent aux ingénieurs de logiciel des améliorations de la productivité pouvant atteindre un facteur dix [MOTARD 89].

2. le 'concept des blocs' convient parfaitement au paradigme orienté objet. Chaque bloc/objet a ses propres données, sait comment les traiter, et comme les représenter et les gérer.

3. ajouter une nouvelle sorte (classe) d'objets n'implique pas le changement d'un des objets actuels ou même des sous-programmes qui gèrent ces objets. Ne pas avoir la contrainte d'ajouter des nouveaux objets/fonctionnalités, même s'ils sont rarement employés par certains utilisateurs potentiels, est important parce que cela n'entraîne pas une diminution de la performance ou un ajout de complexité au programme.

4. il permet des interfaces homme/machine avancées capables de présenter à l'utilisateur les divers objets et données typiques du développement des systèmes de commande et de leur mise en oeuvre. L'orientation objet représente la réalité dans une façon plus naturelle et intuitive [CHEN 94].

5. il diminue la complexité du programme et facilite la détection d'erreurs en isolant des sortes d'objets/fonctionnalité complètement différentes. Cela permet qu'un seul programmeur puisse développer un projet d'une dimension qui, autrement, nécessiterait l'effort de toute une équipe avec une grande habilité d'organisation.

6. il allège la création d'applications qui gèrent des grands volumes de données et de connaissance [BARTHÈS 94].

La combinaison d'environnements ouverts et de méthodes orientées objet fournit une occasion unique pour le progrès dans la conception assistée par ordinateur de système de contrôle [BARKER 95]. En utilisant des techniques orientées objet, il devrait être possible d'atteindre un grand niveau d'intégration entre le développement et la mise en oeuvre de systèmes de commande de procédé. Et encore, une nouvelle approche sera employée en intégrant de la fonctionnalité supplémentaire sous la forme de nouveaux objets qui, n'ayant aucun rôle direct dans le contrôle de procédé, peuvent largement améliorer la productivité de chercheurs et d'opérateurs d'usine. Enfin, le programme sera conçu pour qu'il puisse être aisément extensible (afin que l'utilisateur puisse ajouter de nouvelles fonctions) et ouvert aux autres logiciels (en changeant les données dans des formats ordinaires). Il s'ensuivra que l'intégration de fonctionnalité sera complétée par l'intégration dans l'environnement informatique (système opératoire et autres logiciels).

Les ordinateurs sont déjà employés dans l'industrie pour: la gestion des frais des usines, les fonctions de diffusion et d'administration, la gestion des fonctions de production des usines, les fonctions de gestion du contrôle de procédé [CORRIPIO 88]. A l'avenir, l'intégration du procédé de fabrication de l'entreprise deviendra une réalité et l'importance de l'extensibilité des logiciels [WILLIAMS 94] sera un vecteur prépondérant.

Le programme doit avoir des aptitudes multimédia, disponibles notamment pour l'opérateur dans le contrôle du procédé. Même si son emploi optimal requiert une grande étude et expérience [ALTY 95], le multimédia devrait être un attribut standard présent dans cette sorte de logiciel. Les aspects de l'interaction de l'utilisateur sont

importants [DUC 95a] donc l'interface homme-machine devrait être assez flexible pour concerner toutes sortes d'utilisateurs différents.

Il y a beaucoup d'autres avantages dans la programmation orientée objet, tel que la facilité à créer des composants de logiciel réutilisables [JOHNSON 88], qui ne vont pas être mentionnés dans ce travail car ils ne sont pas pertinents pour le sujet principal de cette thèse.

Réaliser l'intégration est l'objectif de ce travail. L'emploi des techniques orientées objet contribue à garantir la flexibilité et la robustesse du logiciel requises pour répondre aux systèmes de plus en plus complexes [BOOCH 94], et on ajoutera, de plus en plus intégrés.

II.2/ Les Quatre Vecteurs Principaux d'Intégration

L'intégration a beaucoup d'aspects différents qui, une fois assemblés, peuvent livrer tous leurs bénéfices potentiels. Du soin doit être pris pour comprendre et développer tous ces aspects simultanément, parce que tous sont importants et entre eux doit exister de l'interaction, de l'harmonie et de la cohérence.

Thomas et Neymeh [THOMAS 92] décrivent, dans ce très bon article sur l'intégration d'outils de développement de logiciel, ce que seront les quatre vecteurs d'intégration désirée. Ceux-ci ont été adaptés de cet article afin de convenir à l'intégration dans le logiciel de commande et d'acquisition.

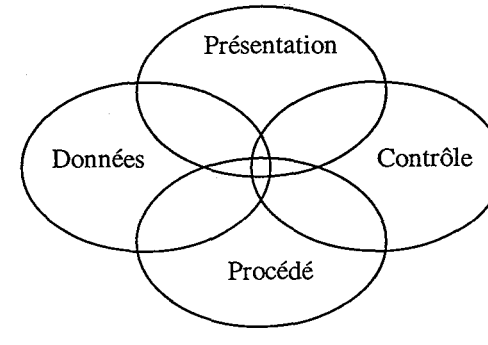


Figure 21 - Les Différents Aspects de l'Intégration

Ici seront présentées les questions principales qui doivent être constamment posées en développant ACRUN. Finalement, elles peuvent être employées comme des critères pour vérifier si le niveau d'intégration désiré été atteint.

INTEGRATION DE PRESENTATION

Apparence et Comportement - à quel point deux objets emploient-ils des apparences à l'écran et des comportements d'interaction similaires?

Paradigme D'interaction - à quel point deux objets emploient-ils des métaphores et des modèles mentaux similaires?

INTEGRATION DE PROCÉDÉ

Etape de Procédé - à quel point les objets pertinents se combinent-ils pour soutenir la performance d'une étape de procédé?

Evénement - à quel point les objets pertinents se mettent-ils d'accord sur les évènements requis pour soutenir un procédé?

La Contrainte - à quel point les objets pertinents coopèrent-ils pour appliquer une contrainte?

INTEGRATION DE DONNÉES

Interopérabilité - combien de travail est nécessaire pour qu'un objet puisse manipuler les données produites par un autre?

Non-redondance - combien de données gérées par un objet sont-elles copiées ou peuvent être dérivées des données gérées par un autre?

Consistance des Données - à quel point deux objets coopèrent-ils pour maintenir les contraintes sémantiques sur les données qu'ils manipulent?

Echange de Données - combien de travail doit-on faire pour que des données générées par un objet soit utilisables par les autres?

Synchronisation - comment un objet communique-t'il les changements qu'il fait aux valeurs de ses données?

INTEGRATION DE CONTROLE

Provision - à quel point les services des objets (outils) sont-ils utilisés par les autres objets dans l'environnement?

Utilisation - à quel point l'objet emploie-t'il les services fournis par les autres objets dans l'environnement?

L'Intégration de la Présentation peut être considérée comme la qualité de l'interface homme-machine et l'Intégration de Procédé comme la mesure de la facilité et de la fonctionnalité de combiner des objets différents afin de réaliser une certaine tâche. L'Intégration de Données sera un objectif majeur dans ce projet - encapsuler des données dans des objets communiquants - tandis que l'Intégration de Contrôle est le but primaire de ACRUN: utiliser les mêmes objets pour réaliser des fonctionnalités similaires existant dans le développement de la commande et de la mise en oeuvre finale.

II.3/ Répondre aux Besoins de Développement de la Commande

> Un puissant environnement d'usage général et de commande de procédé est requis pour soutenir le développement, les essais, et les réglages des stratégies avancées de commande [MCCROSKEY 91].

Une fois le concept de blocs adopté, le point de départ de tout le développement consiste dans la création d'objets de calcul assez flexibles pour :

- garder et traiter des valeurs réelles, valeurs entières, valeurs booléennes et du texte,
- garder et traiter des données dans des matrices de valeur réelle,
- garder tous les chiffres et combinaisons de données,
- réaliser la liaison flexible et facile entre les blocs,
- échanger aisément les données avec d'autres objets par des entrées - sorties,
- sauvegarder et récupérer des données du disque,
- afficher des données importantes sur l'écran.

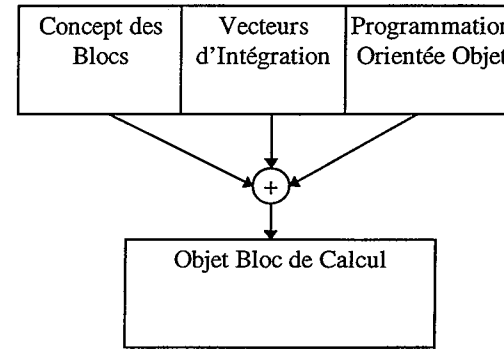


Figure 22 - Objet Bloc de Calcul

Une structure flexible de données doit être réalisée afin que le traitement du signal, les procédures d'identification, les algorithmes de commande, les simulateurs de procédé et l'affichage de données puissent tous être réalisés aisément dans ACRUN. Pour répondre à ces exigences, aux vecteurs d'intégration en question, et aux caractéristiques de la programmation orientée objet, l'architecture fondamentale consistera en différentes sortes d'objets. Chaque objet aura les données structurées (encapsulées) qui serviront mieux sa fonction et qui pourront réaliser des procédures (méthodes) générales pour communiquer avec les autres (pour plus d'information, voir le point 7 du chapitre I).

Le développement des modèles de procédé peut être un important goulot d'étranglement pour la conception et l'application des codes de traitement [MARQUARDT 92] notamment dans les industries chimiques. Ceci est dû :

- Au grand effort pour développer des nouveaux modèles,
- Au manque de réutilisation du modèle dans des projets ultérieurs,
- Au soutien inadéquat pour l'entretien et la documentation du modèle

Le premier problème peut être résolu par l'intégration des algorithmes les plus communs ainsi que les fonctions statistiques et dérivées. Davantage d'améliorations des blocs de

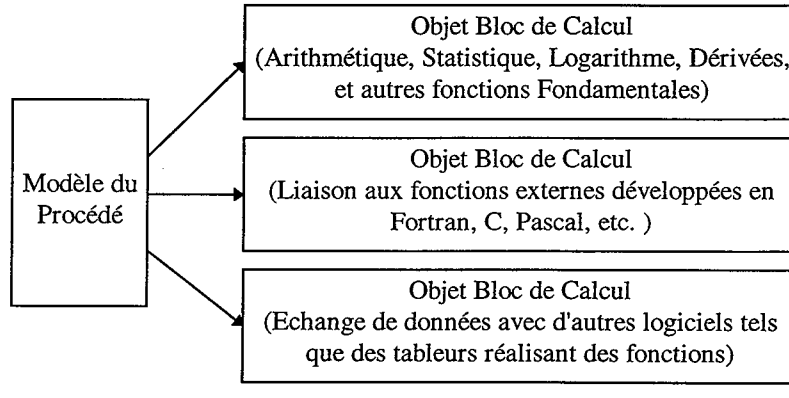


Figure 23 - Réalisation des Modèles de Procédé

calcul par les utilisateurs devraient être possibles dans le langage de programmation de leur choix ou en utilisant un autre logiciel commercial, tel que des tableurs.

Les deux derniers problèmes devraient être résolus par la possibilité de sauvegarder les algorithmes développés ainsi que leur documentation dans des bibliothèques qui pourraient être ultérieurement employées facilement et de façon ordonnée. Cela peut aussi être appliqué au développement de bibliothèques d'algorithmes de commande.

La représentation graphique des signaux est essentielle pour le développement de la commande. Les graphiques, créés comme objets, devraient lire les sorties des blocs de calcul. Donc le raccordement entre des blocs et les graphiques doit être possible et effectué d'une façon aussi simple que l'interconnexion entre les blocs eux-mêmes. L'utilisateur doit être capable de sauvegarder tout graphique incluant ses données et ultérieurement rappeler les graphiques en les comparant à d'autres (ultérieurement, ceci sera cité comme la "gestion de données").

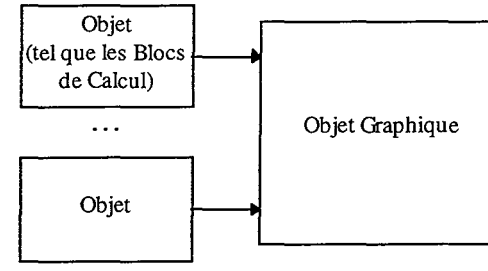


Figure 24 - Lecture des Données par l'Objet Graphique

En développant la commande, beaucoup de simulations sont exécutées pour tester les algorithmes développés. L'utilisateur doit définir la fréquence d'échantillonnage employée et combien de points seront calculés.. Le logiciel doit rendre ces paramètres disponibles et offrir un nombre d'options pour faciliter le travail des développeurs.

Tout l'environnement et les paramètres d'une simulation, les objets employés et les données résultantes, devraient tous avoir la possibilité d'être sauvegardés pour une utilisation ultérieure.

II.4/ Répondre aux Besoins de la Mise en oeuvre.

Comme cela a été établi dans le chapitre I, certains besoins de mise en oeuvre de la commande sont communs aux besoins du développement. Les blocs de calcul sont employés dans les deux cas et quelques graphiques aussi. La différence principale réside dans le fait de travailler ou non en ligne avec le procédé. Par exemple, le même graphique peut être complètement dessiné après avoir terminé le calcul de la simulation du procédé (étape de développement) ou être périodiquement actualisé après chaque échantillonnage ou si de nouvelles données du procédé deviennent disponibles.

L'acquisition de signal et sa sortie vers le procédé sont les besoin principaux dans la mise en oeuvre de la commande. Comme tels, des objets spécifiques devraient être développés pour gérer l'interface avec le procédé. Ces objets devraient lire des données du procédé, les rendre disponibles à tous les autres objets (blocs de calcul, graphiques, etc.), et devraient aussi être capable de lire des données d'autres objets et les fournir au procédé. Comme l'approche orientée objet le recommande, tout ce qui a rapport à la connexion avec le procédé devrait être concentré dans un même objet (une fonctionnalité, un objet). Par la suite, cet objet rend certains services disponibles aux autres objets.

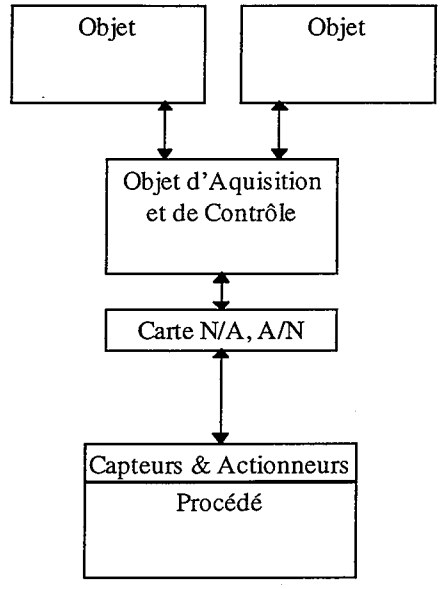


Figure 25 - Objet Interface d'Acquisition

La collecte de données est un besoin très commun dans des systèmes de mise en oeuvre où beaucoup de valeurs doivent être sauvegardées dans des fichiers sur une longue période. Dans ACRUN, un objet ayant la capacité de lire des données d'autres objets et de les sauvegarder sur le disque doit être réalisé. Comme ces fichiers peuvent atteindre des tailles importantes (actuellement, plusieurs méga-octets) et en raison de l'importance stratégique de ces données, certaines procédures auxiliaires devraient aussi être réalisées afin d'éviter la perte de données.

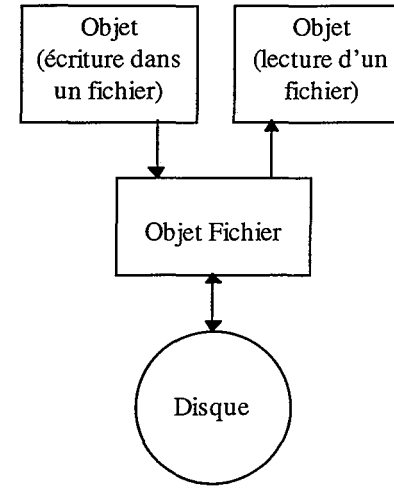


Figure 26 - Objet Fichier L'Accès aux Données sur le Disque

L'importance du contrôle de procédé dans les usines et le nombre de variables ayant des significations différentes qui doivent être surveillées, exigent une mise en oeuvre du schéma du procédé. Des objets doivent être disponibles pour représenter le procédé (équipement, soupapes, canalisations, etc.), la signification des variables et, dans certains cas, pour changer aisément une consigne ou enclencher le début/arrêt d'une action particulière.

Des alarmes doivent être réalisées afin que toute situation anormale puisse être reconnue par l'opérateur ou puisse déclencher une réponse automatique. Etant donné la grande importance des alarmes pour avertir des anomalies en empêchant les dommages, elles devraient constituer une sorte indépendante d'objet. De cette façon, leur opération peut être bien differenciée d'autres fonctions et dans des situations d'urgence, les alarmes peuvent rapidement être isolées et vérifiées. L'établissement de la hiérarchie des priorités des alarmes doit aussi être possible et une chronologie du

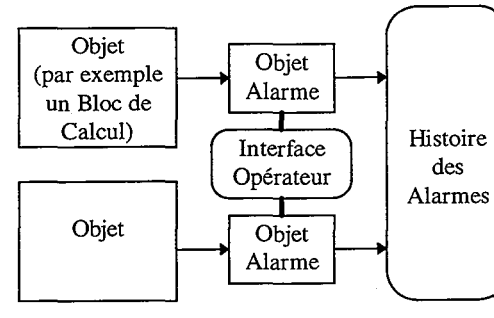


Figure 27 - Objet Alarme

déclenchement et de la constatation des alarmes doit être gardée, en repérant leurs temps de déclenchements et les retards des actions des opérateurs.

Cela nous conduit à une autre caractéristique importante du contrôle et de la surveillance des systèmes: le contrôle de l'accès. Le logiciel doit avoir la possibilité de valider une combinaison de connection/mot de passe qui puisse permettre aux utilisateurs de s'identifier et d'obtenir les prérogatives appropriées. En usine, cela s'avère indispensable parce qu'un opérateur ne peut pas, par exemple, modifier des stratégies de commande ou des valeurs de consignes qui sont sous la responsabilité du responsable de contrôle. Habituellement, les prérogatives sont basées sur le genre d'action qu'une certaine catégorie de personnel est ou n'est pas autorisée à faire. En utilisant l'orientation d'objet, une nouvelle sorte de sécurité peut être ajoutée, basée sur des prégoratives spécifiques pour chaque objet individuel.

Enfin, dans des procédés avec des chronométrages critiques et de hautes fréquences d'échantillonnage, les caractéristiques temps réel du système d'exploitation et/ou du logiciel sont très importantes. [RAVN 93] dit que la caractéristique temps réel doit être la première démarche dans la conception d'un système et sa vérification la dernière. Comme on sait que l'opération parfaite temps réel est idéale et impossible à atteindre, il est essentiel de définir tous les retards maximum garantis en répondant aux événements temps réel comme le temps correct pour commencer une acquisition ou débiter de nouvelles valeurs de commandes. Ces chronométrages dépendent de la combinaison équipement/logiciel qui est employée et c'est la caractéristique majeure à vérifier pour des systèmes de mise en oeuvre où la sécurité serait critique.

II.5/ Travailler sur une Large Gamme d'Ordinateurs

En tant que projet de recherche, ACRUN était limité à être employé sur les ordinateurs IBM PC et compatibles. Au début, ce n'était pas une option technologique, mais plutôt une approche pragmatique. Il était facile d'avoir des PCs disponibles, les accessoires pour l'acquisition et le contrôle étaient nombreux et bon marché, et il y avait déjà un

certain équipement de laboratoire relié au ports séries des PCs. Une autre question importante était l'expérience déjà acquise dans un langage orienté objet (Turbo Pascal appelé ultérieurement Borland Pascal et ensuite Delphi) disponible pour les PCs. Bien que n'appartenant pas aux langages modernes comme MODULA 3, Oberon-2, SATHER ou SELF [HENDERSON 94] ni aux langages courants dominants tels que C++ et SMALLTALK, Pascal de Borland avait néanmoins un très bon environnement de développement, des outils extensibles orientés objet et un très bon deboggueur. Pascal étant un langage fortement typé, il possède les meilleures caractéristiques pour être adapté à la programmation orientée objet [DANFORHT 88], et cela était correctement fait, de notre point de vue, par Borland.

Le système d'exploitation choisi était DOS avec l'interface graphique de Windows. Cela, couplé à la programmation orientée objet, a conduit à un développement rapide du projet. La grande disponibilité de ces systèmes faisait aussi que l'essai et l'utilisation du programme par d'autres chercheurs étaient faciles ainsi que l'emploi dans une usine de fabrication. Le début de cette thèse a coïncidé heureusement avec la sortie de la version 3 de Windows. En effet, le plus grand potentiel pour l'impact de Windows sur l'ingénierie de procédé par ordinateur repose sur l'intégration [PREECE 91], précisément l'objectif principal de ce travail.

L'inconvénient majeur était le manque d'une combinaison parfaite du matériel et du système d'exploitation avec de véritables caractéristiques temps réel. Des systèmes d'exploitation temps réel sont sortis depuis, tels que Windows NT et aussi Windows 95, mais il n'y a encore aucune nouvelle version du compilateur Borland Pascal (maintenant renommé DELPHI; la version 32 bits pour ces versions de Windows est annoncée pour bientôt) qui tire parti des nouvelles fonctionnalités.

Le choix d'un système d'exploitation comme NextStep serait le meilleur choix pour ce qui concerne l'orientation objet [MELLIER 95]. Malheureusement, ce n'était pas une alternative bon marché et populaire parce que ce système d'exploitation n'a été porté que récemment vers les IBM PCs et compatibles, où un grand choix d'équipement d'acquisition est disponible.

Les sondages de l'industrie des ordinateurs d'usage général indiquent clairement que la famille des systèmes d'exploitation Windows est et continuera à être la plus populaire.

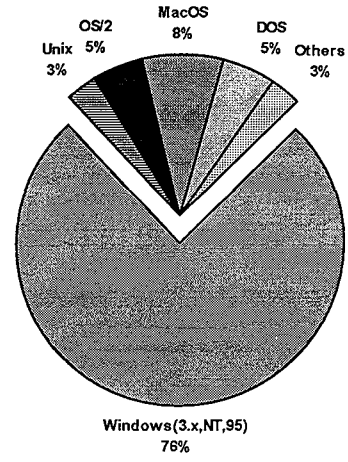


Figure 28 - Estimation du Parc Mondial de Systèmes d'Exploitation en 1995

(Etude par Dataquest)

Des PCs préparés spécialement à résister aux environnements industriels sont déjà disponibles, protégés contre les champs électromagnétiques, contre l'humidité et les vapeurs. Du point de vue technologique, Windows NT possède les caractéristiques avancées les plus importantes pour le contrôle des systèmes tels que le multitâche pré-emptif, multi-filières et un fonctionnement temps réel. Finalement, Windows NT fonctionne non seulement sur les IBM PC et compatibles, mais encore sur d'autres plates-formes d'ordinateur tel que ceux possèdant les microprocesseurs Alpha, MIPS et PowerPC. Tout cela valide le choix initial pour développer dans le système DOS/Windows.

Sur les stations de travail, qui sont largement employées par les développeurs de contrôle, un nombre de produits est déjà disponible pour faire fonctionner des programmes Windows sur Unix. Bien que ce ne soit pas la meilleure solution à cause de pénalités de performance (émulation logicielle), cela peut être intéressant dans certains cas particuliers quand, par exemple, l'équipement d'acquisition et de contrôle est attaché à une station de travail Unix.

Même si les Mips (million d'instructions par seconde) et les Mflops (million d'opérations virgule-flottante par seconde) d'un ordinateur sont une de ses caractéristiques importantes, nous pensons que les possibilités d'un ordinateur sont moins limitées par la puissance de calcul que par son aptitude à communiquer avec les utilisateurs [HARTSON 89]. Les interfaces graphiques disponibles sont donc un facteur important de choix de l'ordinateur/système d'exploitation pour lequel le logiciel est visé.

II.6/ Caractéristiques Avancées pour le Développement et la Mise en oeuvre.

Pour devenir une alternative sérieuse dans le domaine du développement de la commande, ACRUN devrait réaliser tous les outils majeurs de contrôle disponibles et offrir quelque chose en plus.

Écriture et Débogage des Pilotes des Cartes d'Acquisition	Options Avancées pour l'Affichage de Données	Utilisations Avancées de Procédures d'Alarmes (multimedia)	Simulations Automatique (mode batch).	Gestion Avancée de Données	Visualisation et Édition du Flux de Signaux

Figure 29 - Outils Avancés de Développement et de Mise en oeuvre

Du côté de la mise en oeuvre, un problème fréquent est l'écriture des pilotes des cartes d'acquisition (pour gérer les entrées et les sorties de données) insérées dans l'ordinateur. Quand on utilise un logiciel commercial, des cartes pour lesquelles des pilotes sont disponibles doivent être employées ou, dans certains cas, le pilote peut être écrit par l'utilisateur technique. Cela n'est jamais une tâche facile parce que le langage d'ordinateur peut rarement être choisi (C est presque toujours imposé) et le processus de relier le pilote au

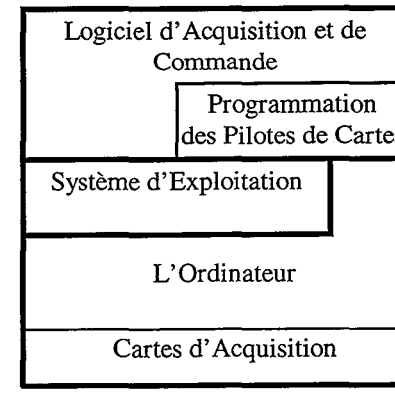


Figure 30 - Développement Intégré des Pilotes des Cartes d'Acquisition

programme requiert des outils spéciaux qui doivent être commandés séparément. Ce problème devrait être résolu par ACRUN, en intégrant la programmation du pilote et son débogage. ACRUN doit permettre à l'utilisateur d'écrire le pilote sans requérir aucun outil supplémentaire. De cette façon, quand n'importe quel nouvel équipement d'acquisition est employé, il peut être rapidement opérationnel dès l'écriture d'un nouveau pilote.

Quand l'acquisition est exécutée dans une usine pendant des mois et parfois des années sans interruption, une grande quantité de données est sauvegardée pour traitement ultérieur. Des questions comme 'quelle est la quantité de la substance A qui a été produite un certain jour?' ou 'combien de carburant doit être brûlé pour produire une certaine quantité de vapeur?' sont assez communes. En utilisant des techniques orientées objet, les données disponibles devraient aisément être accessibles par des objets communs tels que calendriers et agendas. Cela montrerait comment la puissance de l'approche objet peut permettre aux gens d'accéder aux données sous beaucoup de formes différentes et naturelles, en les intégrant au programme, sans y ajouter ni de la complexité, ni de la formation supplémentaire.

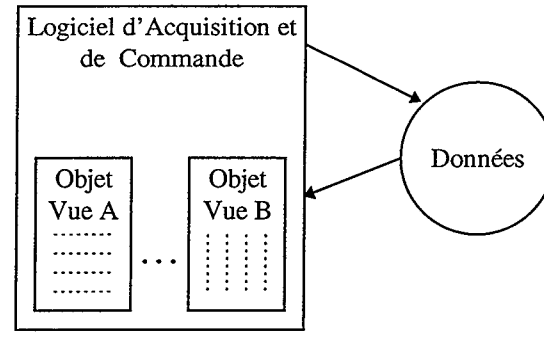


Figure 31 - Des Vues Intuitives des Données

De plus, grâce à l'approche orientée objet, des procédures d'alarme devraient être réalisées d'une façon assez avancée. Une réponse rapide de l'opérateur et la diminution des erreurs d'opération devraient être prises en compte. Cela devrait être accompli en utilisant les moyens disponibles les plus avancés (par exemple, son et images) et par une intégration complète dans la présentation d'ACRUN. Toutes ces améliorations ne devraient pas

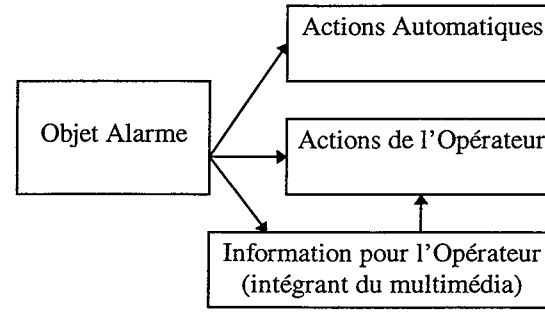


Figure 32 - Information Avancée des Alarmes

toutefois distraire l'opérateur de son rôle essentiel de surveillance des procédés.

Les fonctions du système d'alarme sont réparties en trois domaines principaux [DESPRÈS 91]:

1. les fonctions d'aide à la conduite, qui sont utilisées lors de la conduite en temps réel et servent essentiellement à présenter à l'utilisateur les alarmes utiles à un instant donné afin qu'il puisse comprendre et agir plus rapidement sans être perturbé par des alarmes inutiles.
2. les fonctions d'analyse, qui sont utilisées le plus souvent a posteriori pour étudier certains phénomènes sur le procédé, faire des corrélations, rechercher et comparer des incidents, compter des événements.
3. les fonctions de configuration qui servent à saisir l'ensemble des données gérées par le système d'alarmes. Cette base de données contient une partie des données du système de contrôle commandé, mais aussi des données nécessaires à la gestion dynamique des alarmes, qui lui sont propres.

Côté développement il faut souvent effectuer des simulations multiples, en changeant, dans chacune d'elles, un ou plusieurs paramètres. Typiquement, cela est effectué manuellement et dans certains cas, cela force l'utilisateur à attendre la fin d'une simulation avant d'en lancer une nouvelle. Des facilités de réaliser des simulations multiples de contrôle, ayant chacune un ensemble spécifique de paramètres initiaux devraient être réalisées pour automatiser le travail typique de recherche (en laissant l'ordinateur travailler plusieurs heures sans demander la présence du développeur). Le chercheur devrait être capable de définir quelles sont les données qu'il veut garder au bout de chaque simulation.

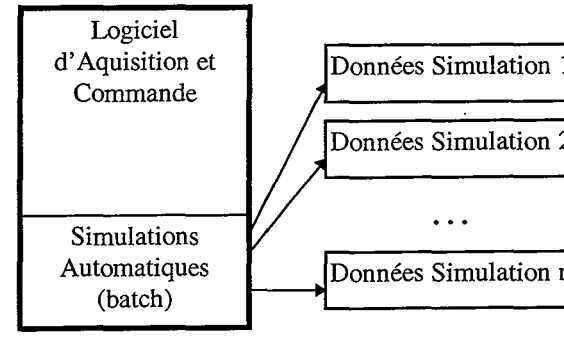


Figure 33 - Simulations Batch Intégrées

De toutes ces simulations peut résulter un grand nombre de données à analyser. Pour étudier l'effet de variations des paramètres initiaux sur une variable de sortie donnée, des graphiques en deux et de trois dimensions devraient être disponibles. La transformation des résultats des simulations multiples sous forme graphique peut amener une contribution importante au développeur de la commande et peut donner un aperçu des significations et effets des variables impliquées.

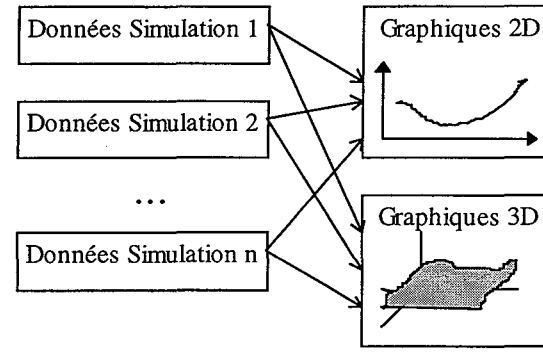


Figure 34 - Graphiques de Simulations Multiples

Même si le concept populaire de bloc est employé dans ACRUN, un langage à base de fonctions du style de Matlab et de Matrix est aussi utile et il devrait être réalisé d'une façon cohérente. Le langage de script à réaliser devrait être traduit en objets bloc de calcul et être intégré avec tous les autres objets. De cette façon, tous les outils disponibles pour manier les objets pourraient aussi être employés dans des formules entrées par le langage de script.

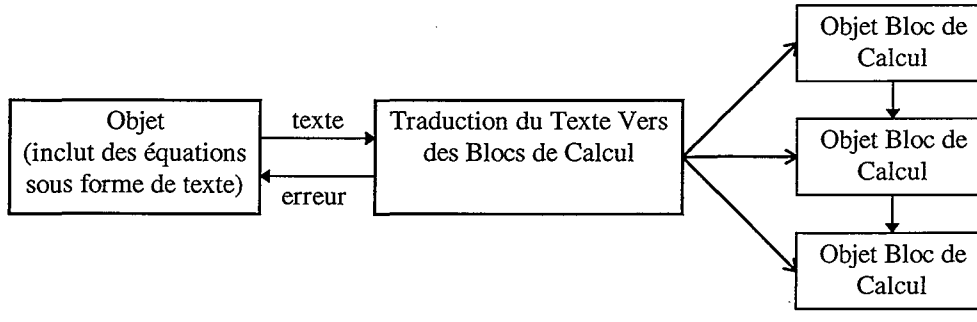


Figure 35 - Traduction des Fonctions pour des Objets Blocs de Calcul

Le flux de signaux est très important pour les développeurs de la commande à la fois pour visualiser l'interaction entre des objets et pour corriger des erreurs. Non seulement le raccordement entre des objets (blocs de calcul, graphiques, et ainsi de suite) doit être visible mais encore une procédure pour suivre aisément le flux d'un signal - entrées/sorties - devrait être mise en place. L'ordre dans lequel les blocs de calcul sont traités est aussi très important et devrait être réalisé automatiquement là où c'est possible. Dans les cas où des boucles de commande existent, par exemple, souvent la séquence de calcul établie ne correspond pas au flux de signal que l'utilisateur envisageait. Pour répondre à ces cas, l'utilisateur devrait être capable de vérifier ce qui est l'ordre de calcul courant, en le changeant à volonté, et en arrêtant la mise en ordre automatique quand elle ne répond pas à ses besoins.

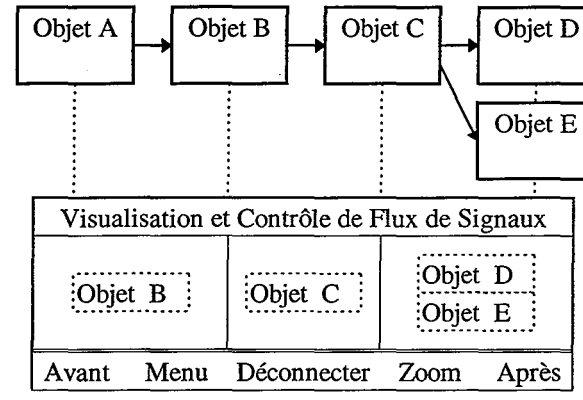


Figure 36 - Visualisation des Flux de Signaux

II.7/ Faciliter le cycle Développement/Mise en oeuvre

L'intégration est l'objectif principal du projet ACRUN. Tous les besoins communs au développement et à la mise en oeuvre doivent être réunis.

Un des problèmes majeurs est de faire cela sans compliquer à la fois le travail du développeur et le travail de l'opérateur par des fonctions inutiles à chacun (en permettant que, avec exactement le même logiciel, différentes sortes d'utilisateurs interagissent seulement avec les caractéristiques

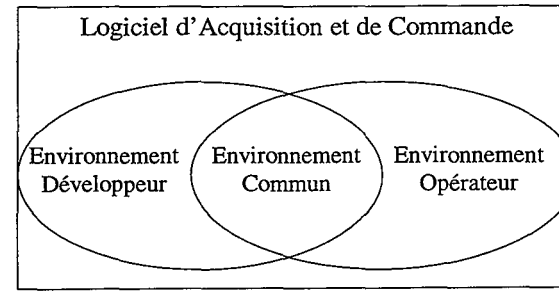


Figure 37 - Développement et Mise en oeuvre

qu'ils recherchent). Par exemple, on a vu qu'un mot-de-passe doit être réalisé avec des niveaux différents de prérogatives, afin de pouvoir être employé pour différencier les utilisateurs. De la même manière, alors que le développeur est capable de créer, déplacer et redimensionner les graphiques, l'opérateur doit être empêché de le faire. Ou encore, alors que l'opérateur n'est pas autorisé à changer les paramètres du modèle de commande, ce n'est pas habituellement le cas de l'ingénieur responsable du procédé.

Néanmoins, les tableaux et les graphiques employés par le développeur sont exactement les mêmes que ceux que l'opérateur voit. Les blocs de commande que le développeur a créés et interconnectés pour contrôler le procédé sont exactement ceux qui réaliseront la commande du procédé. Mais, tandis que le développeur est capable de les visualiser et de changer ces blocs, pour l'opérateur ils peuvent être rendu complètement invisibles, calculant dans l'arrière-plan.

ACRUN doit posséder toutes les caractéristiques de temps réel des logiciels de commande et d'acquisition qui ne sont pas réalisées dans Matlab, Matrix ou dans tout autre logiciel pour le développement des systèmes de commande et du traitement du signal. Aussi, des outils informatiques doivent exister dans ACRUN qui ne sont pas disponibles dans les logiciels spécialisés temps réel de contrôle comme Labtech/Control et Labview.

Une variable booléenne devrait être associée au fait qu'une simulation est en train d'être exécutée ou qu'une acquisition et un contrôle en ligne est exécuté avec le procédé. De cette façon, certains objets pourraient être actifs ou inactifs, des flux de signal pourraient être modifiés, des alarmes pourraient être ou non activées, conformément à ce fait. Ainsi, toute la différence en développant hors ligne ou en réalisant du contrôle en ligne sera de choisir de commencer une simulation ou l'acquisition et le contrôle. Atteindre cet objectif accélérera indubitablement le développement de contrôle à petite échelle où le procédé à contrôler est toujours disponible comme c'est surtout le cas dans les laboratoires de recherche.

Comme on l'a vu dans le chapitre I, un besoin commun en développement de la commande est l'identification de système habituellement réalisée à partir de l'enregistrement simultané des entrées - sorties d'un procédé pendant un certain temps (parfois il est possible et commode d'imposer un signal spécial à l'entrée du procédé). Alors, un certain développement de contrôle est fait loin du procédé. Enfin, la performance de la stratégie de contrôle doit être mise à l'essai en se connectant à nouveau au procédé. Si tout cela peut être fait aisément avec ACRUN, alors on pourra dire que l'intégration dans un seul produit du développement et de la mise en oeuvre de la commande de procédé était une réussite.

II.8/ Interface Homme-Machine Avancée et Cohérente

La conception de l'interface homme-ordinateur est cruciale pour bâtir fructueusement un logiciel de commande de procédé [LEVESON 90] parce que même si toute la fonctionnalité nécessaire est disponible, elle doit aussi être intuitive à employer, aisément accessible pour qu'on puisse agir sur elle avec cohérence. Personne ne pense plus à l'interface seulement comme composant supplémentaire parce que, désormais, elle est largement acceptée comme partie intégrale du système entier [NEEMLAKAVIL 90].

Par interface avancée homme-machine, on entend:

- interface graphique avec la contribution du clavier, de la souris, et de la voix,
- absence de mémorisation des ordres complexes,
- accès facile à tous les paramètres du programme et aux données des objets,
- configuration facile de l'écran par l'utilisateur,
- emploi de techniques intuitives comme glisser - lâcher quand c'est possible,
- disponibilité d'un système d'aide en ligne électronique avec hypertexte,
- intégration des capacités multimédia notamment du son et de l'animation.

Par interface homme-machine cohérente, on entend:

- l'emploi de normes habituellement rencontrées dans d'autres programmes (menus de fichiers, barreaux déroulants, actions des boutons de la souris, menus locaux, etc.),
- une approche commune pour manier tous les objets (avec le même modèle mental),
- l'adaptation à la fois au développement de la commande et à la mise en oeuvre.

ACRUN devrait employer les caractéristiques fondamentales de l'interface homme-machine fournies par le système d'exploitation et l'améliorer avec des concepts supplémentaires adaptés au contrôle de procédé et à la manipulation d'objets. L'idée est d'alléger les développeurs en gérant de grandes quantités de variables des modèles de commande (dans les afficheurs, les tableaux, les graphiques, etc.) ainsi que de réaliser des synoptiques dans l'industrie de procédé avec les normes de visualisation des opérateurs.

En vue des différentes sortes d'objets à réaliser et à employer, il devrait posséder les possibilités suivantes:

- grands écrans virtuels avec des capacités de zoom et de recherche,
- pouvoir montrer/cacher tous les objets d'une sorte particulière,
- sauvegarder et rappeler des configurations d'écrans et le positionnement des objets utilisés,
- écrans hiérarchiques où exposer les objets,
- possibilité de grouper/séparer des objets qui ont une relation entre eux.

La qualité de l'interface utilisateur est cruciale parce qu'elle est le composant clé pour intégrer toutes les caractéristiques d'ACRUN. Elle devrait être facile à appréhender (peu de concepts à maitriser) et elle doit se comporter comme un outil plutôt que comme un problème supplémentaire [ZHU 94].

CHAPITRE III - Objets d'Acquisition, de Contrôle, et Autres

III.0/ Introduction

"Toutes les abstractions de données et les choses qui sont identifiables et qui ont une signification dans le problème en question peuvent être conçues comme des objets" [RUMBAUGH 92].

Un besoin d'intégration est ressenti lors de la réalisation d'une tâche complexe et une division logique est d'en isoler des aspects différents qui sont présents. Alors, quand chacun de ces aspects est résolu, ils doivent être réunis pour effectuer la tâche initiale (intégration de procédé). Dans la planification orientée objet, une séparation est faite par rapport aux données et à la fonctionnalité. Les fonctions réalisées dans un objet peuvent néanmoins être employées pour agir sur les données d'autres objets garantissant l'intégration de données (l'interoperabilité des objets évite la redondance de données).

Les abstractions de données devraient être réalisées comme objets de même que l'analyse fonctionnelle devrait être faite pour optimiser la réutilisation des objets. Selon [WELKE 94], les applications orientées objet de l'avenir, permettront à l'utilisateur de construire un programme par la combinaison d'objets. ACRUN a déjà l'environnement pour faire cela avec plus d'une douzaine de sortes différentes d'objets et la possibilité d'en intégrer beaucoup plus (voire l'objet OLE dans la page93).

Ce chapitre consiste en la description des objets dans ACRUN et en l'analyse de la manière d'effectuer l'intégration. Les objets qui étaient développés explicitement pour l'acquisition et le contrôle sont introduits en premier. Alors, les objets qui ont une application plus générique seront présentés en montrant comment les personnes travaillant sur le développement et la mise en oeuvre du contrôle peuvent les employer.

Notre attention sera portée sur les caractéristiques et le potentiel des objets. Quand cela est possible, les liaisons qui peuvent être faites entre les objets seront décrites. Comme il est difficile d'être exhaustif sur les applications possibles des objets et de leurs

interconnexions, leurs caractéristiques majeures et leur possibilités typiques pour atteindre l'intégration des procédés seront discutées.

III.1/ Héritage des Données et des Méthodes de l'Objet Rectangle

La classe d'objet principal (racine) dans ACRUN est le Rectangle. Toutes les autres classes d'objets sont des descendantes de classe Rectangle, directement ou indirectement (comme c'est le cas de l'objet Tableau).

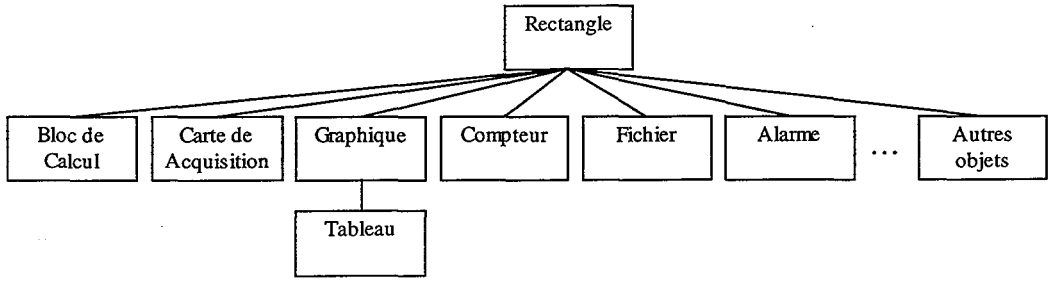


Figure 38 - Classe Objet Rectangle et ses Descendants

L'objet Rectangle renferme les données fondamentales telles que les coordonnées sur l'écran, les couleurs d'avant et d'arrière-plan, un code unique d'identification, l'information sur les prérogatives, et ainsi de suite. Il inclut aussi des méthodes fondamentales telles que pour leur affichage sur l'écran, leur sauvegarde et la récupération du disque, leur redimensionnement, etc. .

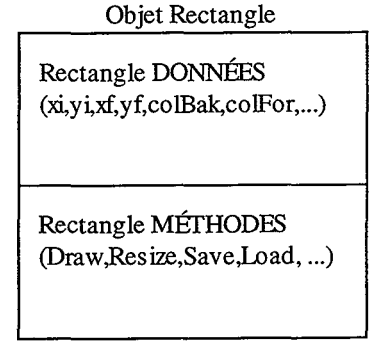


Figure 39 - Objet Rectangle

Les descendants de l'objet Rectangle héritent de ses données et de ses méthodes. Ils renferment aussi leurs propres données et leurs propres méthodes. De nouvelles variables sont habituellement requises pour contenir l'information pertinente à l'objet, mais l'accès aux données du Rectangle reste aussi possible. De nouvelles méthodes sont ajoutées pour agir sur les données de l'objet (les nouvelles et celles dont il hérite) pour que la fonctionnalité de l'objet puisse être réalisée. De nouvelles méthodes peuvent aussi être ajoutées pour remplacer celles qui sont héritées (voir Annexe D - Programmation Orientée Objet).

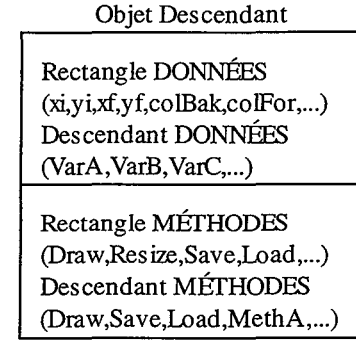


Figure 40 - Descendant Objet

Le noyau d'ACRUN travaille seulement avec des données et des méthodes héritées de l'objet Rectangle. Dans une pure mise en oeuvre de l'orientation objet, il devrait seulement traiter avec des méthodes du Rectangle (ces méthodes étant le seul accès aux données) mais pour obtenir une plus grande performance, ce n'a pas été le cas. Les méthodes du Rectangle peuvent être divisées en deux catégories: celles dont le but principal est de répondre aux besoins du noyau et celles qui peuvent être employées pour communiquer avec les autres objets.

III.2/ Propriétés Communes des Objets et leur Maniement

Parfois le terme 'objet' est employé en se référent à ses données et parfois il est employé en se référant à sa fonctionnalité. Cela est dû à la philosophie intrinsèque des objets qui contiennent simultanément les données et les procédures pour les manipuler et pour les représenter (voir section I.7/). Si l'objet A doit accéder aux données de l'objet B, il doit le faire par les méthodes de l'objet B. L'utilisateur doit être capable de modifier les données renfermées dans les objets et de définir la communication entre eux.

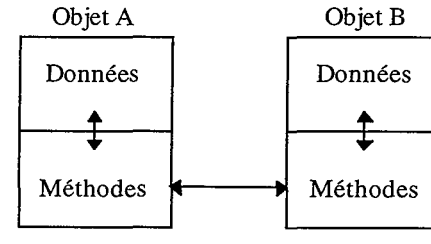


Figure 41 - Méthodes: Méthode Pour le Partage des Données

La manière ordinaire pour accéder aux données des objets est de cliquer avec le bouton droit de la souris sur l'objet exposé à l'écran. Un menu local apparaît donnant accès aux données des objets (par le 'Setup', 'Edit' et d'autres commandes) et à la fonctionnalité des objets (par exemple: exécuter ses calculs, optimiser l'échelle graphique, etc.).

La manière ordinaire d'un objet pour accéder aux données des autres objets s'obtient par le Glisser - Lâcher, c'est-à-dire, en pressant le bouton gauche de la souris au-dessus d'un objet, le déplacer et le lâcher au-dessus d'un autre objet. Les méthodes des deux objets communiqueront alors entre elles pour essayer de définir l'interaction qui peut avoir lieu entre ces objets.

Quand on parle de la 'manière ordinaire', une référence directe à l'intégration de présentation est faite. En effet, si le même paradigme d'interaction est appliqué par tous les objets, un modèle mental cohérent et unique est alors développé. Cela, à son tour, donne des incitations à l'exploration de la fonctionnalité des objets dans de nouvelles situations qui sont aussi cohérentes avec le modèle mental. Dans ACRUN, les méthodes des objets sont développées en tenant compte non seulement de l'intégration au procédé qui est à leur origine, mais encore de la possibilité d'une réutilisation ultérieure. Pour que cela soit atteint, il est essentiel que l'interface graphique d'utilisateur soit commune pour tous les objets.

Quand les connexions sont faites entre les objets et que leur données sont partagées, des problèmes de synchronisation parmi les données communes surviennent. Tous les changements d'une variable se répercutent à la totalité du programme et tous les objets concernés avec les modifications effectuées sont mis à jour, simultanément d'une façon interne et à l'écran. La partie du noyau d'ACRUN qui est responsable de la synchronisation emploie trois fonctions, 'listaDeRectInputs', 'listaDeRectOutput', et 'alterouVariavelNesteEndereco' (variables changées à cette adresse'). Ces méthodes demandent à chaque objet quels sont ses objets d'entrée (objets dont il utilise des données), quels sont ses objets de sortie (objets auxquels il fournit des données), et informent les objets qu'un changement d'une variable à une certaine adresse a été

effectué. Ces informations sont employées pour afficher sur l'écran les liaisons entre les objets, pour mettre à jour les objets quand des données pertinentes peuvent avoir changé, et pour définir la séquence de calculs à exécuter.

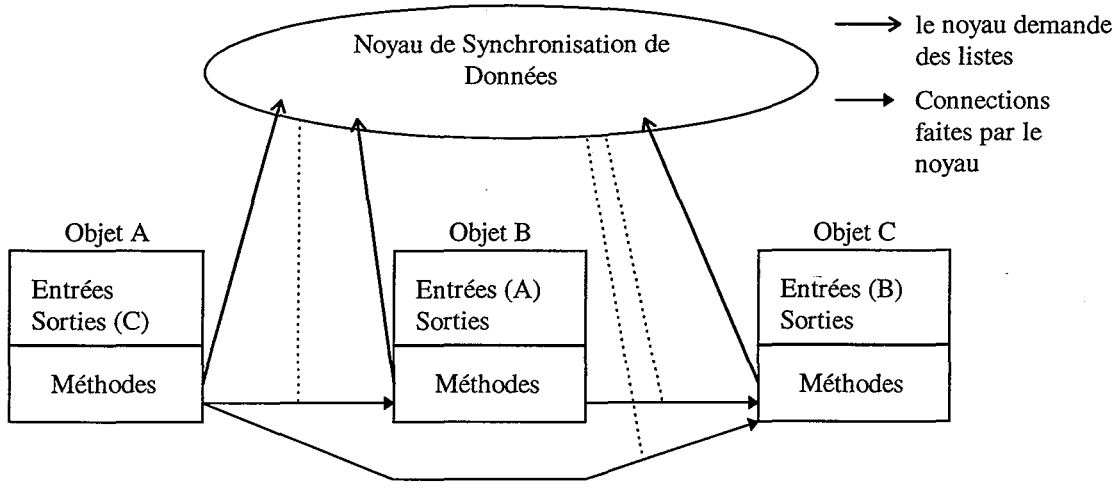


Figure 42 - Noyau de Synchronisation des Données

Les entrées et les sorties de chaque objet sont gardées dans des structures de données définies par chaque objet (Figure 42). Quand le noyau d'ACRUN interroge un objet sur ses entrées et sur ses sorties, l'objet doit répondre aux méthodes respectives et délivrer une liste consistante d'objets. Si un objet ne possède pas des entrées et des sorties, alors il n'est pas nécessaire de réaliser ces méthodes et, par défaut, il va employer celles héritées de l'objet Rectangle.

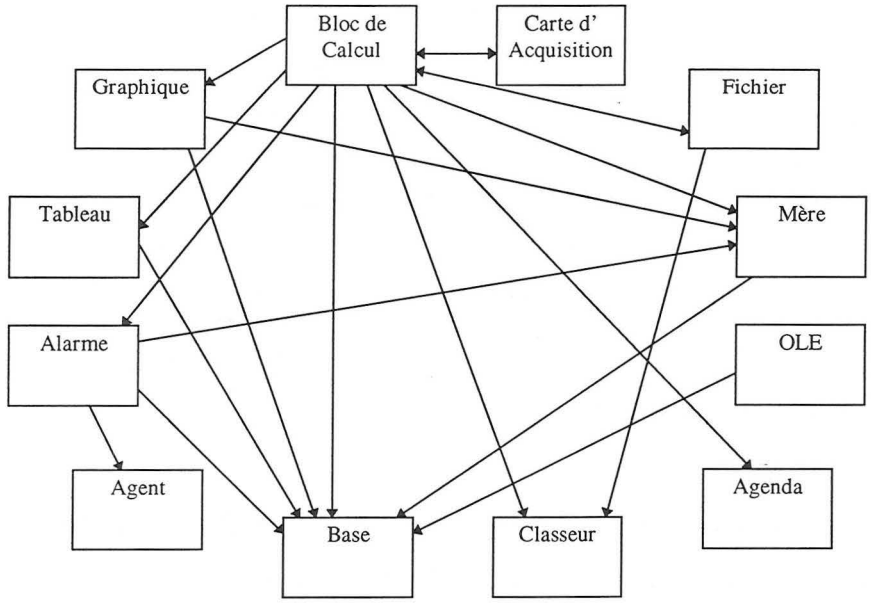
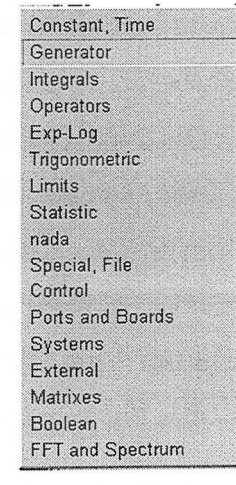


Figure 43 - Les Connexions les Plus Utilisées entre les Objets

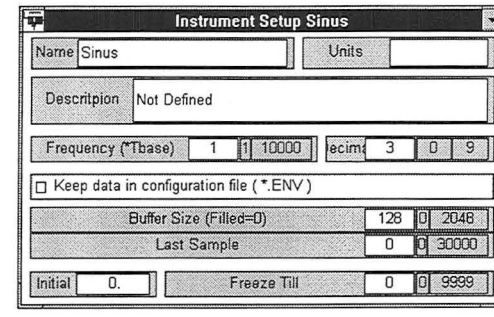
Certaines des liaisons les plus employées entre les objets sont montrées Figure 43 pour que la description des objets qui suit puisse être mieux comprise sur le plan d'intégration (interoperabilité) d'objets.

III.3/ Bloc de Calcul



Ce sont les objets responsables de la production et de conserver les données issues des simulations et de l'acquisition et contrôle. Ils réalisent les fonctions de manipulation des données qui vont de la géneration d'un signal sinusoïdal à l'exécution d'une transformation rapide de Fourier (FFT). Les Blocs de Calcul peuvent accéder aux données d'autres objets et, aussi, fournir des données pouvant être employées par d'autres objets. Le nombre d'entrées, de sorties et l'algorithme réalisé varient beaucoup pour chaque objet instancié, ainsi que le type de données qu'ils manipulent. Ces données peuvent être des valeurs réelles, des valeurs entières, des valeurs booléennes ou du texte.

La possibilité commune à tous les Blocs de Calcul est de définir:



Name (Nom) - comment l'objet doit être identifié et differencié des autres.

Units (Unités) - pour décrire la signification (physique ou autres) des données.

Description - sert comme un commentaire pour décrire davantage la fonction du bloc ou une certaine caractéristique spéciale importante à rappeler.

Frequency (* Tbase) (Fréquence)- à quels intervalles le bloc doit être calculé en relation avec la période d'échantillonnage. Si cette valeur est '1', alors le calcul aura toujours lieu à chaque période d'échantillonnage, si c'est '2' le calcul sera effectué une fois sur deux, et ainsi de suite.

Decimal (Décimal) - le nombre de décimales employées pour représenter les données gardées dans le bloc.

Keep Buffer in Configuration File (Maintenir les Données de la mémoire Tampon) - un tampon (un espace transitoire de sauvegarde) est disponible pour garder les valeurs passées des blocs. En sauvegardant le Bloc de Calcul sur le disque, les données du tampon sont aussi enregistrées si cette option est active.

Buffer Size (Taille du Tampon) - définit combien de valeurs peuvent être gardées dans le tampon. Si le tampon est rempli, les données les plus anciennes sont substituées par les plus récentes.

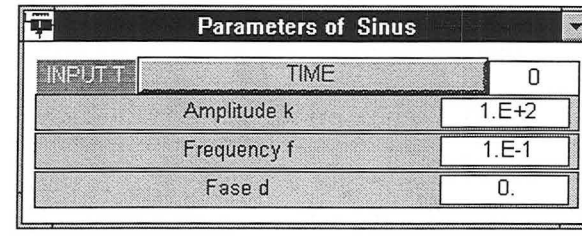
Last Sample (Dernier Echantillon) - définit ce qui est le dernier échantillon emmagasiné dans le tampon. Cette valeur est presque toujours automatiquement mise à jour par le programme.

Initial Value (Valeur Initiale) - valeur défaut emmagasinée dans le Bloc de Calcul avant que l'acquisition ou la simulation soit déclenchée.

Freeze Till (Geler Jusqu'à) - l'échantillon à partir duquel le bloc commence à réaliser sa fonction. Quand l'échantillonnage commence, il y a souvent certaines fonctions qui doivent seulement être activées après un certain retard (nombre d'échantillons).

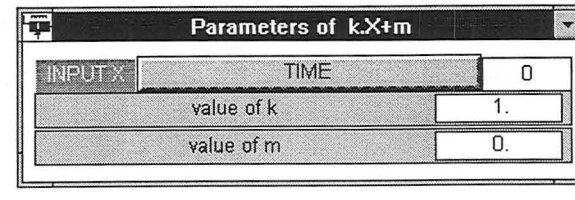
De nombreuses sortes de blocs de calcul sont disponibles. Réalisant des fonctions différentes, elles ont un nombre de paramètres et des types différents. Ci-après sont donnés deux exemples de Blocs de Calcul.

Le Bloc de Calcul SINUS - quand on utilise ce bloc, quatre paramètres peuvent être définis: l'entrée T, l'amplitude k, la fréquence f et la phase d. La sortie du Bloc de Calcul est le résultat de la fonction $k.SIN(2\pi f\, t + d)$. A chaque moment d'échantillonnage, cette fonction est calculée et son résultat rendu disponible à tout autre objet qui le demande.

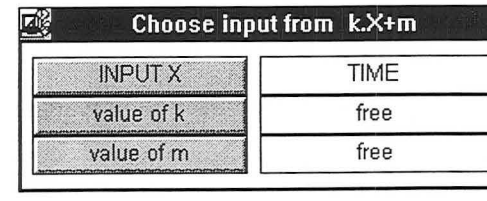


Les valeurs passées de la fonction SINUS sont emmagasinées dans son tampon si bien que les 'n' derniers échantillons de la sortie sont aussi rendus disponibles, non seulement aux autres Blocs de Calcul mais encore à tout autre objet. La valeur du nombre 'n' est celle qui est le plus petit nombre d'échantillons passés ou du paramètre de la taille du tampon. Donc, à chaque nouvel échantillon, la dernière valeur est emmagasinée dans le tampon et une nouvelle valeur est calculée utilisant la fonction du bloc.

Bloc de calcul 'k.X+m' - il y a trois paramètres dans ce bloc (k, X et m) et sa sortie est le résultat du calcul de la fonction k.X+m. Par défaut 'k' est égal à 1, 'X' est le temps et 'm' égal à 0. En éditant les paramètres, 'k' et 'm' peuvent être changés à toute valeur réelle. 'X' peut être lié à un autre bloc de calcul en utilisant un menu établissant ainsi une connexion entre les deux blocs.



Une autre façon d'interconnecter les blocs de calcul utilise la technique déjà décrite de glisser - lâcher. En connectant le Bloc

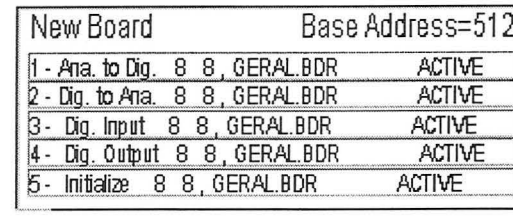


'SINUS' au Bloc 'k.X+m', la sortie du SINUS pourra être non seulement l'entrée du paramètre 'X' mais aussi l'entrée de 'k' ou de 'm'.

Cela donne aux fonctions réalisées par les Blocs de Calcul une plus grande flexibilité et puissance parce que tout paramètre peut avoir comme entrée la valeur d'un autre Bloc ou d'un autre objet. Quand un paramètre lit l'entrée d'un autre Bloc, un retard peut être défini pour que soit utilisé un échantillon passé de l'entrée. Par exemple, si dans 'k.X+m' les paramètres sont définis comme 'k'=1, 'X'=SINUS avec un retard de 1 et 'm'=0, alors la sortie du bloc sera le signal Sinus décalé d'une période d'échantillonnage. Il faut faire attention au fait que si la sortie d'un bloc est requise avec un retard N alors le tampon de ce bloc doit garder au moins les N dernières valeurs de sa sortie (Taille de Tampon > = N).

Pour une liste complète des Blocs de Calcul existants voir Annexe A - Fonctions Réalisées dans les Blocs de Calcul.

III.4/ Cartes d'Acquisition

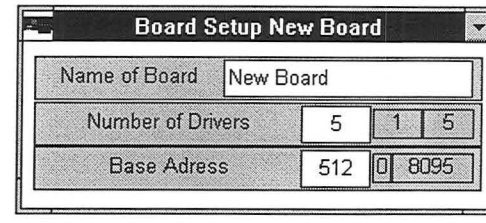


Comme évoqué au chapitre I, les ordinateurs d'usage général peuvent être étendus avec de l'équipement d'acquisition et de contrôle pour se connecter au procédé. Pour simplifier, on utilisera le terme générique de carte d'acquisition même si certaines parmi elles ont aussi, ou en exclusif, des aptitudes de contrôle (envoyer des sorties vers le procédé). La plupart des cartes d'acquisition disponibles sont normalement insérées sur le bus d'expansion interne de l'ordinateur.

Pour se connecter avec ces cartes, un nouvel objet a été développé dans ACRUN: l'objet Carte d'Acquisition, cité quelquefois simplement comme l'objet 'Carte'. L'objectif est de concentrer dans cet objet toute la complexité intrinsèque de la connexion avec l'équipement. Les autres objets auraient alors l'accès aux valeurs lues de l'acquisition, simplement en communiquant avec l'objet Carte correspondant

responsable de la gestion de la vraie carte d'acquisition. Il en serait de même pour l'écriture des données vers la carte de contrôle, ceci étant réalisé par ce même objet Carte.

La possibilité commune à tout objet Carte d'Acquisition est de définir:



Name of Board (Nom de la Carte) - comment l'objet doit être identifié.

Number of Drivers (Nombre de Pilotes) - le nombre de programmes différents requis pour utiliser les possibilités d'entrées/sorties de la carte d'acquisition.

Base Address (Adresse de Base) - comme la carte est reliée au bus interne de l'ordinateur, le système requiert une adresse unique pour que la communication puisse avoir lieu.

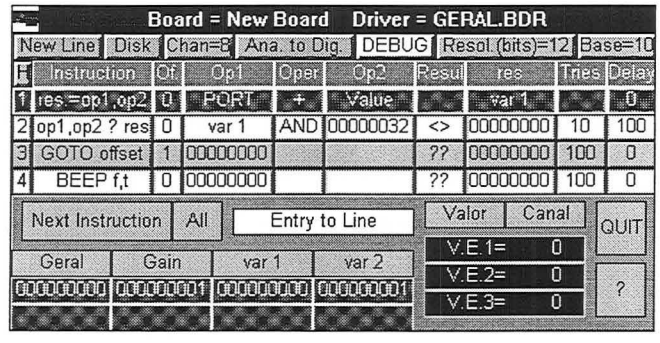
Les cartes d'acquisition peuvent présenter la combinaison de quatre sortes de conversions du signal: analogique/numérique, numérique/analogique, entrées numériques et sorties numériques. Pour chacune d'entre elles qui est présente, il faut développer un pilote d'interface. Ce pilote est en fait simplement un programme qui communique avec la carte, l'interrogeant sur les valeurs de ses entrées et envoyant des valeurs qui doivent être mises à ses sorties. Parfois il est aussi nécessaire d'initialiser la carte, donc un programme supplémentaire (pilote) doit être développé pour faire cela. Ces programmes doivent respecter les instructions du manuel des cartes qui consistent à écrire et lire des valeurs de ses registres à des adresses définies avec un décalage par rapport à l'adresse de base.

A partir de l'analyse de nombreuses cartes de quelques fabricants majeurs différents, un langage spécial a été réalisé pour que l'utilisateur puisse aisément écrire les pilotes nécessaires. Cette approche présente quatre avantages principaux:

1. ne requérir aucun logiciel supplémentaire, éviter des retards et des coûts supplémentaires pour trouver une solution quand une nouvelle carte est employée.

2. comme il s'agit d'un langage spécialisé dans ce but, il convient parfaitement à la tâche, contrairement aux différents langages d'usage général (Basic, C, etc.).
3. en utilisant une approche graphique, ce langage peut être employé dans des buts pédagogiques pour enseigner les principes d'interface d'équipement d'acquisition et de contrôle.
4. en utilisant un debogueur intégré avec exécution pas à pas, le test peut se faire en simultané avec le développement ne requérant pas le cycle type: développement - compilation - liaison au programme - test.

L'inconvénient de cette approche est que, puisque c'est un langage interprété, son temps d'exécution tend à être plus long que s'il était un programme pré-compilé. Cela peut être un problème quand les taux d'échantillonnage sont plus grands que 10 kHz (dépend aussi de l'équipement), ce qui est rarement le cas du contrôle des procédé chimiques.



Caractéristiques de chaque pilote:

Disk (Disque) - peut être sauvegardé sur le disque et lu pour que des bibliothèques de pilotes puissent être créées et distribuées.

Channels (Canaux)- nombre de canaux que le pilote contrôlera (de 0 à 32).

Description - Initialisation, Convertisseur Analogique Numérique, Convertisseur Numérique Analogique, Entrées Numériques, Sorties Numériques.

Resolution - nombre de bits des données manipulées (de 2 à 32).

Programme - lignes de code réalisant l'interface entre une fonction spéciale de la carte d'acquisition et l'objet Carte. Pour une description du langage de programmation, voir Annexe B - Langage de Programmation pour les pilotes des Cartes d'Acquisition.

L'objet Carte d'Acquisition, comme indiqué ci-dessus, peut lire et écrire les valeurs des cartes d'acquisition. Comme chaque carte peut avoir des fonctions différentes (Analogique/Numérique, Numérique/Analogique,...) et chaque fonction plusieurs canaux, il n'est pas possible de relier les objets Carte à d'autres objets de la même façon que cela a été fait avec les Blocs de Calcul (chacun avait seulement une sortie). Pour résoudre ce problème, une instance spéciale du Bloc de Calcul a été créée pour se connecter avec des objets Cartes d'Acquisition. Le bloc de calcul a été appelé 'Connexion à la Carte' et il présente trois paramètres: 'BOARD' qui indique à quel Carte il est relié; 'Driver Number' indiquant à quel pilote de la carte il est relié; 'CHANNEL' indiquant quel canal de l'objet Carte il utilise. Avec ces trois paramètres, une relation unique à un canal de la carte d'acquisition et contrôle, d'entrée ou de sortie, est complètement définie.

Finalement, une traduction des valeurs lues ou écrites d'une carte d'acquisition est habituellement nécessaire. Par exemple, dans une carte d'acquisition avec une résolution 12 bits, les valeurs de l'entrée vont de 0 à 4095 qui peuvent correspondre à un courant entre 4 et 20 milliampères. Cela à son tour peut représenter une température allant de - 10°C à 200°C. Pour faire la conversion de chaque canal, deux constantes k et m existent. Si le canal correspond à une entrée (acquisition), alors la fonction exécutée est VALEUR = k * VALEUR DE L'ENTRÉE + m. Si le canal correspond à une sortie (contrôle), alors la fonction exécutée est VALEUR DE LA SORTIE = k * VALEUR + m. Pour trouver les valeurs 'k' et les valeurs de 'm' appropriées, un menu spécial est disponible.

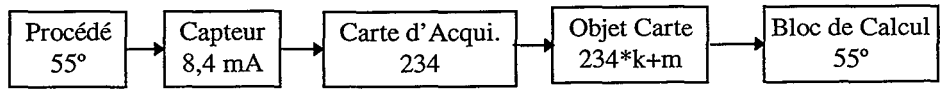


Figure 44 - Lecture d'une Variable du Procédé

Dans la Figure 44, on voit les conversions successives pour lire une valeur du procédé dans le programme ACRUN.

De même, quand une variable est délivrée à la sortie de la carte, elle doit être traduite en valeur appropriée avant de l'écrire dans la carte de contrôle. L'actionneur doit alors débiter au procédé la valeur originale du Bloc de Calcul.

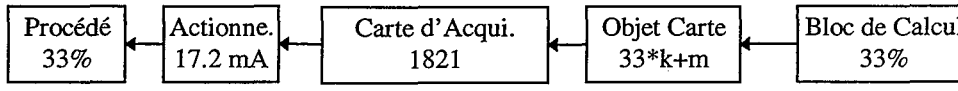


Figure 45 - Sortie d'Une Variable Vers le Procédé

Même si le fonctionnement de l'objet Carte d'Acquisition semble compliqué, cela est dû à la nature des conversions et des interfaces impliquées. Un progrès significatif été fait en simplifiant et en isolant sa complexité du reste du programme le confinant en un seul objet.

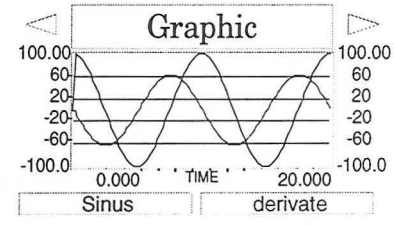
III.5/ Graphiques, Tableaux et Compteurs

Les graphiques de visualisation des signaux sont essentiels au développement et à la mise en oeuvre de la commande de procédé. Typiquement, les programmes développés par les chercheurs ont seulement des chiffres défilant sur l'écran tandis que la simulation progresse. Puis, quand la simulation est terminée, ils peuvent utiliser un second programme pour représenter ces nombres sous une forme graphique. Cela doit être considéré comme une situation normale pour ces chercheurs qui ont pour tâche essentielle de développer le logiciel de calcul qui est complètement différent du développement des interfaces d'utilisateur [HURLEY 93]. Même quelques logiciels de simulation ne produisent pas de débit visuel pendant que la simulation progresse.

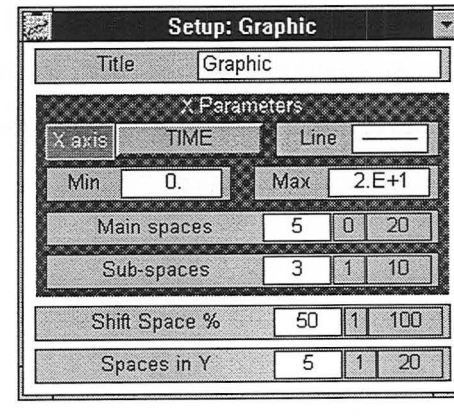
Il est courant d'avoir des simulations qui s'exécutent sur plusieurs heures, il devient alors très utile d'avoir des graphiques représentant certains signaux qui sont en cours de calcul. De cette façon, il est beaucoup plus facile de détecter toute anomalie et d'arrêter la simulation immédiatement.

Dans l'opération de contrôle, les graphiques temps réel sont une nécessité, pour un meilleur fonctionement et la surveillance; des compteurs temps réel sont aussi communément employés. Voulant prendre en compte simultanément les environnements de développement et de mise en oeuvre, ACRUN inclut des objets Graphiques, des Tableaux et des Compteurs qui peuvent être employés dans des simulations de commande et en ligne avec le procédé. De cette façon, le même graphique créé par l'ingénieur d'automatique pour l'aider dans sa recherche peut être employé par l'opérateur d'usine pour surveiller le procédé. Le travail supplémentaire exigé pour réaliser un graphique défilant avec des signaux produits en ligne est aussi un bénéfice pour le développement de la commande.

GRAPHIQUES



Les objets graphiques réalisés sont bidimensionnels avec un axe X représentant habituellement le temps et admettant n'importe quel nombre de signaux sur l'axe Y. L'axe X défile automatiquement vers la gauche ou vers la droite conformément à l'échelle définie et à la valeur du signal représenté (si c'est le temps, alors il défilera uniquement vers la gauche quand le temps actuel dépasse la limite supérieure de l'échelle de X).



La possibilité commune à tout graphique est de définir:

Title (Titre) - comment l'objet doit être identifié.

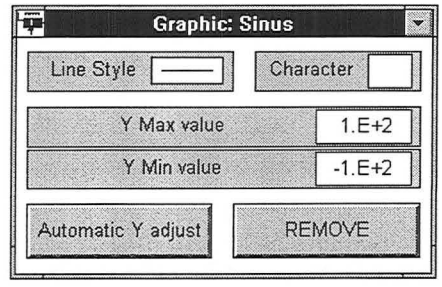
X Axis (Axe X) - le signal représenté sur l'axe X (par défaut c'est le temps).

Line (Ligne)- la ligne pour faire les divisions dans le graphique.

Min, Max - les limites supérieures et inférieures de l'axe X.

Main Spaces, Sub Spaces (Espaces Principaux, Sous Espaces) - divisions dans l'axe X.

Shift Space % (Espace de Défilement %) - le nombre, exprimé en pourcentage, indiquant la quantité de défilement qui doit être exécutée.
Spaces in Y (Espaces en Y)- divisions dans l'axe Y.



Pour chaque signal représenté sur l'échelle Y, les paramètres suivants peuvent être définis:
Line Style (Style de Ligne) - pour représenter le signal sur les graphiques joignant des points consécutifs.
Character (Caractère)- pour être dessiné à chaque point de l'échantillonnage.
Y Max Value (Valeur Y min, Valeur Y max) - les limites supérieures et inférieures de l'échelle Y pour ce signal.

Un ajustement automatique de l'échelle Y est possible pour établir les limites minimum et maximum de l'échelle Y du signal (quand des données sont déjà disponibles). Il y a aussi la possibilité d'uniformiser l'échelle Y pour tous les signaux représentés en Y ou de maximiser chaque échelle Y individuellement.

Avec la souris, on peut faire le zoom d'une certaine portion des signaux représentés sur le graphique. Ce zoom peut être effectué sur l'axe X et sur l'axe Y. Cette manipulation visuelle des signaux sur les graphiques peut être très utile au développement de la commande parce qu'il s'agit d'une façon aisée d'accéder à l'information du type "Au temps = t, quelles étaient les valeurs de y1, y2 et y4?".

TABLEAUX

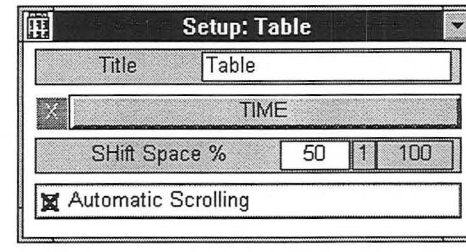
L'objet Tableau simule le défilement des données à l'écran en visualisant dans chaque ligne la valeur des signaux correspondants à un moment d'échantillonnage. Quand il n'y a plus d'espace, le tableau défile vers le haut, aussi bien en faisant de la simulation de contrôle qu'en traitant des données en ligne avec le procédé.

Table	Sinus	derivate
0.200		0.000
0.400	96.858	-11.766
0.600	92.978	-19.403
0.800	87.631	-26.735
1.000	80.902	-33.645
1.200	72.897	-40.024
1.400	63.742	-45.772
1.600	53.583	-50.799
1.800	42.578	-55.024
2.000	30.902	-58.381
2.200	18.738	-60.818
2.400	6.279	-62.295
2.600	-6.279	-62.791
2.800	-18.738	-62.295
3.000	-30.902	-60.818
3.200	-42.578	-58.381
3.400	-53.583	-55.024
3.600	-63.742	-50.799

L'avantage que ces objets amènent est que leurs contenus peuvent défiler vers le haut et vers le bas au gré de l'utilisateur et n'importe laquelle des valeurs représentée dans le tableau peut être changée.

En montant ou en descendant dans le Tableau l'utilisateur peut sélectionner un certain temps d'échantillon associé à la simulation. En allant vers la gauche ou vers la droite dans le Tableau, il peut sélectionner une colonne correspondant à un certain signal (des données venues d'un autre objet, par exemple, d'un Bloc de Calcul) et enfin il peut éditer cette valeur.

La possibilité commune à tous les objets Tableau est de définir:



Title (titre) - comment l'objet doit être identifié.

X - le signal définissant la signification de chaque ligne dans le Tableau (le temps par défaut).

Shift Space % (Espace de Défilement %) - le nombre, exprimé en pourcentage, indiquant la quantité de défilement à exécuter quand cela est nécessaire.

Automatic Scrolling (Défilement Automatique) - valeur booléenne indiquant si le défilement automatique est permis ou s'il utilise seulement les ordres manuels de l'utilisateur pour exécuter les défilements.

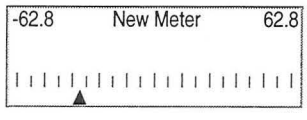
Les Tableaux, comme les Graphiques, peuvent seulement représenter les données qui sont disponibles dans les autres objets parce qu'ils n'ont pas d'espace pour emmagasiner les données. Quand un Tableau (ou un Graphique) semble inachevé (certains points manquent), habituellement c'est parce qu'il n'y a pas assez de données dans les objets représentés (dans les Blocs de Calcul, cela veut dire que la taille du tampon de données utilisé doit être augmentée).

COMPTEURS

Importants dans l'opération de contrôle et surveillance, les Compteurs sont une partie intégrée du plan synoptique des procédés. Trois sortes de compteurs sont réalisées dans la même classe d'objet Compteur: le Compteur à barres, le Compteur analogique et le Compteur numérique. Ces compteurs peuvent lire la valeur d'un objet, comme le Bloc de Calcul, et le représenter. Dès que la sortie de cet objet change, le Compteur est automatiquement mis à jour pour tenir compte du changement.

Le Compteur à Barres - possède une barre qui croît ou rétrécit conformément à la valeur de l'entrée du Compteur. L'orientation de la Barre peut aller vers le haut, vers le bas, vers la gauche ou vers la droite. Les valeurs minimum et maximum de la barre peuvent être définies ainsi que la couleur et la texture de cette barre et de son arrière-plan.

Le Compteur Analogique - Consiste en une échelle avec les valeurs minimum et maximum indiquées; un curseur se déplace verticalement ou horizontalement indiquant la valeur du moment. Il donne plus d'information que le compteur à barres à cause des subdivisions de son échelle, mais a moins d'impact visuel.



Compteur Numérique - montre la valeur de l'entrée. La police et la taille de caractères dans laquelle il est écrit

-63.742

peuvent être choisies ainsi que le nombre de décimales utilisées.

Tous les Compteurs ont un nom qui les identifie parmi les autres objets. Beaucoup d'autres types de compteurs peuvent être aisément développés. Dans les opérations de procédés, il y a beaucoup de demandes pour des synoptiques animés qui puissent simuler ce qui se passe dans l'usine.

III.6/ Fichiers

Si la simulation peut produire de grandes quantités de données, la mise en oeuvre de systèmes avec surveillance du procédé en génère encore plus. Il est courant d'avoir, par exemple, six variables d'un même procédé qui sont échantillonnées deux fois par minute et pour lesquelles l'historique sur une période de deux mois doit être gardé. Plus d'un million de valeurs réelles devrait alors être sauvegardées, habituellement sur le disque, pour des raisons techniques et de sécurité. On a vu que les Blocs de Calcul ont des tampons où sont gardées les anciennes données, mais ces tampons n'ont pas vocation d'emmagasinage à long terme, parce qu'ils consomment beaucoup de mémoire vivante (RAM), qui est assez limitée, tout en pénalisant la performance du logiciel. De plus, s'il y a une panne de l'ordinateur ou une anomalie du logiciel, toutes les données seront alors perdues. Une sauvegarde des données sur un support permanent est impérative.

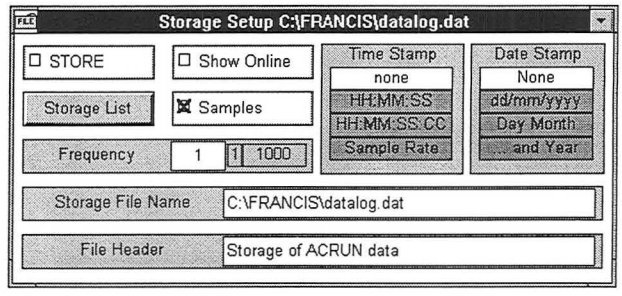
Mêmes dans de courtes simulations, il peut être utile de garder certaines données calculées dans un fichier à employer ultérieurement, pour fournir ces données à un autre programme (programmes statistiques et graphiques, feuilles de calcul) ou pour les transférer vers un autre ordinateur.

L'objet Fichier a été développé pour sauvegarder des données dans un fichier sur le disque. Il lit les données des autres objets et les écrit sur le disque. Les données de chaque échantillon sont écrites dans une nouvelle ligne sur le fichier en disque et, le

plus souvent possible, elles sont effectivement transférées vers le disque (l'antémémoire du disque est vidée) pour éviter leur perte.

La possibilité commune à tous les objets Fichiers est de définir:

Storage File Name (Nom du Fichier de Sauvegarde) - comment identifier l'objet. Le nom est exactement le même nom que celui du fichier sur le disque avec information du lecteur et du répertoire.



File Header (En-tête de Fichier) - la chaîne de caractères à écrire en début de fichier.

Store (Garder) - permet ou pas l'écriture des données sur le disque.

Show On-Line (Montre En-Ligne) - pour montrer dans une fenêtre ce qui est en train de s'écrire dans le fichier.

Samples (Echantillons) - écrire ou pas le numéro de l'échantillon au début de chaque nouvelle ligne du fichier.

Time Stamp (Affichage du Temps) - choisit le format du temps à écrire dans chaque nouvelle ligne du fichier.

Date Stamp (Affichage de la Date) - choisit le format de la date à écrire dans chaque nouvelle ligne du fichier.

Frequency (Fréquence) - à quel intervalles les données doivent être écrites dans le fichier en relation avec la période d'échantillonnage. Si cette valeur est '1' alors elles seront toujours écrites, si c'est '2' elles ne seront écrites qu'une fois sur deux, et ainsi de suite.

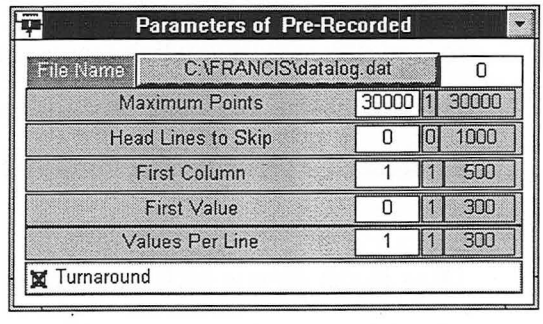
Storage List (Liste de Sauvegarde) - la liste d'objets dont les valeurs doivent être écrites dans le fichier sur le disque. L'ordre dans lequel les objets apparaissent sur cette liste est celui de la colonne correspondante de données dans le fichier.

Si la période d'échantillonnage d'un procédé est, par exemple, de 10 secondes, il n'est peut-être pas nécessaire de garder les valeurs pour tous les échantillons. Si le

paramètre Fréquence a la valeur '6', alors les données seront écrites sur le disque seulement à chaque minute économisant ainsi de l'espace disque.

Quand une simulation ou une acquisition commence, le fichier peut déjà exister sur le disque. Dans ce cas, l'utilisateur est interrogé sur le choix, soit de l'effacer, soit de lui adjoindre les nouvelles données à la fin. Le contenu du fichier peut être visualisé et édité en utilisant les processeurs de texte (il ne contient que du texte). Le programme Notepad de Windows peut être exécuté automatiquement, à partir du menu local de l'objet, pour éditer le fichier.

De même qu'il est utile de sauvegarder des données dans un fichier, il est souvent nécessaire de lire aussi les données d'un fichier (ces données peuvent répresenter un signal). Cela est réalisé en ACRUN en utilisant un objet Bloc de Calcul nommé 'Pre-Recorded' (préenregistré). Les paramètres de ce Bloc sont au nombre de sept et ils suffisent pour répondre aux formats de fichier généralement utilisés pour conserver des données.



Ces paramètres sont:

File Name (Nom de Fichier) - nom du fichier sur le disque où les données sont sauvegardées. Normalement, le nom vient d'un objet du type Fichier.

Maximum Points (Maximum de Points)- limite le nombre de données requises.

Head Lines to Skip (Lignes d'En-tête à Sauter) - ignore un nombre de lignes du début du fichier.

First Column (Première Colonne) - position du premier caractère à compter du début de la ligne où commencer à lire.

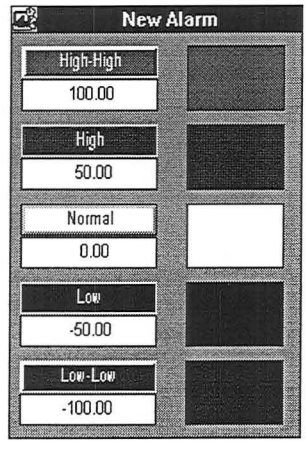
First Value (Premier Valeur) - définit la valeur initiale à lire.

Values per Line (Valeurs par Ligne) - nombre de valeurs à lire dans chaque ligne.

Turn Around (Tourner Autour) - valeur booléenne indiquant si, quand on atteint la fin du fichier, il devrait continuer à lire à nouveau à partir du début. Ceci est utile quand les données emmagasinées constituent un signal périodique.

Avec l'objet Fichier et le Bloc de Calcul 'Pre-Recorded', ACRUN réalise toutes les possibilités de sauvegarder et de récupérer des données d'un fichier sur le disque.

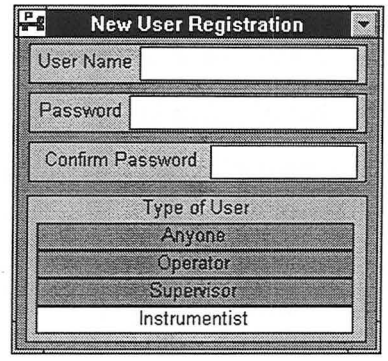
III.7/ Les Alarmes



Importantes dans le contrôle et la surveillance de la mise en oeuvre en usine, les alarmes peuvent être décisives pour la bonne performance du logiciel de commande et, à la limite, en dictant la qualité et la raison d'être du programme pour une mise en oeuvre spécifique. Les alarmes vont de pair avec la sécurité en fournissant un environnement aussi sûr qu'optimal. Elles visent non seulement à empêcher que les accidents se produisent mais elles sont aussi en partie responsables de la qualité de production du procédé contrôlé. Un système d'alarme puissant et flexible peut être un complément majeur de la stratégie de commande appliquée au procédé facilitant le travail de l'operateur et du superviseur.

Dans le cas où une certaine anomalie ou accident affecte le procédé, les alarmes et la sécurité implantées peuvent aider à la reconstitution des événements qui ont conduit à cette situation. A la limite, les responsabilités peuvent être attribuées aux gens qui étaient supposés en être en charge et qui ont réagi lentement, ou pas du tout, à certains événements d'alarme qui se sont déclenchés.

Un composant important de la sécurité est le Contrôle de l'accès des utilisateurs. Dans ACRUN, ce contrôle est réalisé par un Nom d'identification, par un mot-de-passe d'entrée et par des Privilèges d'Utilisateur. Les deux premiers permettent à l'utilisateur d'accèder à ACRUN, ce qui veut dire, que le programme est informé à partir d'un certain instant de l'utilisateur responsable des actions. Les Privilèges d'Utilisateur peuvent être PERSONNE, OPERATEUR, SUPERVISEUR et INSTRUMENTISTE dans l'ordre croissant d'actions qu'il est permis à l'utilisateur d'exécuter.



L'objet Alarme a été réalisé pour répondre aux objectifs suivants:

- avertir l'utilisateur quand certaines situations surviennent.
- commencer une procédure quand certaines situations surviennent.
- enregistrer la chronologie des occurrences de certaines situations.
- enregistrer la chronologies des actions des utilisateurs en réponse à l'ocurrence de certaines situations.

'Certaines situations' que les alarmes détectent provoquent une action qui sera référée comme déclenchement de l'alarme. Cela résulte normalement de la violation par une variable de quelques conditions normales d'opération. Le déclenchement d'une alarme ne signifie pas toujours qu'une condition dangereuse ou anomale survienne. Il signifie toujours qu'une situation spéciale a eu lieu et qu'elle doit être enregistrée et/ou prise en compte par l'utilisateur. L'action de la part de l'utilisateur pour informer le programme qu'il est au courant du déclenchement d'un alarme est appelée 'constatation'.

Il y a quatre niveaux standard d'alarmes dans l'industrie: *Low-Low* (Bas-Bas), *Low* (Bas), *High* (Haut) et *High-High* (Haut-Haut). Dans ACRUN, ces quatre niveaux sont présents et des procédures spéciales peuvent être associées à chacun d'entre eux. Chaque objet Alarme présente les paramètres suivants:

Name (nom) - comment l'objet doit être identifié.

Operation (Opération) - définit quatre situations quand il doit être opérationnel: 1) jamais opérationnele; 2) quand la simulation survient; 3) quand l'acquisition en ligne survient; 4) toujours opérationnel.

Low-Low, Low, High and High-High (Bas-Bas, Bas, Haut et Haut-Haut): niveaux d'alarmes avec chacun des paramètres associés.

Input (Entrée) - l'objet qui fournit la valeur de sortie qui doit être contrôlée et comparée avec les quatre valeurs (niveaux) de l'alarme.

Chaque niveau d'alarme a les paramètres suivants:

Value (Valeur) - c'est la valeur à laquelle la sortie de l'objet d'entrée est comparée. Pour les cas des niveaux Bas-Bas et Bas, quand la contribution est inférieure ou égale à cette valeur, alors l'alarme est déclenchée. Pour les cas des niveaux Haut et Haut-Haut, l'alarme est déclenchée quand la contribution est supérieure ou égale à cette valeur.

Priority (Priorité) - numéro de 1 à 100 exprimant l'importance croissante de répondre à l'Alarme. Si une Alarme déclenchée a la priorité 50 et qu'aussitôt une autre a la priorité 90 est également déclenchée, alors cette dernière demande en priorité l'intervention de l'utilisateur (opérateur).

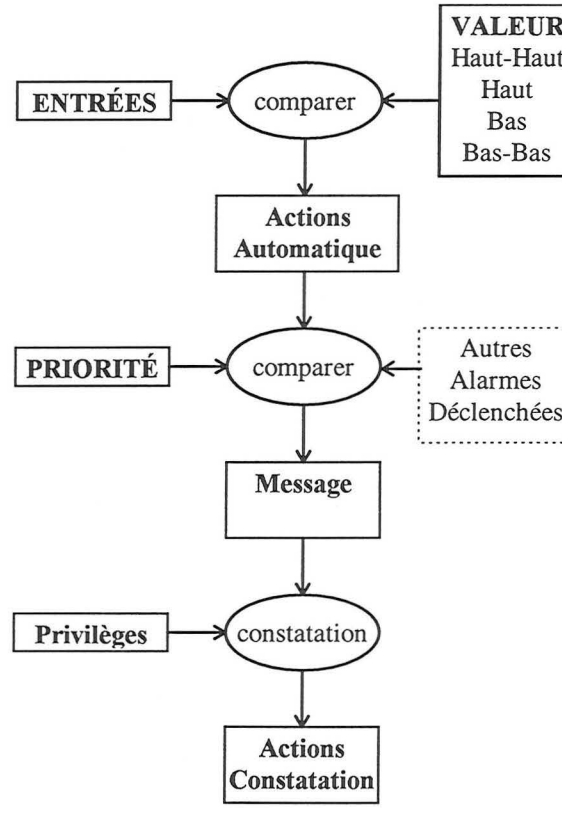


Figure 46 - Structure des Procédures d'Alarme

Message - message de notification à exposer quand l'alarme est déclenchée.

Privileges (Privilèges) - les privilèges que l'utilisateur doit avoir pour faire la constatation d'alarme. Faire la constatation d'une alarme correspond à cliquer sur le bouton 'Acknowledge' (constatation) qui paraît avec le Message.

Register (Registre) - enregistrer ou non l'alarme dans l'historique des alarmes. S'il est opérationnel, le moment où l'alarme a été déclenchée et le retard que l'utilisateur a pris pour le constater sont enregistrés.

Automatic Action (Action Automatique) - liste d'objets à exécuter aussitôt que l'alarme se déclenche. Cela peut être l'ouverture d'une soupape de sécurité ou la mise en marche d'une sirène d'alarme.

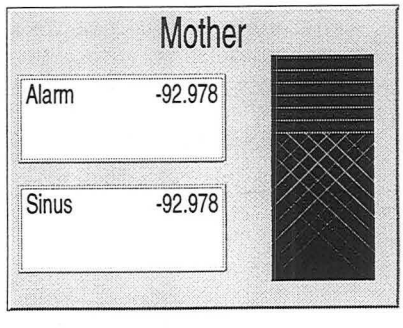
Acknowledge Action (Action de Constatation) - liste d'objets à exécuter après la constatation par l'utilisateur. Cette action peut avoir pour conséquence l'affichage des instructions pas à pas afin de traiter la situation.

Une couleur est associée à chaque niveau d'alarme et un menu d'alarmes global est disponible pour vérifier rapidement les états de toutes les alarmes. La visualisation de l'histoire des alarmes et l'effacement de ses lignes peuvent être d'accès limité (par exemple aux superviseurs et aux instrumentistes). La chronologie des alarmes est aussi sauvegardée avec les fichiers de configuration d'ACRUN pour que l'histoire entière du procédé puisse être analysée ultérieurement. La possibilité d'activer ou pas toutes les alarmes simultanément est aussi disponible et devrait être employée avec soin.

Finalement, la couleur des objets Alarme est celle correspondant à l'état d'alarme. Si la valeur d'entrée est, par exemple, entre les niveau Bas-Bas et Bas, alors l'Alarme aura la couleur correspondante au niveau Bas. La couleur des Alarmes peut être employée comme la couleur du tracé dans un graphique, par exemple, ou dans la couleur d'un Compteur à Barres pour informer de l'état d'alarme de sa valeur d'entrée.

III.8/ LA MÈRE

Cet objet est employé pour grouper d'autres objets. Il suffit de les placer sur l'écran sous les autres objets et d'exécuter la commande 'Capture' de son menu local. Dès lors, les objets capturés ne



peuvent plus être déplacés individuellement parce qu'ils sont rattachés à l'objet Mère.

L'utilité de combiner des objets ensemble s'applique à la structuration organisationnelle et fonctionnelle groupant dans un objet Mère, par exemple, un bloc de Calcul et l'Alarme qui le surveille, ou groupant un Graphique et un Compteur à barres qui ont le même Bloc de Calcul comme entrée.

Les objets Mère peuvent aussi être groupés comme tout autre objet; donc une structure hiérarchique peut être créée. Dans le développement de la commande, il est commun d'avoir des objets Mère groupant ensemble d'autres objets Mère. Cela peut être important pour expliquer visuellement le procédé et les algorithmes réalisés. L'objet Mère est considéré comme un objet essentiel pour la commande de procédé en utilisant ACRUN.

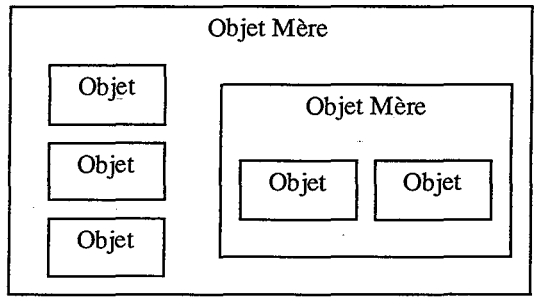


Figure 47 - Hiérarchie d'Objets Mère

Finalement, les Blocs de Calcul qui sont reliés au même objet Mère peuvent avoir leurs valeurs par défaut rédigées dans un même menu, allégeant la tâche du développeur d'établir et de vérifier les données initiales d'une simulation (voir le chapitre VI - Dans Le Laboratoire).

III.9/ Les Autres Objets

Les objets décrits antérieurement sont directement liés au développement de la commande et/ou à la mise en oeuvre. S'il en manque quelques-uns parmi eux, ce projet sera incomplet même si les techniques orientées objet employées avaient donné aux objets restants, une grande puissance d'opération, de flexibilité, et de bonnes aptitudes d'intégration.

Comme il s'agit d'un projet de recherche, il était important de faire appel aux avantages offerts par la programmation orientée objet, en intégrant d'autres domaines. Cela donne lieu à de toutes nouvelles fonctionnalités, autres que celles que l'on attend d'un logiciel de commande et d'acquisition. Ces objets supplémentaires étaient développés sans accroître la complexité du projet et sans même exiger le moidre changement au code déjà développé. La fonctionnalité croissante (objets), sans la complexité croissante de code, est un des attributs principaux de la programmation orientée objet dû à l'encapsulation et au polymorphisme.

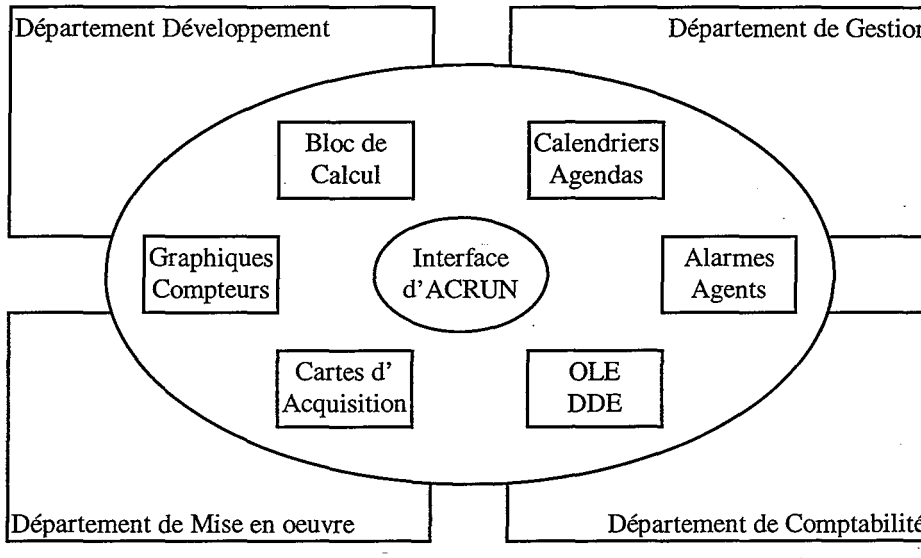
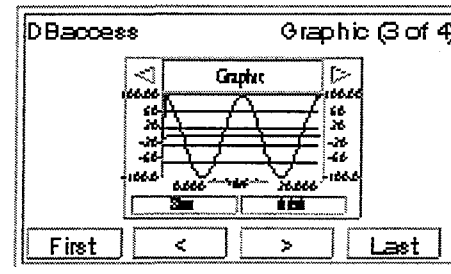


Figure 48 - L'Intégration d'Objets Multi-Fonctionnels

En ayant non seulement intégré le développement de la commande avec la mise en oeuvre, mais aussi en intégrant à ceux-ci d'autres outils d'utilisation générale, l'environnement résultant devient plus productif et complet.

La BASE D'OBJET

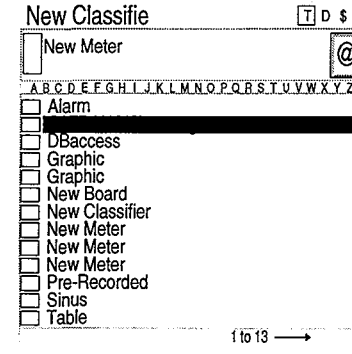


Cet objet a la vocation d'accéder à des objets différents dans une sorte de base de données. Il peut contenir une liste d'objets qui sont visualisés en utilisant des contrôles comme dans les vidéos. L'objet actuellement sélectionné peut être activé et changer tout comme si l'objet réel était présent et tous les changements faits sont réfléchis conformément dans tous les autres objets.

Il peut être utile, par exemple, d'avoir une Base d'Objet avec tous les objets graphiques employés dans le développement de la commande. Alors n'importe quel graphique peut être aisément sélectionné d'une liste ou visionné l'un après l'autre. Au cas où beaucoup de graphiques sont employés, cela peut se révéler un outil intéressant.

Disposer sur l'écran de deux objets Base d'Objets, chacun d'entre eux ayant dans ses listes tous les objets disponibles, fait que comparer deux d'entre eux devient immédiat. Cela aide à résoudre le problème d'avoir trop de données à notre disposition en rendant plus facile le filtrage de l'information qui est pertinente à un certain moment.

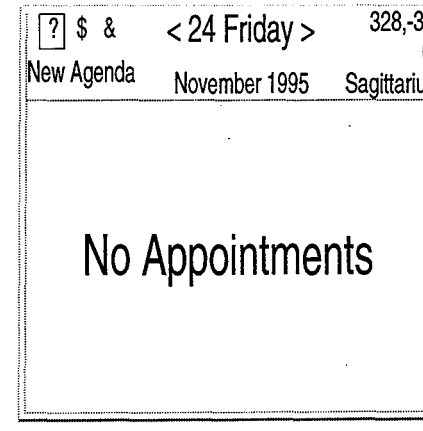
Le CLASSEUR



Basé sur l'analogie classique du répertoire téléphonique, l'aspect visuel de cet objet est la répartition en pages avec des lettres de A à Z. Au lieu d'emmagasiner des noms, des adresses et des numéros de téléphone, il emmagasine des objets (qui, eux, peuvent avoir tout cela).

Comme on l'a vu, chaque objet a un nom qui l'identifie. Les objets réalisant la commande du procédé A peuvent être reliés à l'objet Classeur A . Des recherches peuvent alors être faites aussi aisément qu'en cherchant des noms sur un carnet de téléphone. Un autre Classeur peut être créé, on va le nommer B, et tous les objets qui traitent avec le procédé B se relient à lui. Enfin, un Classeur C est créé et le Classeur A et le Classeur B sont reliés à lui. Maintenant, des recherches dans le Classeur C peuvent être faites simultanément pour les objets liés aux procédés A et B. Une fois encore, cela peut être très utile pour traiter la surabondance d'informations mentionnées plus haut.

L'AGENDA



L'objet Agenda sert de journal où des rendez-vous quotidiens et des listes de choses à faire peuvent être conservés et rappelés. Etant utile à tout le monde, il est aussi utile à un développeur de commande ou à un ouvrier d'usine. Sans l'approche objet, ce serait probablement une mauvaise pratique d'ajouter l'Agenda parce que, même s'il est utile, la fonctionnalité supplémentaire pourrait ne pas compenser la complexité due au développement entier du logiciel. Ajouter un Agenda pourrait provoquer des erreurs à paraître dans le noyau du logiciel ou sur la fonctionnalité essentielle du programme: le système de développement et la mise en oeuvre du contrôle.

Comme ce n'est pas le cas avec le développement orienté objet, aucune hésitation n'a été ressentie en ajoutant cet objet. Si l'utilisateur ne le demande pas, l'objet Agenda n'entraînera aucune pénalité au niveau de la performance quand, par exemple, l'écran est redessiné.

En plus, il y a une fonctionnalité supplémentaire dans l'objet Agenda très utile pour la supervision de procédé. Il y a un Bloc de Calcul appelé 'Daily Values' (Valeurs Journalières) qui a la particularité d'emmagasiner une seule valeur chaque jour. Par exemple, il peut totaliser le carburant dépensé dans une usine ou toute autre valeur qui doit être enregistrée sur une base quotidienne. Il peut alors être représenté dans un graphique ou dans un tableau; mais ce ne sont pas des façons naturelles aux hommes de consulter les dates. Si ces Blocs de Calcul 'Daily Values' sont reliés à un objet Agenda, alors, connaître la quantité de carburant consommée un certain jour devient facilement vérifiable.

En effet, tout objet qui emploie des dates avec référence à une information peut être visualisé dans l'Agenda. L'ouverture de l'approche d'objet donne la possibilité à l'utilisateur d'interconnecter les objets existants d'une manière jamais imaginée par ses développeurs. Pour réaliser cela, les objets doivent être intuitifs à employer et à interconnecter entre eux en utilisant des protocoles standard (réalisant l'intégration de présentation).

L'AGENT

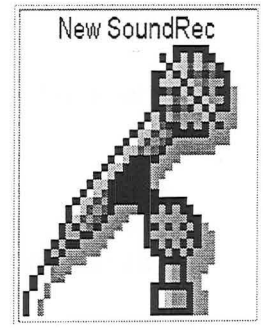
Cet objet est employé pour exécuter des actions glisser - lâcher. Ces actions s'effectuent en lâchant des objets dans des fenêtres ou en lâchant des fenêtres dans des objets. Si, par exemple, un objet Graphique est lâché dans la fenêtre De zoom, alors le Graphique exposé va prendre la taille du plein écran. Beaucoup d'actions différentes peuvent être définies de cette façon.

Exécuter l'objet Agent, c'est exécuter son action. Si un objet Agent est relié aux actions automatiques ou aux actions après constatation de l'objet Alarme, alors des choses intéressantes peuvent être définies. Par exemple, quand la valeur du Bloc de calcul liée à l'alarme dépasse le niveau Haut-Haut, un Zoom peut être fait à l'écran du synoptique qui représente la partie du procédé concerné. Avant que l'action soit

exécutée, sa description peut être montrée à l'utilisateur et la confirmation ou l'annulation de son exécution requise.

Les objets Agent peuvent être reliés entre eux, pour que, quand un objet Agent est exécuté, un certain nombre de points soient envoyés aux autres agents auquel il est relié. Ceux-ci ajoutent les points reçus à ceux qu'ils ont déjà et si le total équivaut ou dépasse un certain nombre de points, ils seront alors exécutés. De cette façon, les objets Agent peuvent être employés comme les blocs fondamentaux pour réaliser des réseaux de Petri, ce qui peut être utile, par exemple, dans l'automatisation de lignes de production (voir Annexe E - Réseaux de Petri Utilisant des Objets Agent).

Les objets OLE



En basant le développement entier sur des objets, les standards émergeant de l'industrie logiciel pour l'intégration des objets ne peuvent pas être ignorés. L'objet OLE (Object Linking and Embedding) a été réalisé sur la base des normes Microsoft, en ayant déjà une large acceptation. Il consiste dans l'incrustation ou la liaison de données dans ACRUN, en gardant l'information du type d'application qui peut manipuler ces données (après tout, c'est essentiellement la philosophie du modèle objet).

Quand la liaison se fait avec un objet, seulement l'indication de l'endroit où se trouvent les données et le programme (qui doit être un serveur d'OLE) qui peut les éditer est gardée. Lors de l'incrustration d'un objet, les données entières de celui-ci sont gardées dans ACRUN ainsi que l'indication du programme qui peut les éditer.

Les avantages de la liaison font que le même objet peut être employé par beaucoup de programmes sans redondance de données. Tous les changements à l'objet sont répercutés dans tous les programmes qui l'emploient, garantissant ainsi la synchronisation des données. Les avantages des données incrustées sont que même

en opérant sur un ordinateur différent, les données sont encore disponibles (tandis que les liaisons peuvent être perdues). Si le même programme serveur OLE est aussi disponible dans cet ordinateur, alors les

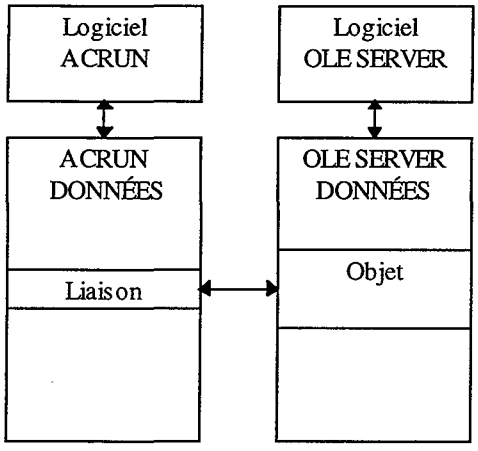


Figure 49 - Liaison OLE

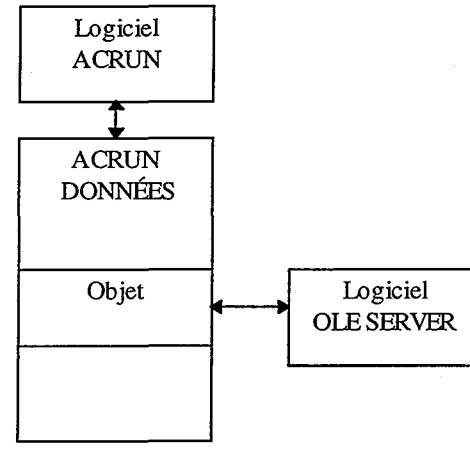


Figure 50 - Incrustation OLE

données peuvent toujours être éditées.

Deux actions sont habituellement réalisées dans les objets OLE : 'exécuter' et 'éditer' l'objet.

Voyons certains types d'objets OLE et ses programmes serveurs déjà disponibles:

Traitement de Texte - les données de l'objet sont le document, et 'éditer' l'objet, c'est travailler dans un environnement typique de traitement de texte. 'Exécuter' l'objet n'a aucune signification concrète.

Programme de Dessin - les données de l'objet sont l'image et 'éditer', c'est changer le dessin ou le tableau. 'Exécuter' l'objet n'a aucune signification concrète.

Programme de Son - les données de l'objet représentent un son (musique, phrases dictées, etc.). 'Editer' l'objet peut vouloir dire qu'on va l'enregistrer avec un microphone. 'Exécuter' l'objet, c'est diffuser le son par les haut-parleurs reliés à l'ordinateur.

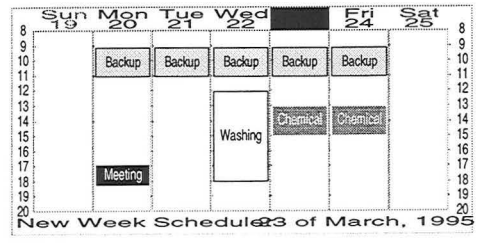
Films Vidéo - les données de l'objet représentent un film. 'Editer' l'objet, c'est enregistrer, comprimer ou sélectionner une séquence du film disponible. 'Exécuter' l'objet, c'est voir ce film à l'écran (il peut aussi avoir une bande sonore jointe).

Des exemples d'intégration des objets utilisant OLE deviennent communs comme, par exemple, en CAPE (Computer Aided Process Engineering) pour intégrer les données des modèles [PREECE 94].

Tandis que le multimédia traite de problèmes difficiles de synchronisation [HORN 93] dûs aux différences dans les formats de fichiers et d'équipement, utiliser des objets OLE pour le multimédia économise beaucoup de temps de développement. Il n'y a même pas le problème d'acheter les programmes qui agissent comme des serveurs d'objets multimédia parce qu'ils sont fournis déjà avec le système d'exploitation ou avec l'équipement multimédia qui est installé dans l'ordinateur.

Comme c'est le cas de l'objet Agent, les objets OLE peuvent être très utiles dans le raccordement aux objets Alarmes. Il est facile d'imaginer les avantages quand l'opérateur fait la constatation d'une alarme, et qu'il écoute le message 'Fermer la Soupape N° 3 et appeler le Superviseur' ou même qu'il regarde un film qui montre comment la soupape N°3 se ferme!

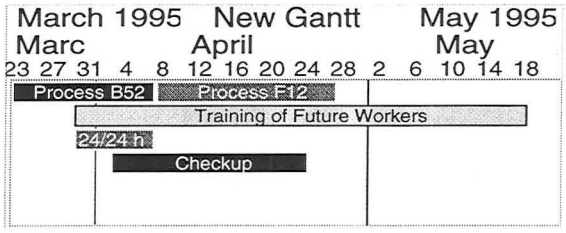
LE PLAN HEBDOMADAIRE, LE CALENDRIER, LE GRAPHIQUE DE GANTT



Enfin, il y a d'autres objets qui, comme cela a été dit précédemment, peuvent être utiles à n'importe quel utilisateur d'ordinateur. Ils sont employés pour gérer des rendez-vous et des tâches. Davantage de développement de ces objets peut néanmoins amener à

d'intéressantes caractéristiques nouvelles pour le développement de la commande et pour la mise en oeuvre, comme la gestion des groupes et des projets.

L'objet Gantt peut être développé davantage pour être employé dans l'analyse et la schématisation de l'opération des procédés. Dans une usine, des applications utilisant des méthodes comme celles-là sont décrites dans [GRAELLS 94] et [JAE 94]. Elles peuvent aussi être employées pour intégrer la planification du cycle développement/mise en oeuvre [NUMAO 95].



Il est possible dans l'avenir que des théories et de la recherche en commande de procédés puissent être appliquées pour contrôler des organisations [DRENICK 90]; donc l'intégration de ces objets, dans cette perspective, peut devenir utile.

Il a aussi été dit, mais cela mérite d'être souligné, que les développements de ces objets ne compromettent ni les performances, ni la fonctionnalité des objets de contrôle, essentielles dans ce travail.

CHAPITRE IV - Outils Avancés de Développement

IV.0/ Introduction

Dans le chapitre III, les caractéristiques fondamentales de ACRUN pour le développement et la mise en oeuvre de la commande de procédé sont décrites. La fonctionnalité avancée visant le développement de la commande de procédé sera maintenant introduite.

Qu'arrive-t-il lorsque l'utilisateur veut réaliser une fonction de commande ou un algorithme de simulation de procédé qui ne peut pas être construit avec les Blocs de Calcul disponibles? Quand c'est le cas, il doit y avoir une façon pour l'utilisateur de développer ses propres algorithmes et de les intégrer d'une certaine manière aux autres déjà existants en ACRUN. Le premier sous-chapitre décrit comment cela peut être fait.

Une tâche importante du développeur simulant des procédés ou essayant des stratégies de commande est de choisir certains des paramètres utilisés. Parfois, ils peuvent être trouvés mathématiquement et êtres interdépendants par une sorte de fonction connue. Bien souvent, il s'agit d'une méthode essai-erreur ou hypothèse-essai permettant d'acquérir ainsi la sensibilité du rôle de chaque paramètre dans le système étudié ou développé. Quand chaque simulation du procédé prend beaucoup de temps, cette tactique de 'essai-erreur' peut être grande consommatrice de temps, enclinte aux erreurs et frustrante. Ce chapitre traite aussi de la manière dont cette tâche peut être automatisée pour que l'ordinateur puisse travailler pour nous en notre absence. Des graphiques spéciaux à deux et trois dimension peuvent alors être employés pour analyser la signification et les interactions entre les paramètres impliqués et les résultats finaux des simulations.

Finalement, la fin de ce chapitre décrit la visualisation des flux de signaux disponibles pour les développeurs de contrôle et sa relation avec les objets présents dans le programme.

IV.1/ Liaison avec des DLLs - Fortran, C, Pascal

ACRUN emploie un Bloc de Calcul appelé 'External DLL' pour intégrer des fonctions développées par l'utilisateur. Ceci est fait en liant dynamiquement pendant l'exécution d'ACRUN du code externe qui est réalisé comme une DLL (Dynamic Link Library) de Windows. Tout compilateur de logiciel qui est capable de produire des exécutables (.EXE) Windows peut habituellement générer aussi des DLLs Windows. Ainsi, l'utilisateur peut choisir une large variété d'outils et de langages.

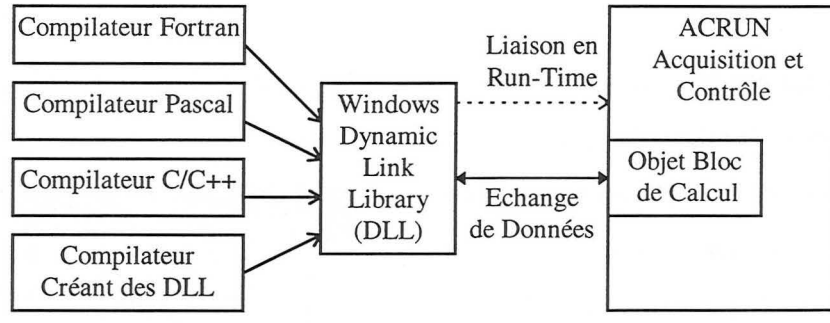
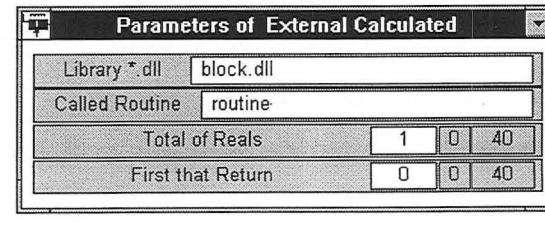


Figure 51 - Fonctions Externes dans les DLL (Dynamic Link Libraries)

Le Bloc de Calcul 'EXTERNAL DLL' a quatre paramètres:



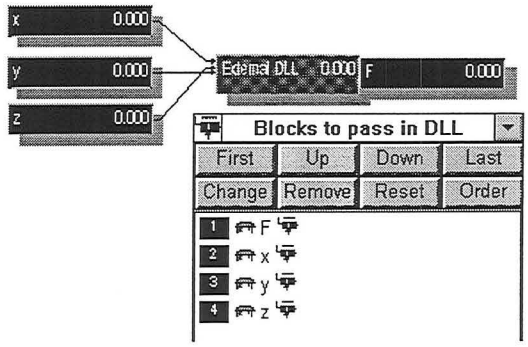
LIBRARY *.DLL (Bibliothèque *.DLL)- nom du fichier avec la bibliothèque.

Called Routine (Sous-Programme Appelé)- nom de la fonction ou procédure appartenant à la bibliothèque et qui réalise l'algorithme.

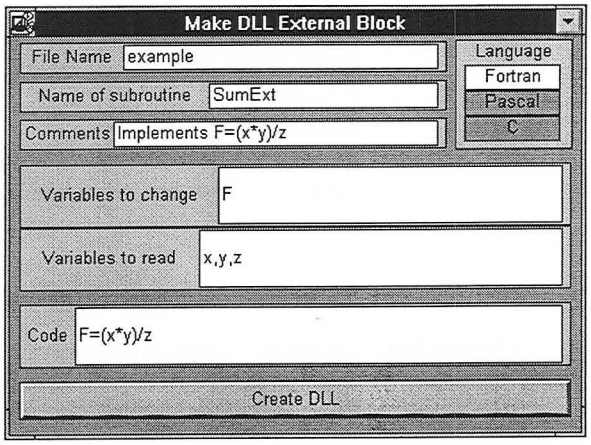
Total of Reals (Total de Réels) - nombre de valeurs réelles qui sont passées du Bloc de Calcul vers le sous-programme de la bibliothèque.

First that return (Les Premiers qui Reviennent) - nombre de valeurs réelles qui sont modifiées par le sous-programme de la bibliothèque et qui doivent être lues par le Bloc de Calcul.

On va voir un exemple pratique réalisant la fonction externe suivante F(x, y, z) = (x * y) / z. Le nombre Total de Réels serait de quatre - trois entrées x, y, z et une sortie F - et le nombre de valeurs réelles qui doivent revenir serait de un - la sortie F. Une liste est créée consistant en quatre Blocs de Calcul qui sont reliés au Bloc de Calcul Externe. Dans cette liste les 'Premiers qui Retournent' sont les Blocs de Calcul employés pour garder les sorties de la 'EXTERNAL DLL'. Le reste des Blocs de Calcul sont ceux qui servent comme entrées vers la 'External DLL'.



Cela peut sembler compliqué mais comme l'expérience l'a prouvé, c'est assez simple à employer. La tâche la plus exigeante et technique consiste à écrire l'en-tête du fichier du code source qui va créer le sous-programme DLL externe. Celui-ci constitue l'interface avec le Bloc de Calcul 'External DLL'. Pour assister l'utilisateur dans cette tâche, un menu spécial appelé 'Make DLL External Block' (Faire Bloc DLL Externe) a été réalisé qui demande à l'utilisateur de définir plusieurs paramètres et, ensuite, sont automatiquement créés:



- le Bloc de Calcul 'External DLL' avec les paramètres convenables.
- les Blocs de Calcul pour les entrées et les sorties s'ils n'existent pas déjà (les noms fournis par l'utilisateur pour les variables sont cherchés dans les Blocs de Calcul existants).
- les fichiers de texte nécessaires pour compiler les bibliothèques externes.

Les combinaisons suivantes de Compilateurs/Langages sont supportés: Microsoft Fortran 5.1, Borland Pascal 7.0 et Borland C++4.5.

Par exemple, en utilisant l'option Fortran, les trois fichiers suivants sont créés:

nom.FOR - programme source de Fortran pour relier avec le bloc 'EXTERNAL DLL'.

nom.DEF - dossier de définition pour instruire le compilateur sur la façon de faire correctement la compilation afin de créer la DLL dynamique.

nom.BAT - si le compilateur Microsoft Fortran est installé, ce fichier batch l'invoque en lui transmettant tous les paramètres nécessaires. Alors, un fichier nom.DLL sera créé par le compilateur pour être dynamiquement relié à ACRUN.

En éditant le sous-programme nom.FOR, l'utilisateur peut insérer son propre code pour réaliser la fonction désirée. Si un compilateur différent d'un des langages ci-

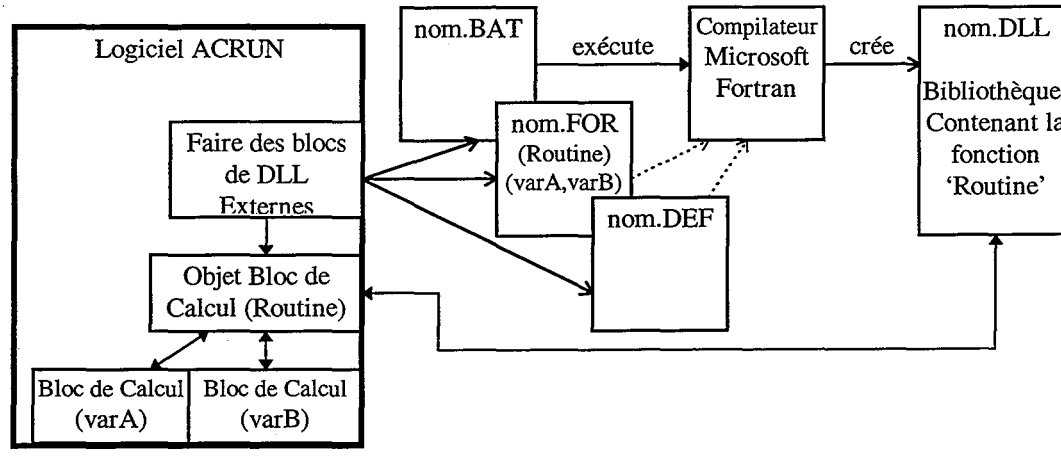


Figure 52 - Création Automatique des DLLs Externes

dessus est employé, l'adaptation des fichiers qui ont été créés ne doit pas être difficile (par exemple, utiliser Visual C++ avec les fichiers créés pour Borland C++).

Le Bloc 'External DLL' et les Blocs de Calcul représentant ses entrées et ses sorties, étant complètement intégrés dans ACRUN, peuvent être représentés dans les graphiques et les compteurs, emmagasinés dans des fichiers sur le disque, et ainsi de suite. La fonction développée par l'utilisateur dans le langage de programmation de son choix est complètement intégrée comme s'il était un Bloc de Calcul standard du programme.

Une problématique autour du Bloc 'External DLL' est le débogage. Etant relié pendant l'exécution et programmé dans un langage choisi, un deboggueur approprié doit être employé. Relier dynamiquement le sous-programme quand il est déjà complètement testé est la meilleure option. L'utilisation de techniques communes d'écriture des variables dans un fichier sur le disque peut aussi être employée fructueusement pour du débogage élémentaire. Le mieux est de planifier attentivement et à l'avance les tests et la stratégie d'intégration à employer [SOLHEIM 93].

Ajout de Blocs Externes à une Bibliothèque

Ayant développé une fonction externe qui n'était pas disponible dans ACRUN, l'utilisateur peut vouloir rendre cette fonction disponible à d'autres, ou, ayant développé toute une gamme de fonctions pour traiter un domaine scientifique spécifique, il peut être tenté de vendre ce savoir-faire sous forme d'une bibliothèque de fonctions.

ACRUN peut être le type d'environnement décrit dans [PERRIS 94] où les utilisateurs peuvent construire de nouveaux modèles d'unités de procédé ou faire la combinaison de phénomènes, en les rassemblant dans une bibliothèque de sous-unités. Pour que cela puisse avoir lieu, la bibliothèque correspondant aux modèles phénoménologiques de bas niveau doit être disponible, donc, construite auparavant.

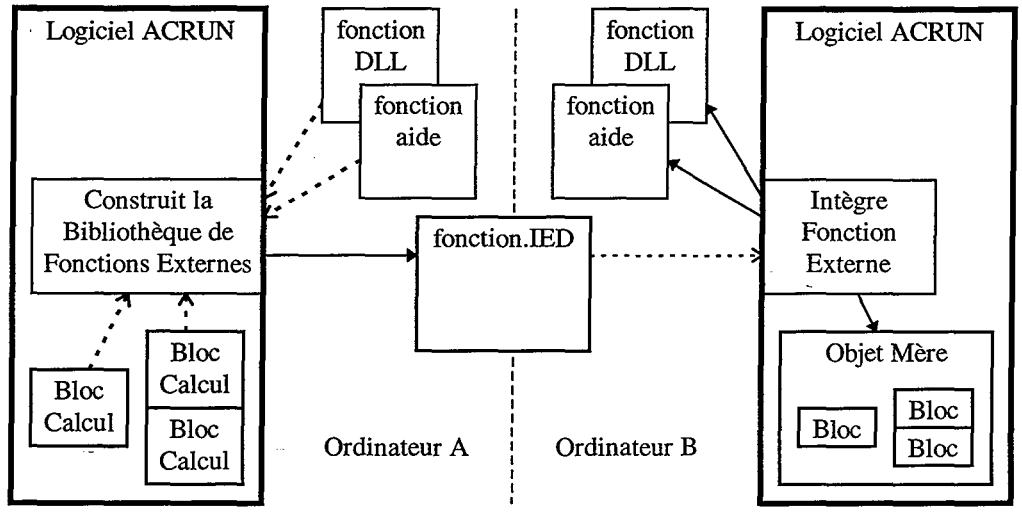


Figure 53 - Création et Importation de Bibliothèques de Fonctions

Cela peut être fait en utilisant le menu 'Create External Library'. L'utilisateur définit le nom de la DLL externe qu'il veut intégrer, et le titre du sujet dans lequel il doit paraître (voir l'annexe A). Si ce titre du sujet n'existe pas, alors il sera créé et ajouté aux autres. Ensuite pour l'utilisateur, lors de l'accès au menu d'une nouvelle fonction, il verra un autre titre de sujet (s'il était créé) et peut sélectionner la nouvelle fonction intégrée. Contrairement aux fonctions ordinaires qui consistent en un bloc, la fonction Externe créée de cette façon consiste en un objet Mère regroupant ensemble le Bloc Externe et tous les blocs pertinents des entrées et des sorties.

Chacune de ces nouvelles fonctions DLLs externes sera gardée dans un fichier séparé avec l'extension IED emmagasinant toute l'information nécessaire: le nom, le sujet et la description de la fonction, le nom et le nombre des entrées et des sorties, le texte d'aide et une image s'il sont disponibles. Il inclura aussi le fichier binaire DLL, donc distribuer les fichiers IED est suffisant pour partager toute l'information nécessaire pour représenter et employer la fonction dans un autre ordinateur.

IV.2/ Echange de Données avec DDE

Une autre caractéristique standard de Windows est la DDE (Dynamic Data Exchange) qui permet à des applications différentes de partager des données entre elles. Elle peut être employée pour échanger du texte, des images ou d'autres types de données de façon facile. La liaison initiale est faite par le Presse-Papiers de Windows. Des commandes 'Copier' dans beaucoup de programmes peuvent être employées pour emmagasiner de l'information dans le presse-papier, des données qui peuvent être partagées en utilisant la DDE. Alors un autre programme utilisant des commandes 'Coller' ou 'Coller Spécial' lit les données du presse-papiers et initie une 'conversation' avec le programme initial pour décider comment à l'avenir plus de données peuvent être transférées.

Dans ACRUN un objet peut être traîné et lâché sur l'icône du Presse-Papiers en copiant vers le presse-papiers les données suivantes: un méta-fichier avec l'image de l'objet, un bitmap de l'image de l'objet, une description du texte de l'objet, information disant que ACRUN est le programme qui a placé les données dans le presse-papiers et de l'information concernant l'objet dans ACRUN d'où provenaient ces données. En utilisant un autre programme, par exemple Word pour Windows, si la commande 'coller spécial' est sélectionnée alors un des trois formats de données peut être choisi (méta-fichier, bitmap ou texte) et la liaison DDE avec ACRUN peut être établie. En ayant choisi le format méta-fichier, par exemple, chaque fois que des changements dans l'objet correspondant d'ACRUN prennent place, l'image sera aussitôt mise à jour dans le document de Word.

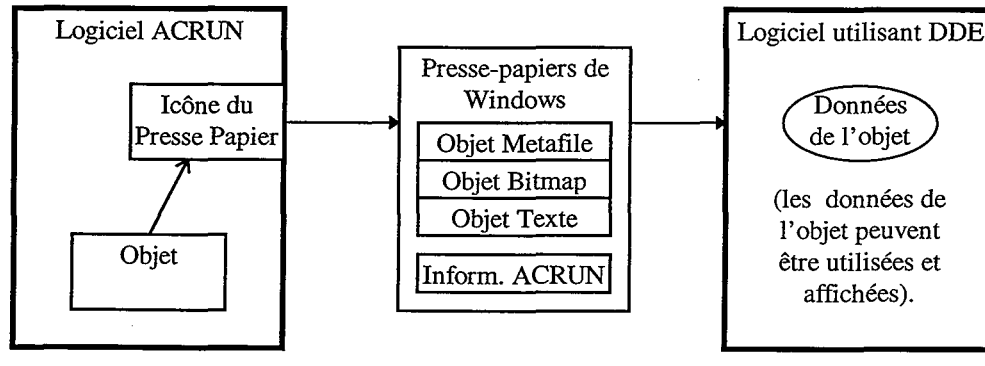


Figure 54 - Exportation de Données DDE de ACRUN

Des rapports peuvent être écrits aisément si, dans un certain document, des données DDE sont insérées et si elles sont toujours en date. Pour enregistrer le statut d'un procédé dans l'usine ou le résultat d'une simulation dans un rapport standard, l'utilisateur doit simplement garder ce document en lui donnant à chaque fois un nom différent (si les données DDE se veulent statiques, il faut désactiver les liaisons).

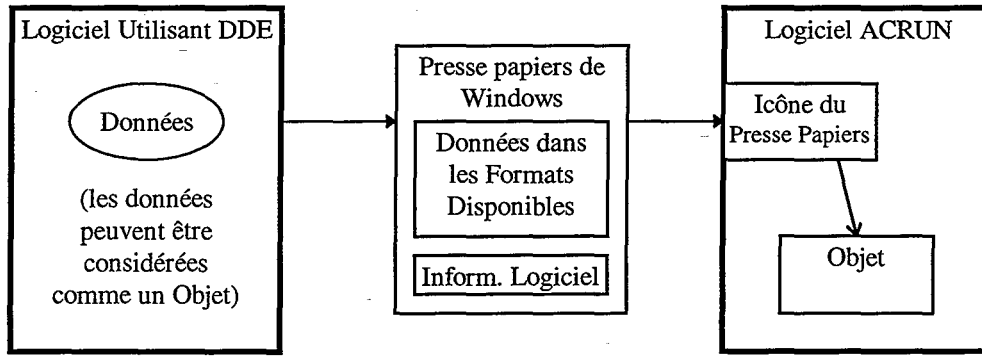


Figure 55 - Importation de Données DDE dans ACRUN

Intégrer les données d'un autre programme dans ACRUN est aussi possible. A partir du programme où les donnés sont créées, copier vers le Presse-Papiers les données qui puissent être partagées par DDE. Alors, dans ACRUN, on glisse l'icône du presse-papiers et on le lâche sur un objet. Un menu demandera lequel des formats disponibles dans le presse-papiers devrait être importé et, si l'information DDE existe, il est demandé à l'utilisateur s'il veut permettre la liaison. En faisant cela, des graphiques d'un autre programme peuvent être représentés dans un objet d'ACRUN avec sa mise à jour automatique chaque fois que le programme original les modifie.

Si le texte d'un autre programme est relié à un objet Bloc de Calcul en utilisant le DDE, il sera alors converti en valeur réelle représentant la sortie du bloc. Une cellule spécifique dans une feuille de calcul peut ainsi transmettre sa valeur à un Bloc de Calcul dans ACRUN et vice-versa. Si une fonction est créée dans la feuille de calcul, une autre manière de réaliser des algorithmes externes à ACRUN consiste alors à utiliser le DDE. Les valeurs d'ACRUN sont transférées à la feuille de calcul, puis elles sont utilisées dans des calculs, et les nouvelles données dans les cellules de résultat sont transferées à ACRUN. Utiliser une feuille de calcul de cette façon est un exemple commun de l'emploi de DDE [PREECE 94].

En raison de la lenteur et de la réalisation asynchrone de la mise en oeuvre des liaisons DDE, la méthode décrite n'est pas efficace en termes de rapidité mais peut être de grande utilité dans certaines circonstances.

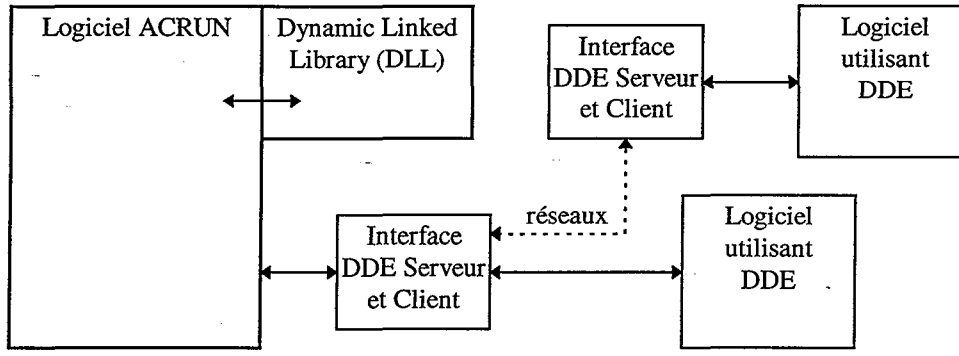


Figure 56 - Liaison Dynamique versus DDE

Un autre avantage majeur de la communication DDE est qu'il n'est pas nécessaire que les deux programmes s'exécutent dans la même machine. En utilisant Windows pour Workgroups, Windows NT ou Windows 95, où le DDE en réseau est permis, une toute nouvelle approche de la surveillance de procédé, commande et sécurité de procédé peut être réalisée. En ce qui concerne la réalisation d'ACRUN dans des réseaux, il faut consulter l'Annexe F - Opération et Supervision en Réseau.

Bien que la DDE soit beaucoup plus flexible et puissante que les fonctions DLLs externes, la réalisation de fonctions externes avec des DLL est plus efficace parce qu'elle implique beaucoup moins de travail du système, appelant directement le sous-programme désiré (voir Figure 56). Quand des images requièrent d'être partagées ou que des données doivent être échangées par des réseaux, alors le DDE est le meilleur choix parce que toutes les complexités inhérentes au protocole sont prises en charge par le système Windows.

IV.3/ L'Interprète de Lignes de Commande

Même si le concept de bloc est flexible et intuitif pour réaliser la plupart des algorithmes fondamentaux employés dans la commande, il est toutefois nécessaire de traduire des fonctions du papier en leur équivalent en Blocs de Calcul d'ACRUN, avec les raccordements adéquats entre eux. Cela peut être un travail avec beaucoup de possibilités d'erreur parce qu'il y a beaucoup de détails qui peuvent être oubliés ou dénaturés comme c'est le cas des priorités d'opération, des niveaux des parenthèses, des retards de signal, et des opérations matricielles complexes.

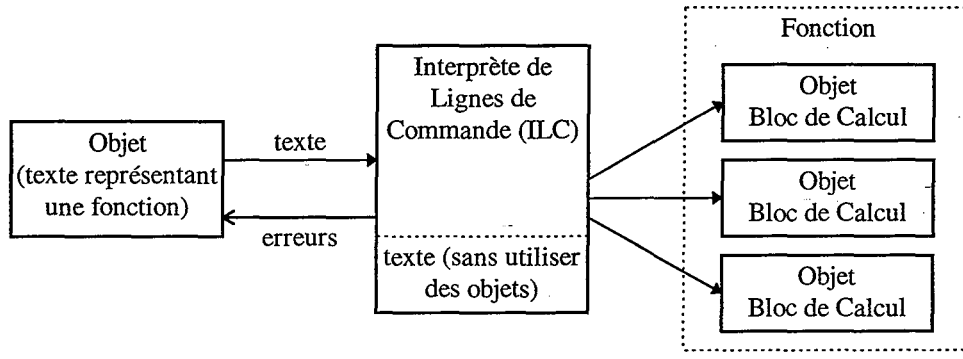


Figure 57 - Création de Fonctions par l'Interprète de Lignes de Commande

Dans ACRUN, existe un menu où les fonctions peuvent être introduites presque comme elles sont écrites sur le papier et converties alors en l'équivalent en Blocs de Calcul et leur raccordements respectifs. Ce menu est appelé l'Interprète de Lignes de Commandes (ILC). Par exemple, si la fonction est G = (2 * x+y) / z, alors des Blocs de Calcul seront créés réalisant les variables - G, x, y, z - et les fonctions - * , + , / . Si aucun des noms des variables de ces quatre Blocs de Calcul n'existe déjà, alors l'utilisateur peut choisir de les employer ou de créer un nouveau Bloc.

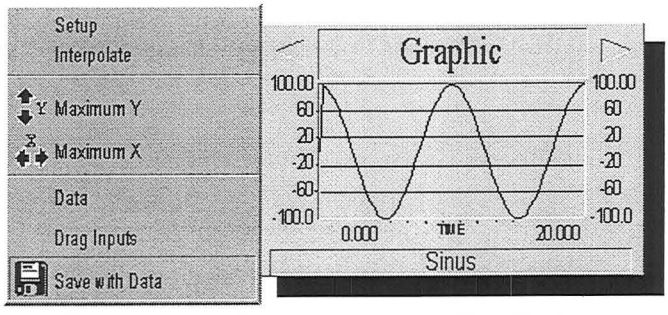
Des combinaisons d'opérateurs scalaires, opérateurs matricielles et vectorielles peuvent être utilisées avec le ILC, ainsi que les opérations plus communes employées pour traiter avec eux: +, -, *, /, MOD, transpositions de matrice, opérations algébriques entre des scalaires et des matrices, et ainsi de suite.

Des niveaux de parenthèse illimités sont acceptés ainsi que les indications de retards des signaux. En chapitre VI, 'Dans la Classe de Cours', un exemple est donné sur un emploi réel du CLI.

Le texte à traiter par le ILC peut venir d'un objet (les objets de texte conviennent particulièrement pour cela) ou d'un fichier sur le disque. Avant que tous les blocs soient créés, la ligne de commande est complètement vérifiée pour des erreurs de syntaxe. Si une erreur est trouvée, sa position est indiquée et un message explicatif est montré pour faciliter la correction.

Une syntaxe spécifique d'ACRUN ayant été réalisée il est souhaitable de la rendre aussi compatible que possible avec les 'presque standards' de l'industrie comme la syntaxe dans les programmes Matrix et Matlab.

IV.4/ Gestion des Données et des Graphiques



Dans le développement de la commande, beaucoup de simulations sont faites pour tester les algorithmes qui sont développés. Les données résultant de ces simulations doivent souvent être sauvegardées sur le disque pour être employées ultérieurement ou comparées à d'autres données des simulations.

L'approche qui a été prise pour assister l'utilisateur dans cette tâche a été centrée sur les objets Graphiques. Cela est dû au fait qu'habituellement les chercheurs observent et emploient des graphiques avec les données pour comparer les simulations et non les données elles-mêmes.

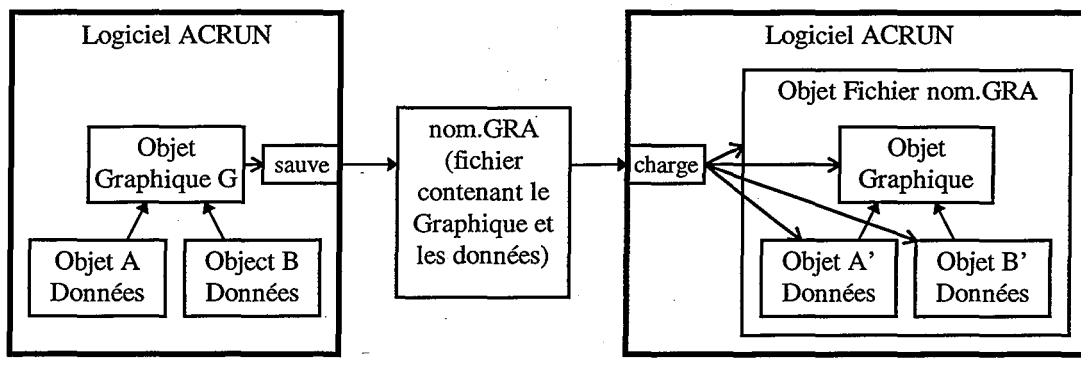


Figure 58 - La Gestion de Données Utilisant les Graphiques

Chaque objet Graphique peut être conservé dans un format spécial dans un fichier sur le disque avec l'extension GRA . Ce fichier garde non seulement une réplique du graphique, mais il crée également des Blocs de Calcul avec les données représentées sur ce graphique. Ces Blocs de Calcul sont sauvegardés avec toutes les données présentes dans leurs tampons, pour que, non seulement l'objet Graphique puisse être redessiné mais aussi pour permettre davantage de manipulation des données. Cela signifie que, quand l'utilisateur sauvegarde un Graphique en utilisant l'option SAVE de son menu local, il garde toutes les données représentées dans le graphique dans un format de Bloc de Calcul qui peut être employé, édité et manipulé à l'avenir.

Non seulement cette approche est intuitive mais elle est aussi très puissante pour gérer de grandes quantités de données. En lisant les fichiers graphiques GRA, ACRUN crée un objet Fichier qui lui est associé. La date et l'heure auxquelles le graphique était sauvegardé faisant partie du fichier créé, il est facile en utilisant les fonctions et objets de date du programme ACRUN de chercher des simulations faites à certaines dates. Cela constitue un moyen supplémentaire de chercher l'information dans ACRUN, par la date associée à chaque objet. Tous les objets dont la date est située avant, entre ou après certaines dates peuvent être trouvés. Enfin, ils

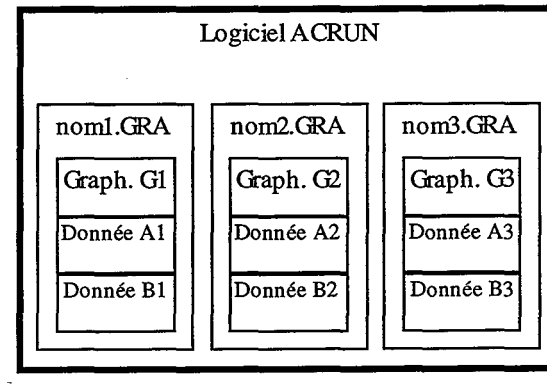


Figure 59 - Fichier avec des Données des Simulations

peuvent aussi être cherchés en utilisant les objets Agenda ainsi qu'avec les autres objets qui traitent avec des dates.

IV.5/ Le Traitement Batch Automatique

On a mentionné auparavant que les simulations de commandes peuvent entraîner des calcul longs, même dans les ordinateurs les plus rapides actuellement disponibles. Depuis longtemps possible sur les stations de travail fonctionnant sous Unix, le calcul batch n'est pas encore arrivé aux PCs DOS/Windows, au moins comme une caractéristique ordinaire. Même dans les stations de travail Unix pour chaque simulation en batch, l'utilisateur doit changer manuellement les paramètres et prendre soin de la manière avec laquelle les résultats sont sauvegardés pour que, à la fin, tous ces résultats puissent être disponibles sur le disque. Le besoin de gérer les combinaisons de paramètres mène facilement à la frustration avec des simulations en batch, parce que la probabilité que des erreurs surviennent est grande.

Dans ACRUN, le calcul avec batch automatisé essaye de résoudre ces problèmes:
- laisser l'ordinateur travailler seul en faisant plusieurs simulations sans l'intervention de l'utilisateur,
- changer automatiquement les paramètres avant chaque simulation,
- sauvegarder les résultats de chaque simulation dans une séquence ordonnée de fichiers.

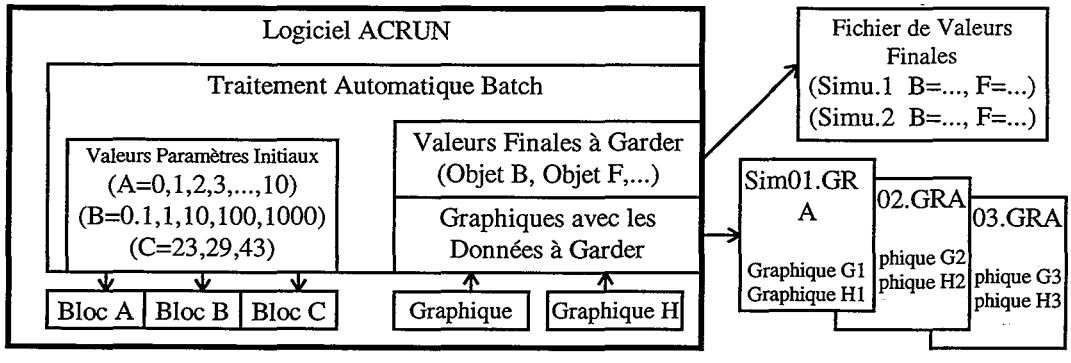
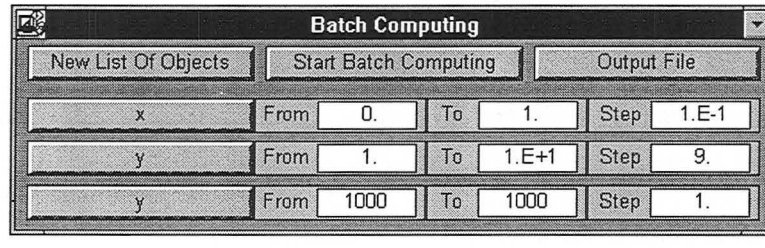


Figure 60 - Traitement Automatique Batch avec la Gestion de Données

Pour ce faire, l'utilisateur définit les paramètres qui changent et les valeurs qu'ils prennent. Par exemple, le paramètre P1 va de 0 à 1 par intervalle de 0.1 et le paramètre P2 prend les valeurs 10, 100 et 1000 afin que 33 (11 * 3) simulations soient faites avec toutes les combinaisons possibles des paramètres. Il n'y a pas de limite au nombre de paramètres ou à la quantité de valeurs différentes qu'ils prennent. Après chaque simulation, une estimation est montrée du temps nécessaire pour finir toutes les simulations (il indique aussi l'heure et le jour où cela se terminera).



L'utilisateur définit aussi quelles valeurs finales des Blocs de calcul doivent être enregistrées à la fin de chaque simulation ainsi que les objets Graphiques qui doivent être sauvegardés. Comme cela a été établi dans le sous-chapitre précédent, en sauvegardant un Graphique, les données représentées sur ce dernier sont aussi sauvegardées dans des Blocs de Calcul. Le fait de sauvegarder les Graphiques après chaque simulation dans des fichiers séparés garantit que toutes les données nécessaires seront disponibles quand toutes les simulations sont terminées. Le nom du fichier sur le disque où les Graphiques sont conservés est défini par l'utilisateur, et sera par défaut BATCH.GRA (chaque simulation successive créera alors les dossiers BATCH001.GRA, BATCH002.GRA, et ainsi de suite).

Graphiques en Deux et en Trois Dimensions

Dans les simulations en batch, les valeurs initiales et finales varient pour chaque simulation. Il est utile d'avoir des graphiques traçant les résultats finaux des simulations en fonction de ces paramètres. Par exemple, si la valeur finale est l'énergie totale ET dépensée dans l'ouverture et fermeture d'une soupape, et si les paramètres variants sont les valeurs de P, de I et de D dans la commande par PID, il pourrait être intéressant de visualiser graphiquement la relation entre les trois paramètres et ET.

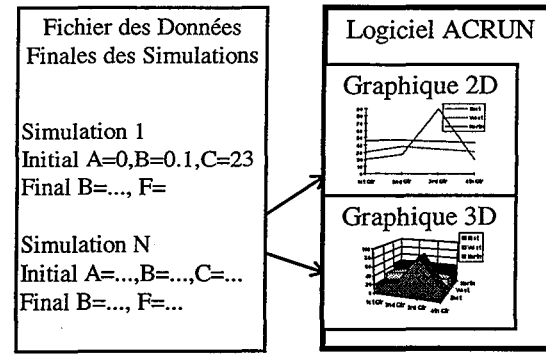


Figure 61 - Graphiques des Simulations

Dans le menu des Graphiques 2D et 3D pour les simulations en batch, les options suivantes sont disponibles: 1) tracer dans le graphique bidimensionnel les valeurs des sorties en fonction d'un des paramètres qui varient; 2) tracer dans un graphique tridimensionnel les valeurs de sortie en fonction de deux des paramètres qui varient.

Dans l'exemple donné, ET peut être représenté en fonction de P pour certaines valeurs pour de I et de D, en créant un graphique bidimensionnel.
On peut aussi représenter ET en fonction de I pour certaines valeurs constantes de P et de D, en créant un graphique bidimensionnel.
On peut aussi représenter ET en fonction de P et de D en fixant une certaine valeur pour I, créant un graphique tridimensionnel.

Comme tous ces graphiques peuvent être visualisés en même temps sur l'écran, une bonne image de la corrélation des valeurs des paramètres à leurs sorties est disponible. Le nombre de paramètres et le nombre de sorties sont définis dans le menu de Batch Automatique. Quand des simulations très longues sont mises en marche, il convient de garder toutes les valeurs possibles des sorties qui peuvent présenter de l'intérêt parce qu'elles ne prennent même pas beaucoup d'espace sur le disque.

Pour le développeur de la commande, cette visualisation graphique des simulations batch multiples est de grande aide dans la compréhension du procédé et par conséquent pour régler les paramètres impliqués. Les graphiques 2D et 3D peuvent aussi être employés pour expliquer les compromis que le choix d'un certain ensemble de paramètres implique.

Enfin, des animations 2D et 3D peuvent être exécutées avec plusieurs avantages [CHATTY 94] comme l'analyse de l'influence de la variation d'un paramètre sur des variables choisies. Pour un graphique 2D, un troisième paramètre doit être choisi et pour les graphiques 3D, un quatrième paramètre doit être choisi pour réaliser l'animation.

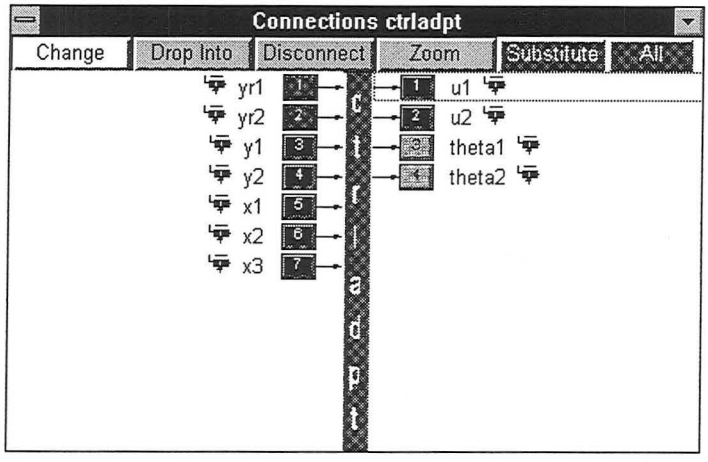
IV.6/ Visualisation et Contrôle des Flux de Signaux

Pour les dévelopeurs de commandes réalisant beaucoup de raccordements entre les Blocs de Calcul, le besoin existe de vérifier si le comportement du signal du procédé est correct. Pour étudier la robustesse d'une commande, par exemple, qui se fait normalement dans le domaine fréquenciel, chaque bloc peut être considéré comme un filtre et c'est l'analyse des flux de signaux, de la consigne jusqu'à la sortie, qui est souvent employée pour tirer des conclusions.

Quand des Blocs de Calcul sont reliés, la représentation des raccordements apparaît sur l'écran. Toutefois, ce n'est pas la meilleure méthode pour visualiser des flux de signaux parce qu'il y a trop de cas où les raccordements à l'écran sont difficiles à suivre ou des cas où ils ne sont même pas représentés (objets dans des écrans virtuels différents, objets groupés dans des objets Mère, et ainsi de suite).

Une façon plus pratique et sûre de visualiser les flux de signaux existe dans ACRUN. Chaque objet a un menu de flux de signaux où son nom est écrit au milieu, tous les

objets qui servent d'entrées sont représentés sur la gauche et tous les objets auxquels il débite une valeur sont représentés à droite. Dans ce menu, non seulement les Blocs de Calcul sont représentés mais encore toute autre sorte d'objets.



Si l'utilisateur presse le bouton de la souris dans un des objets des entrées ou dans une des sorties, alors il devient l'objet central et ses propres objets d'entrée et de sortie sont visualisés respectivement à sa gauche et à sa droite. De cette façon, il est non seulement facile de visualiser toutes les interactions entre les objets, mais il est également simple de suivre le flux de signaux.

Entrant dans le menu de flux de signaux pour un Graphique, les objets en lui représentés seront montrés sur le côté gauche (normalement des Blocs de Calcul). Alors n'importe lequel de ces objets peut être sélectionné et ses raccordements visualisés, et ainsi de suite.

Comme beaucoup d'objets peuvent être présentés - Alarmes, Compteurs, Graphiques, Cartes d'Acquisition - le choix des types d'objets à représenter sur les menus de l'organisation de flux de signaux peut aussi être fait. Un choix courant en contrôle, que les développeurs emploient, est de visualiser seulement les objets Blocs de Calcul. Finalement, n'importe lequel des objets représentés sur le menu peut être agrandi sur l'écran pour que son état ou ses paramètres puissent être vérifiés ou changés. Les aptitudes d'ACRUN en visualisation et en suivi des flux de signaux constituent une de ses caractéristiques les plus importantes pour le développement de la commande.

CHAPITRE V - La Simulation et l'Acquisition en Ligne

V.0/ Introduction

Dans le développement de la commande et sa mise en oeuvre sur le procédé, trois phases majeures sont présentes:

1. **L'Acquisition** - l'entrée des mesures du procédé.
2. **Le Calcul** - le calcul lié à la simulation du procédé et aux algorithmes de commande.
3. **Commande** - le débit des données vers le procédé.

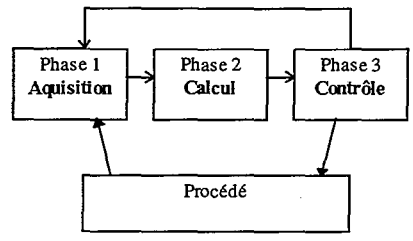


Figure 62 - Phases de Contrôle de Procédé

Quelquefois, une des phases n'est pas présente. Si uniquement la supervision du procédé est requise, alors seulement l'acquisition est nécessaire.

Quand ces trois phases sont exécutées, elles se succèdent sur plusieurs cycles (ou bien, comme c'est le cas dans certaines usines, dans des cycles sans fin). Chaque cycle peut être appelé le 'cycle de commande' du procédé. Ces cycles se produisent à des intervalles réguliers de temps qui correspondent à la période d'échantillonnage du procédé. L'inverse de la période d'échantillonnage est la fréquence d'acquisition. Dans le contrôle des procédé chimiques, comme les constantes de temps sont généralement élevées, la période d'échantillonnage peut atteindre une heure et rarement moins de 1 seconde.

Idéalement, les trois phases s'exécutent exactement au même instant d'échantillonnage, parce que les équations de commande traitent les données exactement aux mêmes instants. Ce n'est pas ce qui se produit dans la réalité. Quand chaque cycle commence, la phase 1 débute; quand elle finit, la phase 2 s'exécute, et à la fin la phase 3 a lieu.

Cet ordre peut sembler logique - acquérir les données, manipuler les données, et débiter les résultats - mais ce n'est pas toujours ce qui convient le mieux. Quand on utilise des cartes d'acquisition et de contrôle, les phases 1 et 3 prennent seulement une fraction de seconde, même quand un grand nombre d'entrées et de sorties sont présentes. La phase 2, au contraire, impliquant souvent des calculs itératifs lourds, des calculs matriciels d'ordre élevé, peut consommer beaucoup de temps. De ce fait, la phase 3 a lieu avec un certain retard par rapport à la phase 1. Si la période d'échantillonnage (fonction de la dynamique de procédé) est beaucoup plus longue que le retard de la phase de calcul, alors ce retard peut être négligé. Sinon, une séquence différente des phases doit être réalisée. Lors du développement de la commande, ce problème ne survient jamais, parce que, quand les simulations sont en cours, un temps fictif est calculé.

Ce chapitre décrira les diverses implications et les différences entre la simulation et l'acquisition en ligne, les problèmes qui surviennent, comment ils sont résolus dans ACRUN et comment l'utilisateur peut agir sur eux.

V.1/ Systèmes Multitâches et Temps Réel

Comme on vient d'en discuter dans l'introduction, la fréquence d'échantillonnage joue un rôle très important pour la commande de procédé. L'exactitude de chaque temps d'échantillonnage est essentielle pour la cohérence entre la commande du procédé et les équations employées pour l'identifier, pour le simuler, et pour le commander. Un algorithme d'identification attend les données du procédé également espacées à des temps de saisie précis, donc si les instants d'échantillonnage ne sont pas respectés, alors les paramètres trouvés pour le modèle peuvent être erronés.

La précision de la fréquence d'échantillonnage est un problème qui n'existe pas en simulation. Quand aucune interaction avec le procédé réel n'a lieu, plus les calculs seront exécutés rapidement, mieux cela sera. Dans ce cas, le temps est aussi simulé et employé n'importe où c'est nécessaire, et cela n'a rien à voir avec une horloge temps

réel. Il peut être plus rapide ou plus lent que le procédé réel, selon les algorithmes exécutés et l'ordinateur employé.

Les ordinateurs sont constitués de circuits intégrés, qui, pour fonctionner et échanger des données, doivent être synchronisés par des signaux haute fréquence. Avec des valeurs de fréquence du domaine du megahertz, ces signaux peuvent aussi être employés par des horloges en temps réel et par les systèmes d'exploitation pour réaliser des commandes logiciel 'retard' et 'pause'. Le système d'exploitation gère l'équipement, notamment l'emploi des cycles du processeur, pour offrir des services au logiciel qui s'exécute.

Deux points importants sont:

- **les opérations multitâches**: plusieurs tâches peuvent s'exécuter simultanément dans le même ordinateur.
- **les listes d'interruptions**: permettre aux programmes de définir quand et avec quelle priorité une interruption doit avoir lieu, sachant que son action fait qu'un certain sous-programme commence à s'exécuter.

Multitâches

Il existe deux sortes de multitâche sur les ordinateurs: multitâches coopératif et le multitâches préemptif.

Dans le multitâches coopératif, chaque programme qui s'exécute doit de temps en temps céder le contrôle de l'exécution aux autres programmes. Plus souvent cela se produit, plus on aura la sensation que les programmes s'exécutent en même temps. Le problème avec ce multitâche, c'est qu'il n'y a aucune garantie pour éviter qu'un programme ne se comporte pas mal, et que, de temps à autre, il garde le processeur pour son emploi exclusif pour plusieurs secondes dans une rangée. Quand cela arrive, les autres programmes ne peuvent plus s'exécuter, même pas le système d'exploitation.

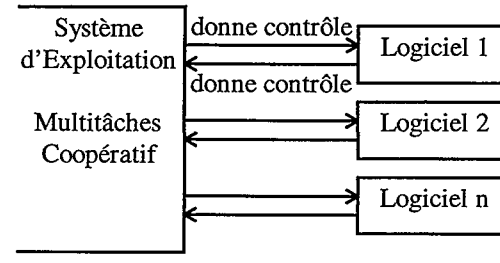


Figure 63 - Multitâches Coopératif

Le multitâche coopératif ne convient pas pour l'acquisition et le contrôle parce qu'il ne peut pas disposer d'un maximum de temps pour lequel il soit certain que le programme soit exécuté au moins une fois.

Le multitâche préemptif présente une approche différente pour partager les ressources de l'ordinateur, notamment de son processeur, parmi les programmes qui s'exécutent simultanément. A certains moments, le système d'exploitation interrompt le programme qui s'exécute et donne le contrôle à un autre programme. En ayant tous les privilèges, le système d'exploitation possède un grand contrôle des programmes et il peut garantir dans une période T que chaque programme aura au moins une fois la possibilité d'être exécuté. La valeur de T dépend du système d'exploitation, de la vitesse de l'ordinateur et rarement de la quantité de programmes qui s'exécutent simultanément.

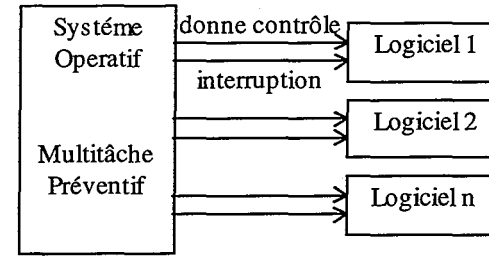


Figure 64 - Multitâche Préemptif

Temps Réel

La liste d'interruptions est également importante. Dans un programme d'acquisition, il est nécessaire qu'à exactement chaque instant d'échantillonnage, le sous-programme responsable du cycle de contrôle du procédé puisse commencer à s'exécuter. Si l'équipement de l'ordinateur et/ou le système d'exploitation nous permettent de le faire, alors l'opération temps réel peut s'effectuer. Dans la pratique, un programme demande au système d'exploitation une interruption périodique (par exemple, chaque 2 secondes) avec une haute priorité et l'association au sous-programme du cycle de contrôle. Au moment venu, le système d'exploitation doit interrompre toutes les tâches avec une priorité inférieure pour servir cette interruption.

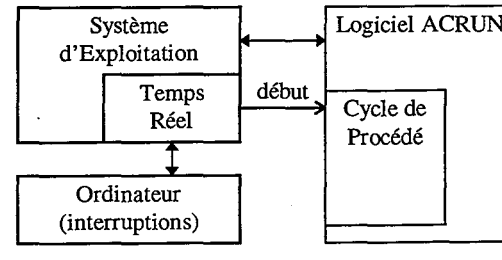


Figure 65 - Opération Temps Réel

Des systèmes d'exploitation Temps Réel conviennent à l'exécution des programmes de commande et d'acquisition parce que, si une interruption est demandée pour un certain temps, alors, avec un retard maximum connu à l'avance, il y a la garantie qu'elle se produira. Par exemple, si une certaine période d'échantillonnage est définie et si le retard maximum garanti est de 10 millisecondes, alors il est certain que l'écart par rapport au temps exact se situera entre 0 et 10 millisecondes.

Puisque le système d'exploitation DOS/Windows n'offre ni un environnement temps réel, ni un multitâche préemptif, comment un programme d'acquisition et contrôle peut-il être réalisé?

Des solutions non standard sont disponibles, bien sûr, comme intercepter et programmer les interruptions de l'horloge système de l'ordinateur. Ces options n'ont toutefois pas été réalisées pour les raisons suivantes:

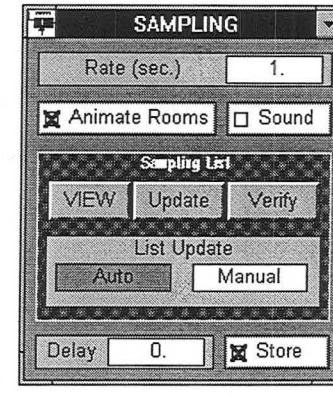
- étant non standard, elles pourraient ne pas fonctionner avec d'autres versions présentes et futures du système d'exploitation et, certainement, elles ne travailleront pas dans des plates-formes d'équipement différent (comme sur les ordinateurs avec les microprocesseurs Alpha, PowerPC et Mips exécutant Windows NT).
- elles peuvent modifier l'exécution d'autres programmes, par exemple de multimédia, requérant aussi des chronométrages précis
- cela implique une quantité considérable de travail et de tests pour aboutir à une solution loin d'être idéale.

Comme il existe déjà des versions de Windows qui sont préemptives et en temps réel, comme Windows NT et Windows 95 (pour les programmes 32 bits), il paraîtrait séduisant de les employer pour exécuter ACRUN. En fait, il n'y a pas de version appropriée du compilateur de langage employé pour tirer parti des nouvelles caractéristiques qui sont d'intérêt pour ACRUN (compilateur pour 32 bits). La deuxième raison est que, même si Windows NT version 3.51 agit en multitâche d'une façon préemptive pour les programmes de 16 bits, il n'est souvent pas disponible dans les laboratoires et dans les universités parce que c'est une solution plus coûteuse. Ces deux

problèmes pourraient être résolus en utilisant Windows 95 (multitâches préventif et mise en oeuvre du temps réel, mais requérant moins d'équipement que Windows NT) et une nouvelle version du compilateur Pascal dont la sortie est annoncée pour début 1996.

Les approches pour l'opération temps réel dans des systèmes non préemptifs sont discutées dans [YUAN 94].

V.2/ Paramètres d'Echantillonnage



Voyons quels sont les paramètres d'échantillonnage disponibles et comment ils sont réalisés:

- **Rate** (Taux) - période d'échantillonnage en secondes à employer à la fois pour la simulation et pour l'acquisition et contrôle.
- **Delay** (Retard) - secondes avant le moment d'échantillonnage où le contrôle de l'exécution du programme est conservé.

Comme cela a déjà été indiqué, on n'a pas de garantie en utilisant DOS/Windows sur le temps qui s'écoulera avant que le programme puisse à nouveau être exécuté. Cela peut causer des écarts dans les temps d'échantillonnage. Pour réduire ce risque, un intervalle de temps avant le temps d'échantillonnage peut être établi, par exemple de deux secondes, dans lequel, si ACRUN prend contrôle de l'exécution, alors il le maintiendra jusqu'à ce que le nouvel instant d'échantillonnage arrive.

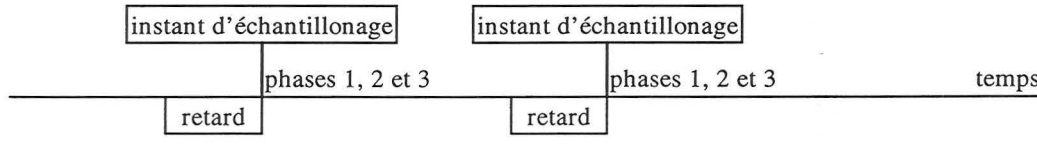
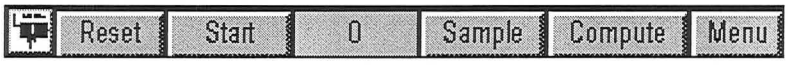
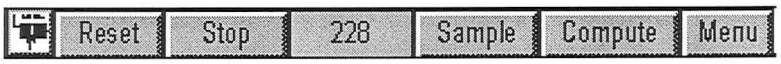


Figure 66 - Retard pendant que l'Exécution du Programme sera Conservée

Cela ne peut pas garantir un échantillonnage parfait parce qu'un autre programme peut garder le contrôle de l'exécution pendant plus de deux secondes, ce qui fait que, quand ACRUN arrive à pouvoir s'exécuter, le moment d'échantillonnage peut déjà être passé. Dans Windows NT version 3.5 ou ultérieure, ce problème a été partiellement résolu parce que les applications 16 bits (comme ACRUN développé pour Windows 3.1) peuvent être exécutées dans des espaces séparés de mémoire avec du multitâches préemptif entre elles.



Pour commencer l'acquisition, l'utilisateur doit seulement presser le bouton *Start* (démarrer). Un compteur d'échantillonnage existe, initialement mis à zéro, qui s'incrémente un par un pour chaque cycle de contrôle. En réalité, en faisant la simulation ou l'acquisition en ligne, c'est ce compteur multiplié par la période d'échantillonnage qui est employé pour calculer le temps actuel. Ce temps est ensuite employé par les Blocs de Calcul, par les Graphiques, et par les autres objets. C'est aussi ce compteur qui définit quand les Blocs de Calcul commencent à calculer, quand les objets Fichiers doivent garder des données sur le disque et quand les objets doivent être mis à jour sur l'écran (voir section 'Mise à Jour des Objets' dans ce chapitre).



L'échantillonnage se termine quand le même bouton, qui indique maintenant Stop (arrêter), est pressé à nouveau. Les prérogatives de l'utilisateur peuvent être établies pour restreindre l'opération du bouton *Start/Stop*. L'acquisition peut être recommencée ou mise à zéro pour que le compteur d'échantillonnage soit mis à zéro. Cela peut être important parce que les Blocs de Calcul ont un paramètre, 'Freeze Till', qui contrôle le nombre d'échantillons (0 par défaut) avant qu'ils soient opérationnels.

Il y a deux Blocs de Calcul spéciaux avec des valeurs de sortie qui sont la période d'échantillonnage et le compteur d'échantillonnage. Les autres blocs peuvent lire leur sorties, ce qui peut être particulièrement utile pour qu'un sous-programme DLL externe

puisse s'ajuster automatiquement à n'importe quel moment où la fréquence d'échantillonnage est changé par l'utilisateur.

Quand l'échantillonnage ou la simulation commencent, des erreurs peuvent être détectées comme le cas de fichiers qui n'existent pas, de cartes d'acquisition qui ne sont pas bien initialisées, etc. . L'utilisateur peut alors choisir d'ignorer ces erreurs ou d'arrêter l'exécution d'ACRUN et consulter une liste des erreurs qui surviennent. Pour chaque erreur, il y a habituellement une explication de sa cause, l'indication de l'objet concerné et parfois une indication des solutions possibles. Quand la simulation ou l'échantillonnage commencent, les dimensions des matrices sont aussi vérifiées et sont automatiquement adaptées les unes aux autres si un conflit existe entre elles. Si une incohérence toutefois persiste, alors l'utilisateur peut utiliser un menu spécial pour voir quelles sont les matrices qui s'opposent et pourquoi.

Une simulation en cours peut être interrompue à tout moment. Quand la simulation se termine, elle peut aussi être continuée si davantage de points d'échantillonnage semblent nécessaires. Pour faire cela, l'utilisateur désactive l'option de mise à zéro pour que les échantillons suivants puissent être calculés.

V.3/ L'Ordre de l'Acquisition, du Calcul et de la Commande

Comme cela a été énoncé dans l'introduction, l'acquisition des variables du procédé devrait être réalisée en même temps que l'information de sortie de la commande vers le procédé. Quand la phase 2 - le calcul des algorithmes - prend un temps non négligeable, alors l'ordre des trois phases doit être changé (voir Figure 67). Si les phases 1 et 3 se déroulent plus rapidement que la phase 2, pourquoi ne pas débuter par la phase 3 à chaque moment d'échantillonnage, en débitant les données de commande calculées sur les données du cycle précédent, en procédant à la phase 1, acquisition des nouvelles données du procédé, et en calculant enfin dans la phase 2 les prochaines données de commande. De cette façon, il n'est plus important que la réalisation de la stratégie de commande prenne une ou dix secondes de calcul pourvu que la somme des temps des

phases 1 à 3 soit inférieure à la période d'échantillonnage. La possibilité existe dans ACRUN de réaliser les deux séquences différentes des trois phases, chacune d'elles impliquant que les algorithmes de commande soient conformément ajustés.

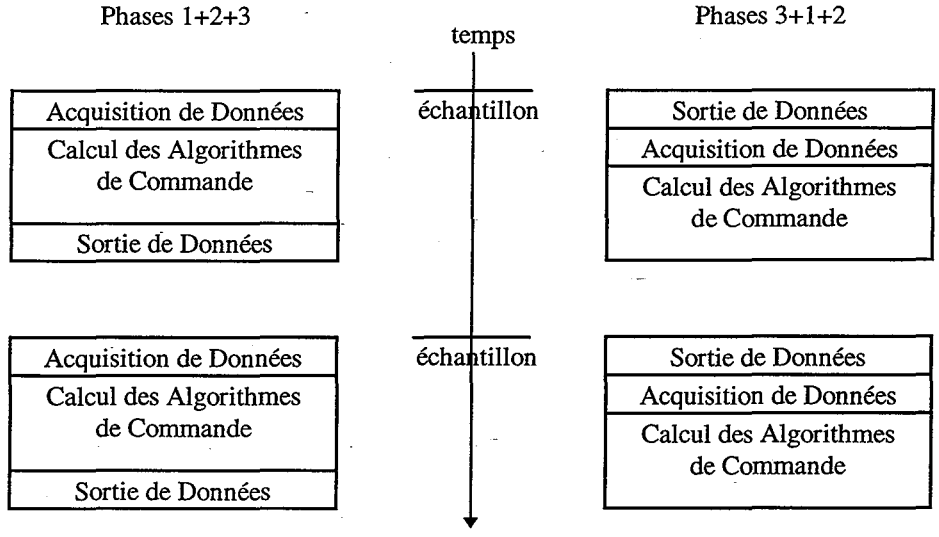


Figure 67 - Séquences d'Echantillonnage du Cycle de Procédé

Ordonnancement des Blocs de Calcul: Automatique et Manuel

Pendant les phases 1 et 3, l'ordre dans lequel les valeurs sont lues ou débitées vers le procédé n'est pas important, parce qu'on peut considérer que toutes les valeurs arrivent simultanément. De plus, les entrées n'ont pas d'interdépendance parce que leurs valeurs ne changent pas quel que soit leur ordonnancement. La même chose se passe avec les sorties.

Ce n'est pas le cas avec la phase 2, où l'ordre dans lequel les Blocs de Calcul sont traités est crucial pour atteindre les résultats désirables. Des interdépendances naturelles existent parmi les Blocs parce qu'ils emploient les résultats des autres comme entrées pour leurs propres calculs. Sur des systèmes simples, l'ordre des Blocs de Calcul peut être réalisé automatiquement avec succès parce que si le Bloc B1 emploie comme entrées la valeur du Bloc B2, alors l'ordre de calcul doit être B2 suivi par B1. Sur des systèmes plus complexes, par exemple lorsque l'on réalise l'identification en ligne ou que des boucles fermées sont présentes, l'ordonnancement n'est pas aussi évident. Bien

que l'on soit sûr que les Blocs de Calcul qui emploient la sortie d'autres Blocs avec un certain retard peuvent être calculés les premiers (ils emploient des valeurs déjà calculées aux instants d'échantillonnage précédents), la présence de boucles fermées n'implique pas toujours l'existence de conflits d'ordonnancement.

Quand les calculs incluent la résolution simultanée d'équations dans plusieurs Blocs de Calcul, leur ordonnancement n'est pas important et les blocs impliqués ne sont pas présents dans la liste de calcul (appelée aussi *Sampling List*). Il existe un Bloc de Calcul, appelé Runge-Kutta, qui réalise les calculs d'intégration d'un système d'équations différentielles (chaque équation est un Bloc de Calcul) et qui apparaît dans la liste de calcul.

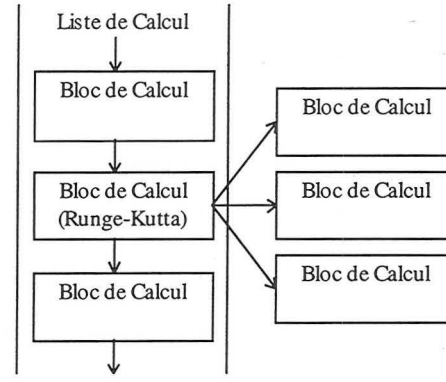


Figure 68 - Bloc de Calcul Runge-Kutta

Pour maîtriser l'important sujet de l'ordonnancement des Blocs de Calcul, certains outils sont disponibles dans ACRUN. Le premier est l'option d'ordonnancement Automatique ou Manuel. Dans les deux cas, une liste des Blocs de Calcul est disponible sur laquelle les blocs qui sont calculés les premiers sont représentés en haut. La liste peut être éditée pour changer l'ordre des Blocs pour que l'utilisateur puisse avoir toute la responsabilité et le contrôle total des algorithmes réalisés. Si l'ordre est changé manuellement, alors l'option Manuel doit être prise. Sinon, le mode Automatique doit être choisi et pour chaque début de simulation ou d'acquisition, un nouvel ordonnancement de la liste de calcul est défini automatiquement.

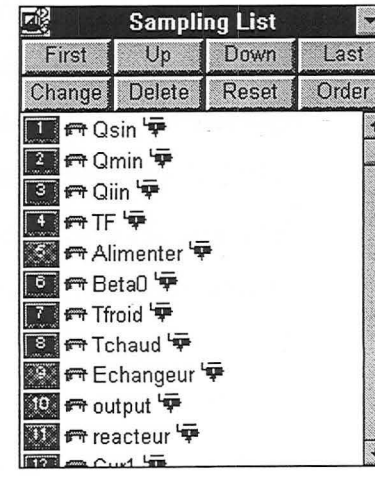


Figure 69 - Menu Liste de Calcul

En mode manuel, un soin spécial doit être apporté quand on ajoute un nouveau Bloc de Calcul parce qu'il va être joint à la fin de la liste de calcul, indépendamment des connexions qui sont établies avec les autres blocs. Dans le mode Automatique, cela n'est

pas nécessaire. Dans le menu d'Echantillonnage, normalement l'ordonnancement en mode Automatique est effectué, puis la vérification en faisant les changements nécessaires s'ils sont requis, et le choix ensuite du mode Manuel.

En mode Manuel la liste d'ordonnance des Blocs est sauvegardée n'importe quand la configuration du programme est enregistrée sur le disque, pour que l'utilisateur ne soit pas obligé de répéter l'ordonnancement manuel au début de chaque session d'ACRUN. Il y a aussi une option 'vérifier' qui montre à l'utilisateur quels étaient les critères que le mode Automatique a employés. Cela peut être utile pour voir les interactions entre les Blocs de Calcul et pour détecter des erreurs éventuelles causées par des interdépendances (un outil complémentaire à la visualisation des flux des signaux). Une option existe, 'Vérifier', pour mieux comprendre les interrelations dans les algorithmes réalisés (spécialement utile dans l'enseignement).

V.4/ Mise à Jour des Objets

Ce chapitre a discuté sur les questions du noyau de la simulation et du contrôle. En effet, les fonctions essentielles, dont les chronométrages sont critiques, sont les entrées et les sorties de données et les calculs impliqués. Maintenant la manière dont les objets représentés sur l'écran sont mis à jour est décrite. Ces objets peuvent être des Blocs de Calcul où sont représentées ses valeurs ou ses éléments de matrice, des Graphiques, des Tableaux, des Compteurs qui sont assez dynamiques, ou des Alarmes dont la mise à jour est très importante. Un soin spécial a été donné pour réaliser l'ordre d'exécution des tâches répétitives (comme le cycle de procédé) et ce qui constitue des exceptions (comme les alarmes), qu'ils soient calculés en ligne ou hors ligne ([AUDSLEY 94] traite de la simulation et du test de ces systèmes).

Pendant la Simulation

En n'ayant aucune contrainte de temps dans la simulation, la mise à jour des objets est entièrement définie par l'utilisateur. Comme un nouveau dessin des objets prend un certain temps, la vitesse de la simulation dépend des options suivantes (également applicables quand les simulations en batch ont lieu):

Vitesse Maximale - l'écran entier est redessiné au bout d'une simulation ou au bout d'une séance de simulations batch.

Normal - l'écran entier est redessiné après chaque simulation et, à chaque seconde, il y a une mise à jour sur le nombre d'échantillons qui ont été calculés.

Animée - les objets sont animés avec un Taux d'Animation, paramètre qui peut prendre des valeurs entières. Si c'est '1', les objets seront mis à jour à chaque échantillonnage, si c'est '2', ils seront mis à jour sur chaque autre cycle d'échantillonnage, et ainsi de suite. Au bout d'une simulation tous les objets sont mis à jour.

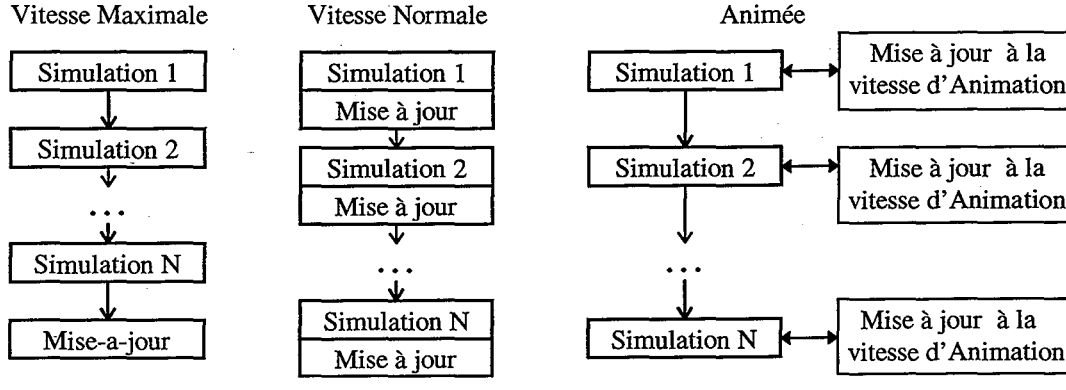


Figure 70 - Mise-à-jour des Objets à l'Ecran Pendant les Simulations

En disant 'tous les objets sont mis à jour', cela signifie réellement que tous les objets qui ont changé seront mis à jour. Particulièrement dans des machines lentes et quand les Graphiques abondent, la mise à jour peut ralentir beaucoup l'opération, donc une attention spéciale était apportée pour détecter quand un objet a réellement besoin d'être redessiné.

Quand une simulation est en cours, elle peut être suspendue et reprise ultérieurement. Cela s'avère très utile parce que n'importe quel objet ou toute une section de l'écran virtuel peut être agrandie ou diminuée sur l'écran. Les objets alors redessinés sont, bien sûr, automatiquement mis à jour avec les données les plus récentes disponibles de la simulation.

Pendant l'Acquisition

Quand l'acquisition et la commande s'exécutent, alors la mise à jour des objets est beaucoup plus compliquée, surtout si la fréquence d'échantillonnage est élevée. Les trois phases fondamentales du cycle de procédé ne peuvent pas être perturbées, donc le temps disponible pour la mise à jour de l'écran est la période d'échantillonnage moins les temps combinés des phases 1, 2 et 3. Comme ces temps peuvent varier, spécialement à cause de la phase 2, après la mise à jour de chaque objet, une vérification doit être faite si un nouveau cycle d'échantillonnage doit commencer ou pas. Si la réponse est positive, la mise à jour est interrompue, les phases 1 à 3 exécutées, et la mise à jour reprise avec le prochain objet.

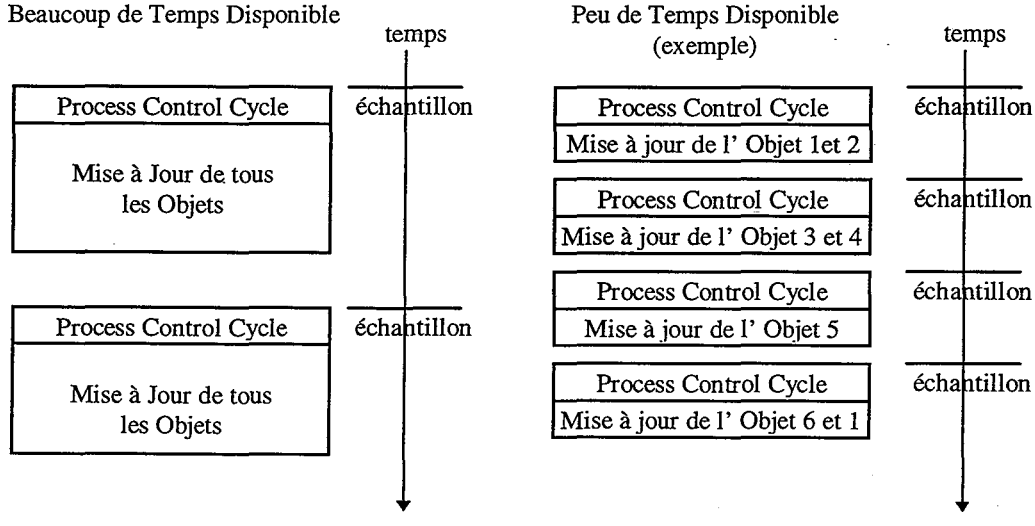


Figure 71 - Mise à jour des Objets à l'Ecran Pendant l'Acquisition

Cette séparation de la mise à jour des objets ne peut pas être planifiée à l'avance, parce que, non seulement les phases 1 à 3 ont des temps qui peuvent varier, mais encore le temps nécessaire à chaque objet pour sa mise à jour peut aussi varier beaucoup. Un Graphique, par exemple, peut avoir une mise à jour rapide s'il requiert seulement le dessin des derniers segments des données qu'il représente, ou il peut être beaucoup plus lent dans la mise à jour s'il atteint la limite de l'échelle X et doit faire du défilement pour loger les nouvelles valeurs.

Si la période d'échantillonnage est si courte qu'ACRUN ne puisse pas mettre à jour un seul objet entre chaque temps d'échantillonnage, alors des solutions doivent être considérées:

- augmenter la période d'échantillonnage - ce que n'est pas toujours possible pour raisons de qualité, de commande du procédé ou pour des raisons de sécurité.
- ajouter un nouvel ordinateur - pour que des tâches puissent être distribuées parmi eux. Dans l'un, on peut réaliser les phases de 1 à 3, tandis que l'autre sera chargé de la surveillance du procédé mettant à jour tous les objets sur l'écran.
- réaliser des méthodes spéciales pour réduire les taux d'échantillonnage comme décrit dans [LURIE 95].

Comme indiqué auparavant, ce genre de problème surgit rarement avec des procédés chimiques qui ont des périodes d'échantillonnage typiquement longues. Pour plus d'information sur l'introduction de tâches apériodiques (comme les alarmes et la mise à jour des objets) dans des tâches périodiques (comme le cycle de contrôle du procédé), il est possible de se référer à [HOMAYOUN 94].

V.5/ L'Enregistrement des Alarmes et leurs Constatations

Quand on a introduit l'objet Alarme, ses caractéristiques principales et ses paramètres ont été décrits pour qu'une idée de ses potentialités puisse être faite. Ici, un examen plus profond sera fait sur la manière dont les objets Alarmes peuvent être réalisés et opérés pour que toutes leurs capacités puissent être employées.

Quand une alarme est déclenchée, certaines actions peuvent automatiquement être exécutées:

- modifier une sortie vers le procédé comme: ouvrir une soupape de sécurité, éteindre un chauffage, allumer une lampe d'alarme ou sonner une sirène d'alarme,
- changer la stratégie de commande pour une autre, qui peut être moins efficace mais plus robuste (cela peut être fait en changeant les paramètres du contrôleur),

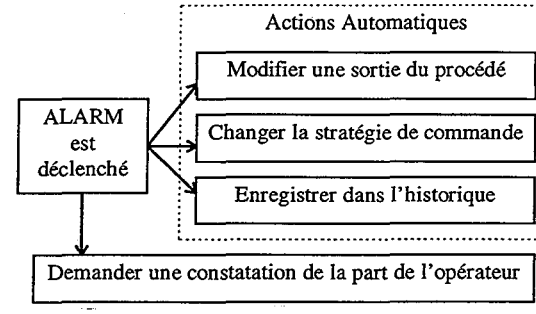


Figure 72 - Actions Possibles des Alarmes

- enregistrer le déclenchement dans une liste de la chronologie des Alarmes qui peut ultérieurement être consultée,
- montrer un message expliquant la condition d'alarme, sonner pour attirer l'attention de l'opérateur et demander une constatation de sa part.

Seul ce dernier point requiert l'intervention de l'opérateur. Seulement après sa constatation de l'alarme, une autre sorte d'action peut avoir lieu qui soit habituellement distincte des précédentes:

- une description des actions à entreprendre est communiquée à l'utilisateur. Cela peut être toute combinaison de messages de texte, des ordres sonores enregistrés, des images et des films.
- la visualisation des synoptiques ou des graphiques de la section du procédé à laquelle la condition d'alarme est attribuée, aidant de cette façon à trouver la cause du déclenchement.
- l'indication pour appeler une autre personne, tel le surveillant ou le chef d'usine.
- l'enregistrement de la constatation dans l'historique des Alarmes et les

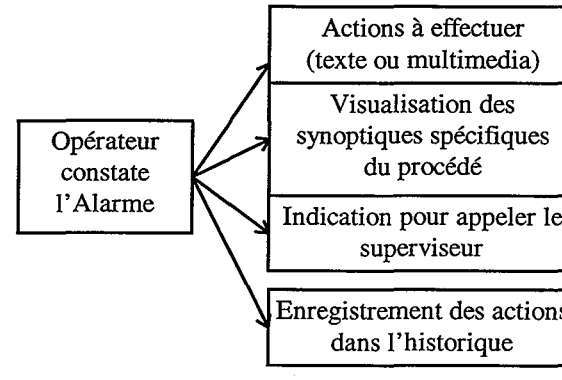


Figure 73 - Séquences de la Constatation des Alarmes

actions exécutées par l'opérateur en réponse à tous ces événements et informations.

Beaucoup de ces points sont directement liés à l'interface homme-machine. Celle-ci doit aider l'opérateur à réagir à temps d'une façon méthodique [LEVESON 90].

Pendant que la simulation a lieu, toutes les alarmes peuvent être inactives pour que le développeur puisse se concentrer sur la stratégie de commande. Comme l'ouverture automatique d'une soupape peut fortement influencer l'état du procédé, cela doit être pris en compte. La même chose peut être dite sur le changement automatique de quelques paramètres de commande ou le changement de la stratégie entière de contrôle. Cela peut nous amener à deux concepts différents de simulation qui sont souvent complémentaires: 1) la simulation d'un algorithme particulier de commande et 2) la simulation de la mise en oeuvre opérationnelle du système entier. Dans les deux cas, si les alarmes sont présentes, elles doivent être convenablement activées ou inactivées pour qu'elles s'ajustent au type de simulation qui est faite.

Le registre d'événements dans l'histoire des Alarmes peut être utile dans la simulation ou ignoré complètement. Quand on l'utilise, alors un des critères de la performance d'une stratégie de commande à employer dans une usine peut tenir compte du nombre de conditions d'alarmes qui surviennent, quels étaient leur importance et le degré d'intervention demandé à l'opérateur. Ces données peuvent être importantes quand l'automatisation complète est l'un des objectifs principaux de la mise en oeuvre qui est développée.

Quand l'acquisition et la commande sont exécutées, alors les enregistrements des Alarmes et leurs constatations jouent un rôle traditionnel fondamental. La chronologie des alarmes permet l'évaluation en ligne et hors ligne des problèmes opérationnels et de leurs diagnostics [DAVIS 94], et permet aussi l'évaluation de l'efficacité simultanément du système et des réactions humaines aux conditions d'alarmes. Cela peut inciter des changements d'alarmes, des observations à l'opérateur ou davantage de formation, et même des changements dans les paramètres du système ou le remplacement complet de la stratégie de commande. De cette façon, les Alarmes peuvent jouer un rôle

fondamental en évaluant tous les facteurs liés au procédé et cela peut être employé comme un outil supplémentaire pour mesurer la performance industrielle de systèmes (par exemple, connaître combien de temps la valeur d'une variable a été supérieure à un certain seuil).

Le système des constatations doit être bien conçu pour faciliter le travail de l'opérateur et tenir compte des conditions de déclenchement multiple des Alarmes. Pour aider à gérer cette situation, la priorité associée à chaque alarme est employée et, naturellement, les conditions Bas et Haut sont ignorées quand les niveaux respectifs Bas-Bas et Haut-Haut ont été atteints.

La mise à jour des objets est considérée comme une tâche de basse priorité, mais informer l'utilisateur des conditions d'alarme ne l'est pas. ACRUN prend tout cela en ligne de compte, donnant la haute priorité à la constatation des alarmes et à toutes les actions associées à celles-ci.
Une visualisation et une surveillance rapide des états des alarmes est aussi réalisée ainsi que des menus graphiques pour diminuer les erreurs de mise en oeuvre sur un sujet aussi délicat. Les simulations intensives d'opérations réelles incluant les alarmes sont recommandées avant d'appliquer le système au contrôle de procédés comportant un risque.

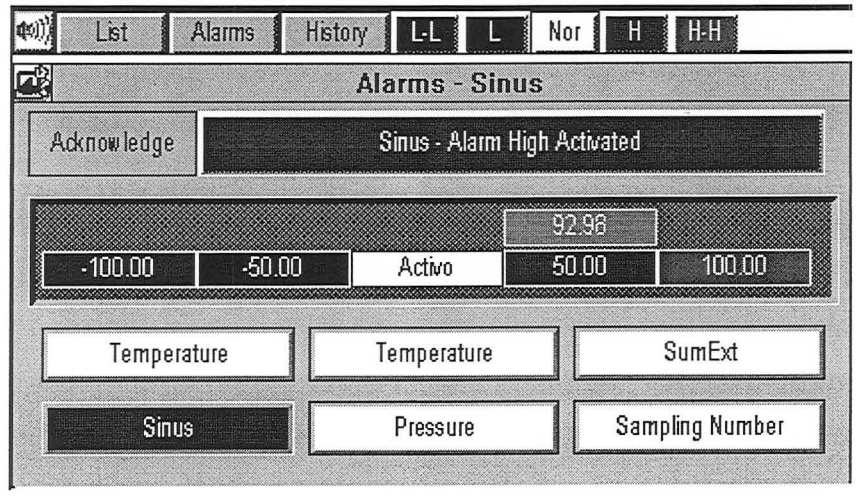


Figure 74 - Panneau de Surveillance des Alarmes

Enfin, n'oublions pas l'importance de l'interface homme / machine / environnement sur la performance des opérateurs, en donnant des réponses correctes et opportunes aux alarmes et à d'autres conditions d'exception [VICENTE 92]. Dans ce but, les alarmes sont intégrées avec l'interface graphique d'ACRUN et respectent le modèle mental employé dans les autres fonctionnalités du logiciel.

V.6/ Mise en oeuvre Distribuée en Réseaux

La synchronisation à la période d'échantillonnage peut être problématique en utilisant seulement un ordinateur. Davantage de problèmes surviennent quand on utilise deux machines ou plus, spécialement si les phases de contrôle, acquisition et surveillance sont distribuées. Dans ACRUN, deux liaisons physiques entre les ordinateurs sont possibles: le port série et les réseaux. N'importe quel nombre d'ordinateurs peut être interconnecté en utilisant une philosophie Maître/Esclave. L'esclave reçoit des ordres du Maître pour savoir quand il doit commencer/arrêter l'échantillonnage. L'esclave change aussi l'information de synchronisation afin de compenser des différences sur les horloges du système (pour la synchronisation d'horloges de système, voir [IL 94]). Un ordinateur Maître peut avoir tout nombre d'Esclaves et être lui-même Esclave d'un autre ordinateur.

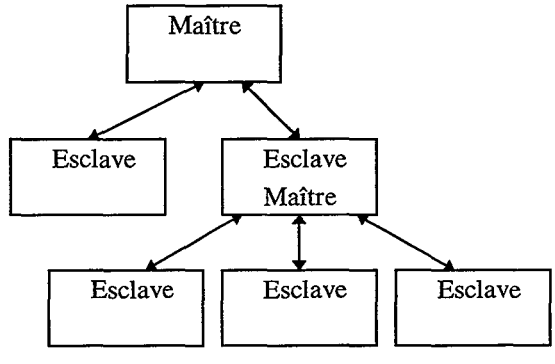


Figure 75 - Hiérarchie Maître - Esclave

Cela donne la possibilité de créer une hiérarchie où l'échec d'un ordinateur n'implique pas la désynchronisation de tout le système.

Les ordinateurs Maître et Esclave d'un réseau peuvent échanger de l'information sur les deux directions utilisant DDE. En utilisant un réseau entre les ordinateurs, des outils supplémentaires sont fournis pour améliorer la supervision des procédés. Ceux-ci permettent à l'utilisateur:

- de prendre une 'photo' de l'écran de tout autre ordinateur exécutant ACRUN,
- de savoir qui est l'utilisateur travaillant sur les autres ordinateurs exécutant ACRUN,
- d'avoir accès à la chronologie des Alarmes dans les autres ordinateurs exécutant ACRUN,
- de visualiser et d'éditer tout objet dans les autres ordinateurs en train d'exécuter ACRUN.

Tout cela peut être contrôlé par les prérogatives d'utilisateur dans chaque programme qui exécute ACRUN. Avec cette approche, beaucoup d'options s'ouvrent pour la mise en oeuvre et pour le domaine de la surveillance de procédé. Finalement, pour la communauté de développement de commande, l'accessibilité à plusieurs ordinateurs en réseaux pourra être utilisée pour répartir entre eux les calculs intensifs et pour avoir accès aux données acquises en ligne par les ordinateurs directement liés aux procédés.

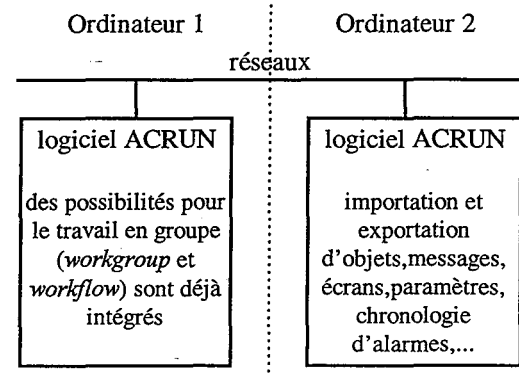


Figure 76 - Possibilités de Travail en Groupe

CHAPITRE VI - Utilisation de l'Environnement Intégré d'ACRUN

VI.0/ Introduction

Après description de la plupart des caractéristiques d'ACRUN, c'est le moment de montrer comment elles peuvent être employées pour répondre aux besoins du développement de la commande et de la mise en oeuvre des procédés.

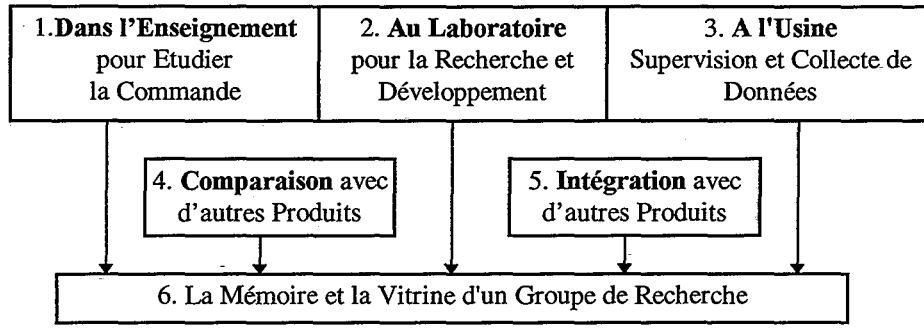


Figure 77 - Structure du Chapitre VI

Trois exemples sont donnés qui couvrent les cibles potentielles de ce genre d'outils: comment un étudiant peut employer ACRUN quand il commence à apprendre la commande de procédé (section 1); comment un chercheur développant la commande adaptive non linéaire peut bénéficier d'ACRUN (section 2); comment certaines contraintes d'usine dans le domaine de la surveillance peuvent être réalisées avec ACRUN (section 3). Impliquer les utilisateurs dans le développement d'un programme est très important [BERRY 94] parce que cela contribue à un produit plus complet, plus robuste et plus facile à l'emploi. C'était le cas en développant ACRUN, bien qu'aucun test utilisateur pleine échelle n'ait été exécuté.

Une comparaison d'ACRUN avec d'autres outils logiciels sera faite dans la section 4, notamment avec les produits décrits dans le chapitre I. Gardant à l'esprit les différences naturelles dans les ressources humaines et financières entre les grandes entreprises de logiciel et celles disponibles dans ce projet, il est cependant possible de détecter un

certain nombre de vertus spécifiques d'ACRUN, notamment dans l'intégration, ce qui peut justifier davantage le développement de son potentiel technologique.

En tant que projet de recherche, l'objectif est aussi d'étudier comment l'industrie de commande de procédé peut se développer à l'avenir. Certaines opinions d'auteurs [DEDENE 94], [MCCROSKEY 91], sont prises en compte, qui décrivent le rôle qu'un logiciel peut avoir comme environnement pour l'intégration d'outils. L'intégration avec d'autres applications, inévitables dans un domaine aussi vaste que l'acquisition et la commande de procédé, est discutée dans la section 5.

Finalement, même si aucune application commerciale ne résulte de ce projet, dans la section 6, les bénéfices qu'il peut amener au groupe de commande où il est développé seront indiqués. Les problèmes qui surviennent en raison de la grande mobilité des chercheurs, seront discutés et comment le travail développé par les étudiants pourra être aisément réutilisé par le groupe même après leur départ. Plutôt que de montrer ses propres technologies, ACRUN devrait être capable de permettre un développement efficace de la commande et de la mise en oeuvre. Plus le fonctionnement du logiciel sera caché, mieux ce sera, parce que son objectif d'avoir une interface intuitive, ainsi qu'un modèle mental cohérent, sera plus proche d'être réalisé.

VI.1/ Dans l'Enseignement: l'Étude de la Commande

Lorqsu'ils commencent à étudier la commande, les étudiants doivent comprendre un certain nombre de concepts. Dans cette phase, le professeur et les livres de classe jouent des rôles fondamentaux. Mais pour une compréhension approfondie des facteurs impliqués, "le couple livre/logiciel est aujourd'hui le vecteur privilégié de pénétration pour les méthodes de commande moderne" [Landau 90].

Il est possible de programmer à partir de zéro, mais avec peu d'avantages pour l'étudiant qui gaspille beaucoup de temps en essayant de résoudre des problèmes purement informatiques et non de commande. Les logiciels spécialisés de contrôle (comme

ACRUN) peuvent à cette étape être un outil crucial pour illustrer les concepts appréhendés en classe et, si l'interface utilisateur est appropriée, inciter l'étudiant à faire de l'exploration non surveillée [ZHU 94]. Cela permet le calcul direct sans la nécessité à compiler des programmes; [RIPPIN 91] fournit un exemple où l'accent mis sur le FORTRAN a été remplacé par l'emploi du logiciel Matlab.

Une tâche typique proposée aux étudiants en commande, c'est l'emploi de l'identification paramétrique pour un procédé donné. Un fichier leur est fourni avec des données du procédé (souvent des données simulées et avec du bruit) et on peut leur demander de:

- trouver les paramètres d'une fonction de transfert discrète en utilisant par exemple l'algorithme Récursif des Moindres Carrés (RMC).
- réaliser ensuite la simulation du procédé avec les paramètres identifiés.

S'il utilise les RMC, l'étudiant applique les équations suivantes [Landau 90]:

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) + F(t+1)\Phi(t)\mathrm{E}^{\circ}(\mathrm{t}+1)$$

$$F(t+1) = F(t) - \frac{F(t)\Phi(t)\Phi(t)^T F(\mathrm{t})}{1+\Phi(t)^T F(t)\Phi(t)}$$

$$\mathrm{E}^{\circ}(t+1) = y(t+1) - \hat{\Theta}(t)\Phi(t)$$

F = matrice de gain d'adaptation

$\Theta^T = [a_1,\ldots,a_{na},b_1,\ldots b_{nb}]$ = vecteur de paramètres

$\Phi(t)^T = [-y(t-1),\ldots,-y(t-na),u(t-1),\ldots,u(t-nb)]$= vecteur d'observation

$\mathrm{E}^{\circ}(t+1)$ = erreur de prédiction a priori

Le modèle de temps Discret employé: $A(q^{-1})y(t) = B(q^{-1})u(t)$

Les concepts théoriques impliqués ne sont pas importants pour le sujet de cette thèse. La raison pour laquelle ce problème est évoqué est que:

- il couvre des sujets importants dans le développement de la commande (identification et simulation).

- tout en ayant une théorie de base simple, il contient cependant beaucoup d'opérations matricielles.
- il illustre certaines fonctionnalités des Blocs de Calcul d'ACRUN, comme: lire les données d'un fichier; générer un signal SBPA (Séquence Binaire Pseudo Aléatoire); ou utiliser le modèle temps discret.
- il emploie l'Interprète de Ligne de Commande (ILC) disponible dans ACRUN.
- il montre le potentiel de la représentation des objets à l'écran, notamment les Blocs de Calcul qui réalisent des matrices sont de grand intérêt pédagogique.
- il illustre l'emploi des modes d'ordonnancement Automatique et Manuel des Blocs de Calcul.

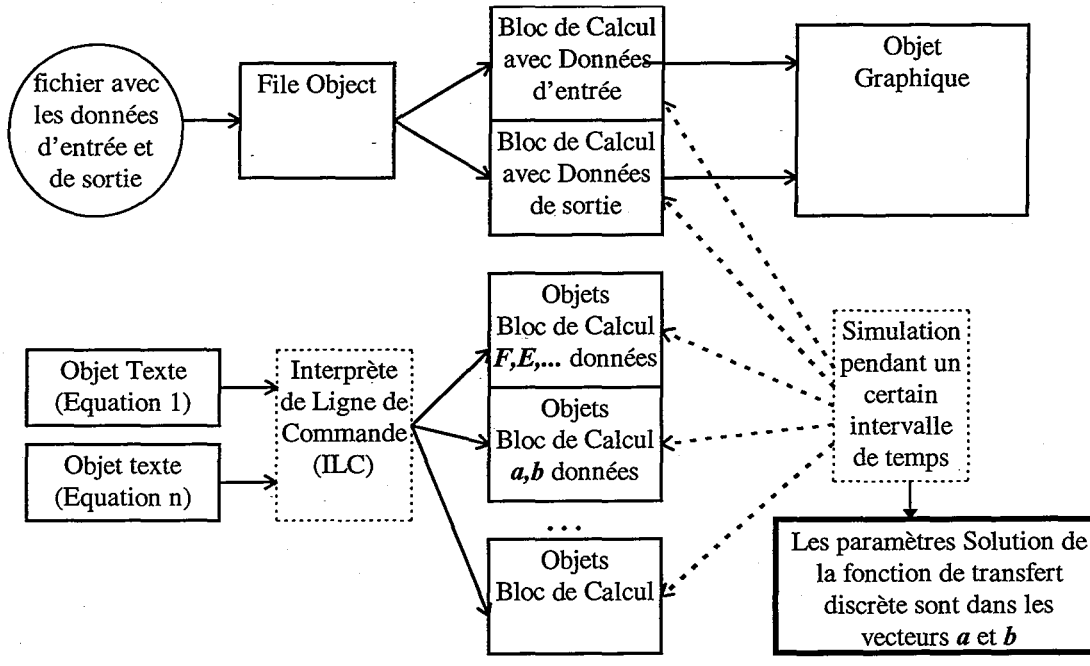


Figure 78 - Recherche des paramètres de la Fonction de Transfert Discrète en Utilisant les RMC

L'étudiant peut commencer par créer un objet Fichier pour représenter les données fournies par le professeur qui proviennent du procédé. Il peut alors le relier à deux Blocs de Calcul qui importent ses valeurs dans ACRUN. Il peut les représenter sur un objet Graphique et faire une première visualisation de ces signaux.

Il peut alors créer des objets de texte pour écrire les équations requises pour l'identification des paramètres du système de temps discret. Il les transfère à l'Interprète de Ligne de Commande (ILC) qui crée les blocs correspondants. La dimension de quelques matrices ou de vecteurs doit être définie (de 'a' et 'b' par exemple) pour que tous les autres ajustent leur dimension en conformité. Plusieurs dimensions de vecteurs 'a' et 'b' peuvent être essayées pour que le meilleur modèle puisse être trouvé.

L'erreur de prédiction peut être représentée dans un Graphique et des statistiques peuvent être employées pour valider les solutions et trouver la meilleure (celle avec la plus petite moyenne d'erreurs ou avec la moindre variance). L'ordre dans lequel les Blocs de Calcul sont exécutés dans la simulation peut être visualisé pour permettre ainsi une meilleure compréhension des relations entre les signaux.

Finalement les paramètres trouvés sont fournis au Bloc de calcul 'Discrete Time' pour réaliser la simulation du procédé (en utilisant les entrées pour le procédé contenues dans le fichier fourni par le professeur et en comparant les résultats avec les données de sortie). La simulation sert à confirmer si les paramètres trouvés sont convenables et pourra ensuite servir au développement de stratégies de commande pour le système.

Il faut noter que la création directe des Blocs de Calcul - sans utiliser le ILC - a aussi été réalisée avec succès. Cela a pris cependant considérablement plus de temps et a donné lieu à beaucoup d'erreurs, comme les retards de temps dans les opérations des signaux, qui étaient difficilement détectés. En utilisant le ILC, tous ces problèmes étaient évités.

VI.2/ Au Laboratoire - Recherche et Développement

Les logiciels de commande de procédé comme Matlab, Matrix, ou de simulation dynamique comme SpeedUp, sont maintenant bien développés pour aider beaucoup le travail de recherche dans ce domaine. Même si ceci est vrai, de nombreux chercheurs continuent à utiliser un compilateur d'usage général pour réaliser leurs simulations. Les avantages d'utiliser les outils informatiques en génie des procédé sont notables, car ils évitent de programmer des fonctions ou des sous-programmes graphiques qui sont déjà incorporés dans les logiciels commerciaux de commande.

ACRUN a été employé avec succès pour la recherche et le développement de la commande. Wang a utilisé ACRUN pour ses développements de recherche concernant la commande adaptative non linéaire multivariable d'un réacteur continu de polymérisation [Wang 95]. La commande géométrique non linéaire utilisée était basée sur un modèle de connaissance du procédé incluant les équations cinétiques et les bilans d'énergie. Comme il s'agissait d'un modèle dans l'espace d'état, un observateur a été nécessaire, dans le cas présent un filtre de Kalman étendu. Enfin, l'incertitude sur le modèle a été prise en compte en incluant un estimateur de deux paramètres: le coefficient d'effet de gel et le coefficient de transfert de chaleur. Le but de la commande était de faire suivre à la température et la conversion de monomère de ce système multivariable les consignes désirées. Le rejet de perturbations a également été étudié.

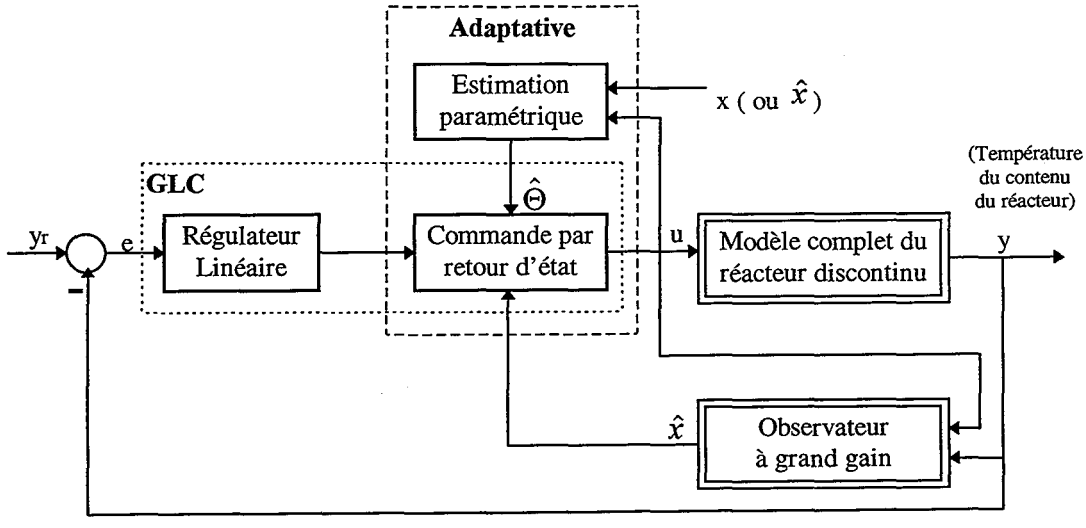


Figure 79 - Structure des Diférentes Combinaisons de la Commande [WANG 95]

La Figure 79 montre la structure du schéma de blocs de la commande non linéaire géométrique globalement linéarisante entrée-sortie (GLC) avec adaptation paramétrique (CA); utilisée par [Wang 95] pour suivre la température du réacteur et la conversion en polymère. En fait, ce schéma de commande sera traduit avec plus ou moins de détails, dans les objets qui composeront l'interface de simulation (Figure 80).

La description d'une partie de ce travail sera faite pour que les avantages du travail utilisant l'environnement intégré d'ACRUN puissent être mis en relief:

- plusieurs des fonctions nécessaires étaient déjà disponibles dans les Blocs de Calcul d'ACRUN.
- des fonctions externes développées en Fortran étaient intégrées dans ACRUN en utilisant des DLLs: le modèle du procédé, l'observateur ou filtre de Kalman, l'estimateur paramétrique,
- des graphiques pour représenter diverses combinaisons de signaux étaient utilisés,
- des objets Fichiers pour sauvegarder sur disque les données des simulations et pour exécuter les programmes éditeurs de Fortran étaient utilisés,
- l'utilisation extensive de différentes couleurs pour les objets et des objets Mère pour faciliter la compréhension, l'emploi et le débogage du système,
- les menus globaux de Blocs de Calcul ont été utilisés grâce aux objets Mère.
- la visualisation des flux de signaux en ACRUN pour vérifier les liaisons,
- le traitement batch et les graphiques 3D respectifs ont été utilisés pour exécuter des simulations et analyser les résultats,
- la gestion extensive des données a été faite en utilisant les capacités de sauvegarde des objets Graphiques.

Un avantage important dans l'utilisation d'ACRUN résidait dans la visualisation des graphiques en cours de simulation, pouvant donner une indication d'un fonctionnement anormal. Le chercheur peut alors arrêter la simulation et procéder aussitôt aux corrections nécessaires. Avant d'utiliser ACRUN, Wang employait un logiciel graphique qui montrait les signaux résultant de la simulation lorsqu'elle était terminée (quand les simulations prennent beaucoup de temps, cela est un grand inconvénient) .

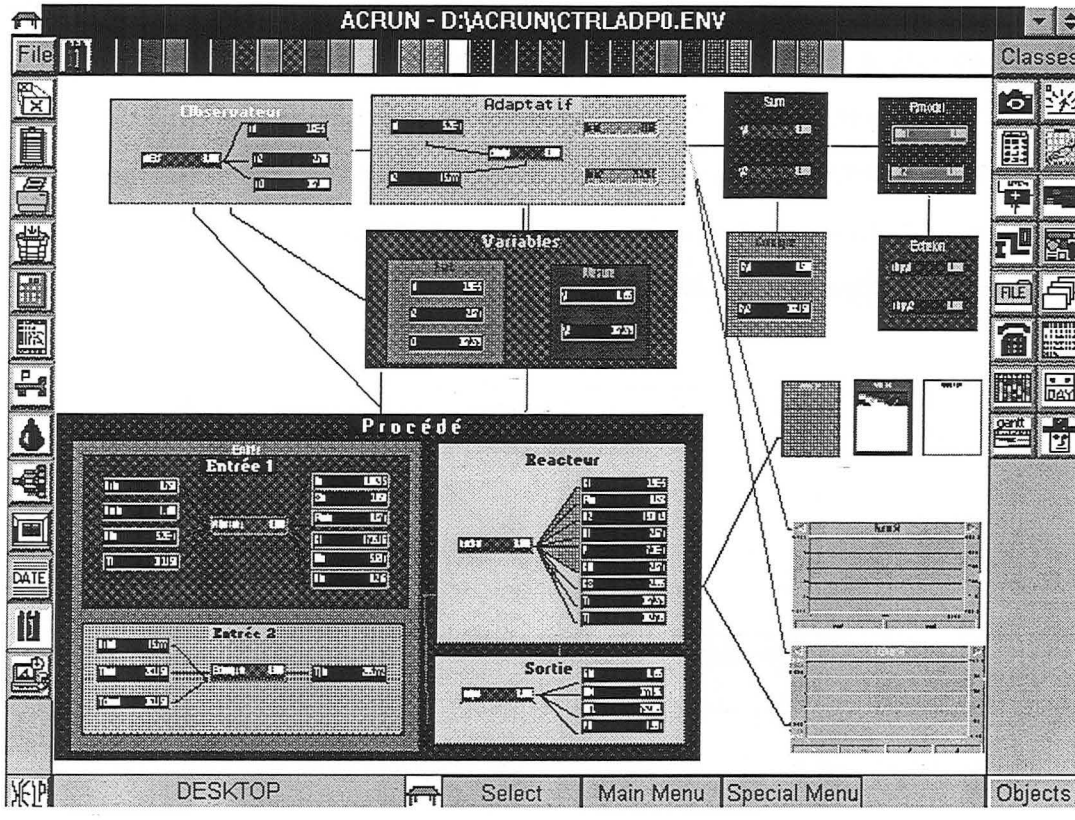


Figure 80 - Ecran Principal Créé pour le Développement de la Commande

Chaque objet Mère créé correspond à une fonctionnalité indispensable à la compréhension du système de commande - les consignes, les entrées, les sorties, le modèle du procédé, l'observateur, l'estimateur, etc., - et était composé de Blocs de Calcul. En temps réel, le modèle du procédé serait remplacé par le procédé lui-même. Ces Blocs de Calcul étaient des fonctions encastrées d'ACRUN ou étaient des fonctions externes programmées par Wang en Fortran. Certains de ces Blocs de Calcul étaient reliés aux objets Graphiques pour que l'évolution de ces variables puisse être suivie pendant la simulation (cela aurait été semblable en temps réel). Les Blocs de Calcul étaient aussi liés à des objets Fichiers pour que leurs valeurs puissent être sauvegardées sur le disque.

En relation avec le schéma de la Figure 79, la partie correspondante au réacteur a été traduite par l'ensemble des Blocs de Calcul dans l'objet Mère 'Reacteur'. Il existe un Bloc (sur la gauche de l'objet mère) lié à une fonction externe qui place les valeurs des variables calculées dans d'autres Blocs (sur la droite de l'objet mère). Ces Blocs représentent les variables internes du procédé et peuvent être connectés à tout autre

objet d'ACRUN. La Figure 81 représente l'objet Mère correspondant au réacteur de polymérisation de styrène. Les variables importantes à suivre apparaissent sur la droite de la figure (concentrations, températures, moments de la distribution) et peuvent ainsi être facilement visualisées en les reliant simplement à un objet graphique. Le "procédé" correspond au modèle du procédé et comprend les équations de bilan de matière, d'énergie et cinétiques.

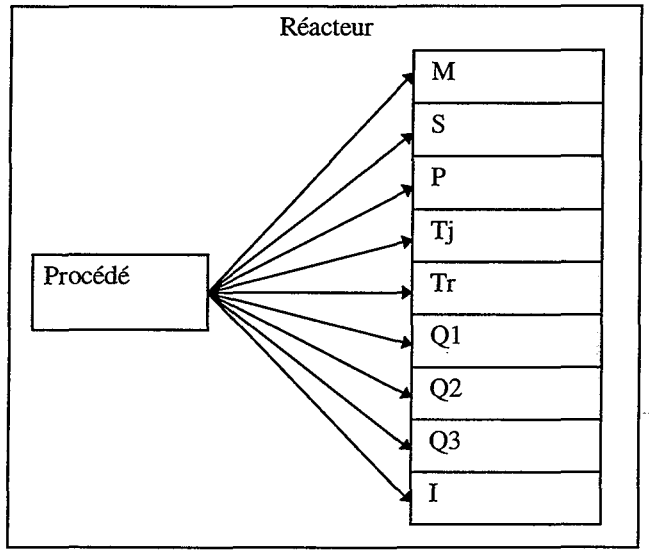


Figure 81 - Réacteur de Polymérisation du Styrène modélisé sous forme objet

Suspendre la simulation, agrandir un Graphique, et reprendre la simulation était aussi une pratique commune de Wang avec ACRUN. Un nouveau graphique pouvait être créé (si la visualisation de certaines variables s'avérait nécessaire) sans qu'il soit nécessaire de recommencer la simulation.

Un autre avantage, en utilisant les couleurs et le groupement d'objets sur l'écran ainsi que la visualisation des connexions entre les objets, est de pouvoir expliquer à des interlocuteurs le principe du système complexe de commande qui était réalisé. Faciles à identifier, les fonctions externes peuvent être vues comme représentant les innovations et les développements principaux qui résultent du travail du développeur. Elles peuvent aisément êtres mises à jour ou remplacées parce que tous les liaisons avec le reste du système (Blocs de Calcul, Graphiques,...) restent de la responsabilité

d'ACRUN. La création des en-têtes des programmes Externe a été réalisée en utilisant le Menu de Bloc Externe décrit dans le chapitre IV.

Ne consommant pas de temps pour réaliser des fonctions comme le bruit aléatoire, les filtres, les sous-programmes d'intégration des équations différentielles, non seulement le travail du chercheur est simplifié, mais il est aussi encouragé pour expérimenter davantage de solutions. Avec un compilateur de langage général, insérer un filtre entre deux signaux peut signifier éditer un ou plusieurs fichiers avec le code source, insérer des variables communes et encore compiler et refaire l'édition de liens de ces fichiers. Dans ACRUN, il est simplement requis d'aller chercher un Bloc de Calcul 'Filtre' et de connecter les Blocs impliqués. Si le filtre n'améliore pas la performance du système, alors il suffit de l'enlever, opération facile dépourvue de possibilités d'erreurs dans les quelques opérations nécessaires.

Des simulations multiples ont aussi été exécutées par Wang, en utilisant le traitement batch. Les résultats et les objets étaient sauvegardés sur le disque et des graphiques tridimensionnel étaient employés pour comprendre et ajuster certains paramètres de commande. Les résultats des simulations batch étaient aussi employés pour étudier le comportement du procédé, notamment sa stabilité en boucle ouverte (réacteur exothermique) et en boucle fermée (liée au système de commande).

Les données de simulation sont gardées dans un fichier sur le disque et employées pour créer les graphiques sur un autre ordinateur. Ces graphiques font partie de la thèse de Wang et cela montrent encore une fois l'importance de réaliser des possibilités d'entrées et de sorties pour communiquer davantage avec les autres logiciels et ordinateurs.

Le changement des consignes par défaut et les paramètres est aussi facilité. Au lieu d'avoir les paramètres dans un fichier qui doit être édité à chaque fois que certains d'entre eux changent, ils sont gardés dans des Blocs de Calcul groupés dans des objets Mère et ils peuvent être vérifiés aisément et édités.

Les possibilités de flux de signaux d'ACRUN étaient fréquemment employées quand des résultats inattendus se produisaient comme des divisions par zéro et des débordements de calculs virgule flottante. La simplicité d'interconnecter des blocs peut aussi conduire à des résultats inattendus, donc la visualisation des raccordements est nécessaire. La puissance dans la réalisation orientée objet des flux de signal d'ACRUN employé par Wang n'est pas limitée aux Blocs de Calcul, mais est appliquée à tous les autres objets. Si des données qui sont enregistrées dans un objet Fichier ou représentées dans un objet Graphique ne sont pas cohérentes, alors l'analyse des flux de signaux d'ACRUN peut aussi être employée pour vérifier les Blocs de Calculs qui sont responsables de la création de ces données (ceux qui sont reliés aux Fichiers et aux Graphiques).

VI.3/ A l'Usine - Supervision et Collecte de Données

Une usine de fibres, Hoechst Fibras, a formulé une demande pour un logiciel intégré pour la supervision et la collecte de données. L'installation consistait en quatre compteurs de vapeur qui sont placés sur plusieurs sites à travers l'usine.

Les compteurs ont des interfaces RS232 et la possibilité d'être mis en réseau parce que chacun peut avoir un caractère spécial d'identification de 'A' à 'D'. Ils sont connectés à un réseau série RS485 qui est relié au port série RS232 d'un ordinateur PC Compatible, après conversion. Même si ces compteurs peuvent être adressés individuellement dans un réseau, le logiciel spécifique qui est vendu pour les contrôler peut uniquement en contrôler un. Ceci constitue un exemple de niches verticales de logiciels de contrôle et surveillance qui sont encore vides et des possibilités de l'approche orientée objet à s'adapter aisément à des situations nouvelles. En effet, il suffit de créer les objets pour communiquer avec un compteur de vapeur, puis de les dupliquer (en changeant le paramètre de 'A' jusqu'à 'D') pour faire les interfaces avec les autres.

Un domaine d'application pour lequel les demandes de spécifications peuvent être critiques est le contrôle de procédé [JAFFE 94]. Même si seulement la surveillance était requise, Hoechst demandait clairement les spécifications suivantes:

1. acquisition de données pour les quatre compteurs de vapeur Spirax, chacun d'eux avec quatre variables: le flux de vapeur, le flux total de vapeur, la pression de vapeur et la température de vapeur,
2. période d'échantillonnage d'une minute,
3. sauvegarde de toutes les données dans un fichier sur le disque avec la date et le temps de chaque échantillon,
4. protection par mot de passe,
5. introduction manuelle du carburant consommé chaque jour,
6. sauvegarde quotidienne automatique de la quantité de vapeur totale produite,
7. alarmes pour surveiller le flux de vapeur.

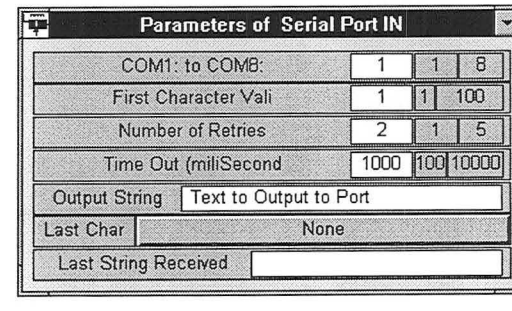
ACRUN était déjà capable de répondre à ces spécifications et seul un Bloc de Calcul a été spécialement développé, le 'Daily Values', pour cette mise en oeuvre. On va voir comment les caractéristiques d'ACRUN ont été employées pour répondre aux besoins de l'usine de Hoechst.

Quatre objets Mère ont été créés représentant chacun un compteur de vapeur. Ils ont été nommés d'après les désignations déjà opérationnelles dans l'usine. Chacun avait une couleur différente pour faciliter la surveillance de l'opérateur.

Quatre Blocs de Calcul réalisant la fonction 'Serial Port In' ont été groupés dans chaque objet Mère,. Ils ont sept paramètres:

COM1 to COM8 - le port série de l'ordinateur à partir duquel les données doivent être lues.

First Character Valid (Premier Caractère Valable)- la position de la chaîne de caractères de l'entrée à partir de



laquelle on doit commencer la conversion à une valeur réelle.

Number of Retries (Nombre d'Essais Supplémentaires) - si la communication échoue en lisant la chaîne de caractères, ce nombre indique combien d'essais supplémentaires doivent être faits pour l'établir.

Time-out (milliseconds) (temps d'attente) - le temps d'attente pour la chaîne de caractères de l'entrée après qu'une requête a été faite.

Output String (Chaîne de Sortie)- la chaîne de caractères à débiter pour faire la requête d'une valeur.

Last Char (dernier Caractère) - Le caractère Spécial à envoyer au bout de la chaîne de caractères de la sortie. Cela peut être un LF (*Line Feed*), un FF (*Form Feed*), un Enter ou bien peut être inactivé.

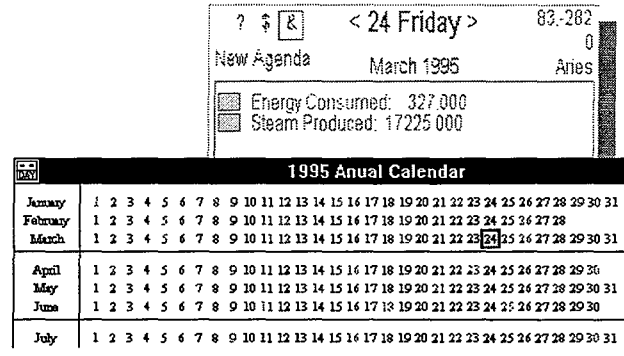
Last String Received (Dernière Chaîne Reçue) - pour aider dans le débogage, ACRUN garde la dernière chaîne de caractères qui est reçue. Elle peut être employée pour vérifier si quelque chose est transmis, si le 'Premier Caractère Valable' est correctement établi, et ainsi de suite.

Pour toute information differente de chaque compteur, une chaîne de sortie différente doit être définie. Dans cette chaîne, un caractère de 'A' à 'D' est aussi inséré pour différencier les compteurs. Un Menu des Ports Série existe dans ACRUN pour définir les paramètres de communication des ports (bits de données, taux de transmission, bits d'arrêt, parité) qui sont valables pour chaque 'Serial Port In' Bloc qui accède à ce port. Dans ce Menu, il y a l'option pour simuler les Ports qui facilite le cycle de développement/mise en oeuvre . En développant et en essayant les objets, le cas peut se présenter où les données ne sont pas disponibles dans les ports séries, par exemple si l'ordinateur employé n'est pas relié au réseau série. La simulation des ports outrepasse ce problème parce que des données aléatoires sont alors produites. Dans la phase de la mise-en-oeuvre, il suffit de désamorcer la simulation des ports pour que des données réelles puissent être lues à partir des ports séries.

Un Bloc de Calcul 'Daily Values' était créé pour conserver les données du carburant consommé chaque jour. Il était relié à un objet Tableau pour que ses valeurs puissent être insérées aisément, visualisées et modifiées. Un autre Bloc 'Daily Values' a été

créé pour garder la vapeur totale produite chaque jour, donc il lit les valeurs des contributions des Blocs de Calcul additionnant la vapeur totale des quatre compteurs. Ce 'Daily Values' Bloc est aussi relié à l'objet Tableau.

En plus, les deux Blocs 'Daily Values' sont reliés à un objet Agenda où ils peuvent être consultés.



Quatre objets d'Alarme ont été créés, chacun ayant comme entrée la valeur du flux de vapeur de chaque compteur. Les valeurs Haut et Bas des Alarmes sont fréquemment modifiées pour mieux contrôler le flux désiré de vapeur. N'ayant aucune possibilité de commande, les seules actions que les Alarmes exécutent sont d'avertir l'opérateur avec un signal sonore et de montrer un message qui doit être ensuite constaté. L'historique des alarmes est aussi conservé.

Des types d'utilisateur: Opérateur, Instrumentiste et Surveillant sont définis pour contrôler les actions dans ACRUN. Arrêter l'acquisition, par exemple, ou sortir du programme est seulement permis aux utilisateurs ayant des prérogatives d'instrumentiste ou de surveillant.

Plusieurs graphiques sont réalisés: un pour chaque compteur avec tous ses signaux représentés, sauf le flux total de vapeur, un pour les quatre signaux de flux de vapeur, un pour les quatre signaux de température, et un autre pour les quatre signaux de pression. Ces trois derniers sont spécialement utiles pour comparer ce qui arrive sur les diverses lignes de vapeur.

Finalement, il y a l'objet Fichier pour sauvegarder les valeurs des Blocs de Calcul sur le disque. Si son taux de fréquence est de '1', donc à chaque minute, il sauvegarde les

valeurs des quatre compteurs sur le disque en écrivant aussi la date et le temps correspondant à l'échantillon. Parfois, chez Hoechst, le fichier est copié vers un autre ordinateur pour être analysé et pour garder un historique de tout le procédé. Quand cela est fait, le fichier original est mis à zéro pour qu'il ne devienne pas trop grand (la dernière fois où une nouvelle version d'ACRUN a été installée, ce fichier avait plus de 9 mega-octets).

La production est rarement interrompue, donc des équipes sont sur place pour assurer une opération de 24 heures par jour. L'ordinateur utilisé pour exécuter ACRUN est basé sur un PC 486 avec 8 Mega-octets de mémoire vivante (RAM) et son système d'exploitation est le DOS/Windows for Workgroups 3.11.

Même si ce n'est pas la mise en oeuvre idéale pour tester les aptitudes de ACRUN, cet exemple nous procure une expérience valable et la possibilité d'employer quelques-unes des caractéristiques du programme. Son opération n'a jamais requis de soutien et la formation du responsable a pris environ 30 minutes, donc l'interface utilisateur semble être adéquate. Cette expérience encourage le développement d'ACRUN en tant que produit commercial opérant en temps réel.

VI.4/ Comparaison à d'Autres Produits

L'évaluation des logiciels, leur comparaison et leur choix sont toujours difficiles même s'il existe quelques études sur des méthodes quantitatives et qualitatives disponibles [LEBLANC 94]. ACRUN peut être considéré comme un outil général parce qu'il offre une large gamme de caractéristiques, quoique chacune d'entre elles ne soit pas toujours explorée en profondeur. L'objectif est de montrer comment la technologie orientée objet peut être employée avantageusement en intégrant tous les outils requis dans la mise en oeuvre et dans la commande de procédé. En gardant cela à l'esprit, on va cependant comparer ACRUN à d'autres produits pour mieux comprendre les points forts et les faiblesses de la réalisation actuelle du projet ACRUN.

Le logiciel de développement de commande MATLAB, décrit dans le chapitre I, présente un ensemble extensif de fonctions réalisées et des options graphiques qui sont beaucoup plus développées que dans ACRUN. Le programme ACRUN a cependant une approche plus flexible. En utilisant Matlab et Simulink et en sélectionnant les blocs de Sources, les mêmes fonctions sont disponibles que dans les Blocs d'ACRUN, mais il n'y a pas la possibilité directe de modifier leurs entrées, mais seulement d'accéder aux valeurs de leurs sorties. Dans ACRUN cela n'est pas le cas, parce que tous les paramètres de Sinus, par exemple, peuvent utiliser des valeurs d'autres Blocs pour que des signaux complexes puissent être facilement générés. Si la sortie d'une onde rectangulaire est l'entrée pour le paramètre de fréquence 'f' du Sinus, le signal de la Figure 82 peut être généré.

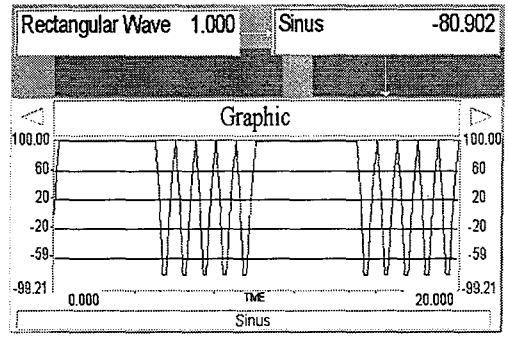


Figure 82 - Ondes Rectangulaire et Sinusoïdal

Dans Simulink, on peut aussi visualiser les signaux pendant que les simulations s'exécutent et celles-ci peuvent être interrompues, certains paramètres modifiés, et ensuite les simulations sont reprises. Cependant le nombre d'entrées ou de sorties ne peut pas être changé pendant une simulation donc, contrairement à ACRUN, ajouter un signal à un Scope (Graphique) existant exige que la simulation soit recommencée.

Les calculs en Matlab sont plus rapides qu'en ACRUN, donc les simulations s'exécutent dans un temps plus court. Cependant Matlab n'a aucune possibilité comparable au traitement Batch d'ACRUN, qui permet qu'une série de simulations successives soient effectuées sans la présence de l'utilisateur.

Enfin, ACRUN présente toutes sortes de fonctionnalités (objets) que Matlab n'a pas, spécialement dans le domaine de la mise en oeuvre de l'Acquisition et du Contrôle qui est simplement en-dehors de sa cible (il y a des logiciels qui permettent à Matlab d'être employé pour l'opération en temps réel, mais il n'est pas possible dans Matlab

de créer ou de connecter des pilotes pour les entrées/sorties de données des cartes d'acquisition et de contrôle).

Dans le domaine des logiciels de mise en oeuvre, la comparaison est faite avec Labtech/Control, décrit dans le chapitre I. Certaines similarités et certaines différences significatives peuvent être indiquées par rapport aux caractéristiques d'ACRUN. Les deux disposent de blocs numériques et analogiques (le bloc échelon dans Labtech/Control est un cas particulier du bloc numérique), des blocs de PID et de contrôle d'alarme, des affichages de données sous forme de graphiques et d'images (synoptiques), de la sauvegarde des données sur le disque continue ou déclenchée par des événements, et d'une interface graphique où placer tous les afficheurs et les blocs de fonctions distribués par un ou plusieurs écrans.

Les différences principales entre ACRUN et Labtech/Control résident dans les possibilités d'intégration. ACRUN peut échanger tout type d'image ou de données sur les formats ordinaires du Presse-Papiers de Windows tandis que Labtech/Control se limite à importer des images BMP. ACRUN emploie le DDE pour échanger des informations avec d'autres applications et intègre des objets OLE tandis que Labtech/Control ne le fait pas. Le travail en réseaux est une partie intégrée d'ACRUN tandis que c'est un ajout de Labtech/Control, limité à seulement deux ordinateurs. Les fonctions externes de l'utilisateur reliées à ACRUN peuvent être développées dans tout compilateur de langage qui peut créer des fichiers conformes à la norme DLL tandis que dans Labtech/Control, cela doit être fait en utilisant le langage C. Enfin, ACRUN inclut un langage spécifique pour développer des pilotes pour les cartes d'acquisition disposant d'outils spécifiques de débogage intégrés dans l'interface graphique, tandis que dans Labtech/Control, une boîte à outils spéciale doit être ajoutée pour développer extérieurement ces pilotes.

Toutes ces différences réflètent le souci principal de cette thèse: montrer comment l'intégration peut être réalisée et les avantages qu'elle procure au logiciel de contrôle et d'acquisition. Non seulement l'intégration d'outils de développement de commande avec opération en ligne était réalisée, nous permettant de comparer ACRUN à deux

produits aussi différents comme Matlab et Labtech/Control, mais dans chacun de ces domaines, davantage d'intégration était réalisée. Autrement dit, non seulement l'intégration de procédé a été réalisée mais encore l'intégration de données, de présentation et de contrôle.

L'implémentation et l'intégration de toutes les caractéristiques décrites est due essentiellement à l'approche orientée objet qui était prise presque dès le début. Matlab et Labtech/Control, ayant été en développement pendant plus de dix ans, ont eu beaucoup plus de fonctions réalisées, mais leurs caractéristiques ne sont pas aussi bien intégrées. Par exemple, les interfaces graphiques disponibles dans les deux produits (Simulink pour Matlab et IconView pour Labtech/Control), ajoutées au cours de ces dernières années, n'ont pas toutes les caractéristiques de l'interface graphique d'ACRUN complètement orientée objet. Un autre exemple est dans l'intégration avec les formats de données de Windows et les possibilités de partager des objets, qu'ACRUN emploie et qui lui permettent de communiquer aisément avec d'autres applications de Windows comme nous le verrons par la suite.

VI.5/ Intégration avec d'Autres Applications

Une caractéristique cruciale de ACRUN est son extensibilité due au fait qu'il est totalement ouvert à l'échange de données avec d'autres applications. Du bloc DLL externe, créé par l'utilisateur, aux données échangées avec d'autres applications Windows en utilisant le DDE, le programme facilite la communication avec l'extérieur. La possibilité d'écrire des pilotes pour toute carte d'acquisition, de lire et d'écrire des données utilisant jusqu'à 8 ports séries de l'ordinateur, et la possibilité d'échanger des données par des réseaux sert à classer ACRUN comme un programme extensible et ouvert qui peut aisément s'intégrer avec d'autres applications.

Ce fait n'est pas un bonus qui a été ajouté, développé pour améliorer la liste de caractéristiques du programme. Par contre, il est la condition principale à la survie du

projet entier. La programmation orientée objet pourrait avoir été employée pour développer un domaine scientifique spécifique comme la classification de connaissance et la systématisation [VIRRANTALO 94]. La décision a été plutôt de l'employer pour intégrer des sujets différents, chacun d'entre eux étant complètement ouvert pour davantage d'améliorations.

Dans le contrôle de procédé, par exemple, il n'est pas possible de réaliser tous les algorithmes de commande ou tous les simulateurs de procédé parce que, dans chaque domaine, beaucoup d'approches différentes peuvent être prises pour résoudre les problèmes particuliers. La solution est que, à la mise en oeuvre horizontale d'ACRUN, puissent être ajoutés et intégrés des outils verticaux pour répondre à ces cas spécifiques.

Pour que cela soit possible, le choix de réaliser des règles propriétaires de connexion avec les autres programmes aurait pu être retenu. Au contraire, les normes qui existent sur le système d'opération choisi étaient employées. En effet, un grand nombre de versions de programmes pour le système graphique Windows sont réalisés, notamment dans les domaines de la mise en oeuvre et du contrôle, la plupart d'eux supportant DDE, DLL, et OLE. Cela semble valider le choix initial de développer ACRUN dans la plate-forme IBM PC / DOS / Windows. Beaucoup de ces programmes de contrôle étaient initialement disponibles sur des stations de travail UNIX, mais la puissance de calcul et les capacités graphiques requises sont maintenant également disponibles dans les PCs.

Tout cela conduit au rôle complémentaire et d'extensibilité qu'ACRUN est préparé à jouer et certains exemples peuvent être les suivants:

- relier ACRUN à des DLLs externes pour utiliser des algorithmes qu'il ne réalise pas,
- transmettre des données d'ACRUN aux autres applications de commande ou à un autre programme ACRUN, dans le même ou sur un ordinateur différent pour des fonctions de surveillance,

- transmettre les données d'ACRUN vers un traitement de texte pour la création de rapports,
- relier ACRUN à un programme de tableur pour la mise en oeuvre d'algorithmes, pour la génération des rapports ou pour la surveillance et validation de données,
- transmettre des données d'ACRUN à un programme graphique pour la création de graphiques ou d'animation,
- employer ACRUN pour gérer les données des entrées et des sorties des cartes d'acquisition et les relier à d'autres programmes spécialisés de commande et de supervision,
- utilisation par ACRUN des objets externes venant de programmes serveurs d'OLE pour réaliser les graphiques, les sons, les animations, les synoptiques, les équations, etc.,
- fourniture par ACRUN d'objets OLE à des programmes clients d'OLE pour y ajouter de l'acquisition et de la commande, des possibilités de lire et d'écrire dans des fichiers, la gestion du temps, etc. .

En ayant beaucoup de fonctionnalités, les objets d'ACRUN et leurs connexions donnent à l'utilisateur beaucoup de choix et d'espace permettant que des fonctionnalités qui n'avaient pas été originalement prévues puissent être envisagées. Il est inutile d'essayer d'énumérer tous les choix pour réaliser une tâche particulière. Les avantages de DDE et, surtout, de OLE sont cependant limités par la puissance de calcul requise pour les faire fonctionner d'une façon rapide et confortable pour l'utilisateur. A mesure que l'équipement des ordinateurs évoluera, il deviendra commun de caractériser un programme en prêtant une attention aussi forte à sa fonctionnalité intrinsèque qu'à ses possibilités d'intégration et de communication.

Dans cette optique, la philosophie réfléchie dans ACRUN est déjà un exemple de ce qu'un programme ouvert et extensible pour la commande et l'acquisition doit comporter.

VI.6/ Mémoire et Vitrine d'un Groupe de Recherche

Aujourd'hui, plus que jamais, un groupe de recherche est un système ouvert. Pendant son existence, il éprouve les entrées et les sorties suivantes:

- des étudiants de recherche avec des séjours qui vont de quelques mois à plusieurs années et qui partent ensuite,
- des projets communs à deux ou à plusieurs groupes appartenant à des universités, à des laboratoires ou même à des pays différents,
- des symposiums, des conférences, des congrès et des ateliers.
- des collaborations avec l'industrie en partageant de l'équipement et du savoir-faire.

En ayant une stratégie de recherche, les groupes non seulement doivent être organisés pour gérer tous ces événements, mais encore être orientés pour bénéficier le plus possible de toutes ces interactions. Il est commun qu'un étudiant qui reste trois années travaillant sur sa thèse de doctorat, parte en laissant une bonne documentation théorique avec l'explication de son travail, mais un héritage expérimental pauvre difficile à réutiliser par d'autres personnes. Dans un groupe de recherche concernant la commande, il arrive qu'une partie importante du travail soit basée sur des simulations d'ordinateur utilisant des algorithmes spécifiques de commande et des modèles de procédé. Ces algorithmes pourraient et devraient ultérieurement être comparés à d'autres solutions même quand les chercheurs qui les ont développés sont déjà partis.

Un programme comme ACRUN peut aider dans la réalisation de cela en imposant à chaque chercheur, qu'avant son départ, il laisse une bibliothèque avec des Fonctions Externes qu'il a développées et l'information respective d'aide à son utilisation. Ces sous-programmes peuvent être développés dans le langage de leur choix et sur l'ordinateur qu'ils préfèrent pourvu que, à la fin de leur travail, ils soient reliés à ACRUN.

Le travail de tout le monde conservera ainsi son unité et son accessibilité par une interface commune. Cela constitue "La Mémoire d'un Groupe de Recherche".

En ce qui concerne les autres acteurs du domaine de l'acquisition et de la commande, les problèmes qui surviennent sont assez différents. En contactant un autre groupe de recherche ou une usine, un des objectifs est de décrire ce qui a été fait et les résultats qui ont été obtenus jusqu'à présent. Dans le développement de la commande, les représentations graphiques sont très importantes et les imprimer peut être une bonne façon de montrer les résultats. Mais, si un exemple dynamique d'évolution d'un système de contrôle est désiré, ou si quelqu'un demande le résultat de l'insertion d'un filtre, le graphique statique pourrait ne pas suffire. Dans des usines où les synoptiques sont très importants, il convient de présenter l'exécution d'un programme, pour mieux motiver les responsables et d'évaluer si une collaboration, par exemple par un contrat industriel, peut avoir lieu.

Cela peut être fait avec un programme comme ACRUN s'il intègre le travail des chercheurs du groupe. Des interfaces de texte avec des nombres défilant vers le haut de l'écran ne peuvent plus être présentées, parce que leur interprétation est difficile et aussi parce que cela risque de causer une mauvaise impression (le *marketing* devient omniprésent). La recherche avancée doit être présentée avec une visualisation graphique appropriée parce que des interfaces homme-machine de haute qualité sont déjà largement utilisées par les groupes de recherche.

Cette interface unifiée du savoir-faire du groupe, qui peut être facilement interprétée et manipulée par d'autres peut être appelée "la Vitrine d'un Groupe de Recherche".

On peut se demander pourquoi ne pas employer un autre logiciel, notamment des produits commerciaux bien connus comme mémoire et vitrine du groupe? Certes, ils sont bien documentés et constituent des produits finis (normalement une garantie d'être source de peu d'erreurs) qui ont des fonctionnalités bien développées et qui peuvent aussi être achetés par d'autres groupes et d'autres institutions, donc ils apparaissent comme le meilleur choix. De plus, tout groupe de contrôle n'a pas de ressources humaines avec le profil nécessaire aux développements du génie logiciel que ces projets exigent.

Considérons les avantages de disposer d'un projet comme ACRUN dans un groupe de recherche:

- savoir-faire dans les domaines de l'équipement informatique et du logiciel liés au contrôle de procédés que les mathématiciens et les automaticiens habituellement n'ont pas.
- être capable d'ajuster le logiciel ACRUN aux besoins du groupe et ne pas faire l'inverse: ajuster le groupe aux caractéristiques du logiciel et espérer qu'une nouvelle version possédera les caractéristiques supplémentaires requises. Souvent cela conduit à l'achat de plusieurs produits pour répondre à des besoins différents et les avantages d'une interface unifiés disparaissent.
- pouvoir exécuter des copies multiples du logiciel sans outrepasser les droits d'auteur du logiciel. Il peut être distribué à plusieurs étudiants, installé dans d'autres groupes pour faciliter la collaboration, et des solutions d'installation sont faciles à négocier pour des usines et des partenaires industriels, sans dépenser une grande portion du budget dans l'achat du logiciel.
- la possibilité de transformer tôt ou tard le logiciel en un produit commercial avec tous les avantages résultant simultanément au point de vue financier et pour le prestige du groupe. Le produit peut être ACRUN lui-même ou des applications verticales ayant ACRUN comme base (comme la solution des compteurs de vapeur décrite plus haut dans ce chapitre).
- le savoir-faire du génie logiciel associé aux techniques orientées objet peut être utile pour développer des algorithmes réalisant le contrôle dans des dommaines différents: par exemple, les réseaux neuronaux [ZOMAYA 94, CUI 93] ou la logique floue. Ce savoir-faire peut également être employé pour développer des systèmes experts utilisant des divisions de classe et les caractéristiques d'héritage qui font partie du concept objet.

Le pour et le contre doivent être pesés dans le choix d'un logiciel, qu'il soit développé intérieurement ou extérieurement, afin d'y concentrer le savoir-faire du groupe. Un programme comme ACRUN doit être considéré comme une option sérieuse.

L'emploi d'ACRUN dans un groupe de recherche amène aussi de grands avantages pour son développement. Premièrement, pour accélérer les tests et le débogage [SINGPURWALLA 91] et deuxièmement, il peut être continûment amélioré par les besoins et les suggestions des chercheurs. S'il y a une collaboration avec l'industrie, alors les contributions à la mise en oeuvre du procédé sont aussi un ajout important à tout le travail expérimental normalement exécuté dans le groupe.

Dans le développement de systèmes interactifs comme ACRUN, il est aussi nécessaire d'être préoccupé par l'interaction qui a lieu entre le système et ses utilisateurs. Ici la perspective du programmeur n'est pas nécessairement la plus appropriée [DUKE 95b]; donc toutes les contributions des utilisateurs doivent être prises en compte.

Conclusions et Perspectives

Un programme, ACRUN, a été développé faisant appel à la technologie orientée objet et répondant à une large variété de besoins de développement de la commande et des systèmes de mise en oeuvre. Il allège le cycle développement/mise en oeuvre, offrant une interface utilisateur commune et utilisant les mêmes objets pour le développement et pour la mise en oeuvre.

N'étant pas un outil spécialisé, il possède une architecture ouverte, qui permet davantage de développements, intègre leurs sous-programmes externes de l'utilisateur, et échange des données avec d'autres programmes sur le même ordinateur ou en réseaux de plusieurs façons.

Il fonctionne sous DOS/Windows ou tout autre système d'exploitation graphique compatible utilisant les normes standard de partage de données et d'intégration d'objets. Il est approprié pour la commande et l'acquisition de systèmes avec des fréquences d'échantillonnage pas trop élevées, comme cela est généralement le cas pour la commande et la supervision des procédés chimiques. Il peut être connecté avec le procédé en utilisant les cartes d'acquisition insérées dans l'ordinateur ou par ses ports séries.

L'originalité de ce travail a été de développer des fonctionnalités très différentes et de les intégrer dans un même environnement. Ce haut niveau d'intégration résulte d'une utilisation du paradigme objet à tous les niveaux de sa réalisation.

En permettant à l'utilisateur de visualiser et de modifier aisément des objets, les buts suivants d'intégration ont été atteints:

- la réalisation d'une grande quantité et variété de fonctionnalités sous une même interface sans accroître la complexité du programme,
- la réalisation des connexions entre des objets de façon intuitive et flexible, livrant un outil puissant dans un environnement graphique avancé,

- assurer la complémentarité des outils de développement et de commande avec d'autres outils d'usage général, ce qui améliore d'une façon unique l'utilisation de ce type de programmes,
- l'utilisation des normes de la technologie d'objet tel que OLE, permettant l'utilisation de fonctionnalité évolutive comme le multimédia dans les systèmes opérationnels de contrôle,
- la réalisation de plusieurs outils pour gérer et visualiser/cacher les objets et leurs connexions rendant ACRUN approprié à manipuler des vues multiples d'un grand nombre d'objets.

Finalement, en se référant aux quatre vecteurs principaux d'intégration décrits dans le chapitre II, ACRUN a atteint un haut niveau d'intégration:

1. **Intégration de Contrôle** - des services uniques sont fournis par chaque type d'objet qui sont rendus disponibles à toutes les autres classes d'objets.
2. **Intégration de la Présentation** - un modèle commun d'interface et d'interaction a été réalisé pour tous les objets, résultant en une visualisation et un comportement à l'écran commun.
3. **Intégration de Procédé** - coopération entre les classes disponibles d'objets pour réaliser toute la fonctionnalité requise pour l'acquisition et la commande.
4. **Intégration de Données** - chaque objet renferme ses propres données éliminant toute duplication ou redondance. Les changements des données d'un objet sont aussitôt communiqués aux autres objets pertinents utilisant l'information des flux de signaux.

L'intégration était l'objectif principal et constitue maintenant la plus grande vertu du programme.

Les perspectives pour ACRUN sont basées sur l'intégration (qui inclut extensibilité et productivité) et réalisées par la programmation orientée objet. Une philosophie sur l'avenir des outils de développement et de contrôle est présente dans ACRUN, montrant la voie que les futures améliorations doivent prendre.

La quantité, la variété et les caractéristiques des objets développés présentent le potentiel pour plusieurs applications commerciales basées sur le travail accompli. En tant qu'outil général de développement de la commande et de la mise en oeuvre, ou en tant que base sur laquelle reposent des solutions verticales pour répondre aux besoins des usines et des laboratoires, beaucoup d'applications peuvent être trouvées pour ACRUN. Même comme simple gestionnaire d'objets (remplaçant les anciennes technologies orientées programme) ACRUN apporte beaucoup dans le potentiel envisagé pour la technologie objet.

Pour la réussite dans les systèmes de mise en oeuvre et de commande, ACRUN requerrait d'être amélioré pour inclure un véritable fonctionnement en temps réel. Son passage à un programme 32 bits est aussi important pour exécuter des calculs plus rapidement et pour une plus grande souplesse de son interface utilisateur. La compatibilité avec d'autres langages pour le développement de la commande, la gestion des exceptions, des aptitudes de débogages pour les sous-programmes reliées dynamiquement, et la documentation extensive dans le système électronique d'aide sont aussi indispensables.

De nouveaux outils (objets) qui sont largement employés dans l'industrie devraient être ajoutés comme, par exemple, réconciliation de données (un domaine complexe et vaste très important [RENGASWAMY 94]) et la programmation d'automates (aussi largement employée [DESPRES 91]).

A l'avenir, le développement ou l'intégration d'un moteur de base de données orienté objet avec les caractéristiques et les avantages décrits dans [KIM 90] sera nécessaire. Cela permettra l'accès client/serveur aux objets employés dans ACRUN réalisant la sauvegarde, le partage (ou ce qui est maintenant appelé les aptitudes pour le travail en groupe), et l'accès à distance avec de grands avantages pour la sécurité et la performance.

En ce qui concerne l'interface utilisateur, la prochaine grande innovation sera l'intégration de la voix et des gestes humains [CADOZ 94] pour contrôler les objets utilisant les équipements multimédia qui sont installés dans les ordinateurs.

Utilisant plus de 70000 lignes de code, ACRUN est néanmoins un projet ordonné et facile à gérer. Cela est dû à l'encapsulation de la fonctionnalité dans des objets différents avec les codes source respectifs résidant dans des fichiers séparés. Programmé en Pascal avec des extensions objets, il est compréhensible, et facile à entretenir et à améliorer même par d'autres que le promoteur initial. Cela ouvre davantage de bonnes perspectives pour la continuation du développement du projet ACRUN, exploitant sa philosophie loin au-delà de la communauté universitaire et de recherche vers le marché industriel.

Finnalement, il y a l'espoir d'avoir contribué à réduire le manque de considération que le monde académique donne [RIPPIN 91] aux domaines qui ont été développés dans ce travail.

ANNEXES

Annexe A - Fonctions Réalisées dans les Blocs de Calcul

Une liste des Blocs de Calcul disponibles dans ACRUN séparée par des familles de fonctions est présentée ici:

CONSTANT AND TIME

Constant - Keyboard as Input
Time - Time from the beginning of sampling

GENERATORS

Sinus - Sinus Wave if T is time
Rectangular Wave - Rectangular Wave if T is TIME
Triangular Wave - Triangular Wave if T is time
Selector of Values - Selects between two or more values
Random Signal - Random Generated Signal
Gaussian Signal - Random Generated Signal with Gaussian Distribution
PRBS - Pseudo-Random Binary Sequence

INTEGRALS

derivative - Derivative
int X,T - Integral

Runge-Kutta - Implements a Runge-Kutta Integration Method
k.X+m - Multiply and Add by constants
X.x+Y.y - Weighted Sum of two Blocks

OPERATORS

X.Y.k - Product of Blocks
divide - Division of two Blocks
XtoR - X power to R
inverse - Inverse of X
ABS X - Module of X
X operation Y - Operation Between two Blocks
Scalar Operations - Scalar Operation Between Blocks

EXPONENTIAL-LOGARITHM

Ln X - Neperian Logarithm of X
Log X - Logarithm of X
EXP - Exponential of X

TRIGONOMETRY

TAN X - Tangent of X
ArcSin X - ArcSinus of X
ArcCos X - ArcCosinus of X
ArcTan X - ArcTangent of X
ArcCoTan X - ArcCoTangent of X

LIMITS

MIN X,Y - Minimum of X and Y
MAX X,Y - Maximum X,Y
LLimit X,r - Se X menor do que r entao igual a 1
ULimit X,r - Se X maior do que r entao igual a 1
Infime X - Smallest Value of Input
Supreme X - Largest Value of Input

STATISTIC

Mean X,n - Mean of the n last values of X
Standard Dev. X,n - Standard Deviation of last n Values
Whiteness - Whiteness of last n values

SPECIAL - FILE

PT 100 ohms - Receives a V and outputs a temperature
Controls Sampling - Registers the Error in Sampling
Voltage to Value - Receives a (4-20 mA * 220 OHM)Voltage
Daily Values - Stores ONE Value each Day
Pre-Recorded - Reads Values From a File

CONTROL

PID X,S - PID with Error and parameters to T0

PORTS AND BOARDS

Serial Port IN - Reads data From Serial Port
Serial Port OUT - Writes data to Serial Port
Paralell Port IN - Reads data From Parallel Port
Paralell Port OUT - Writes data to Parallel Port
Board - Receives a value from an Aquisition Board
Ana. to Dig. - Receives a value from a Board

SYSTEMS

Polynomial in X - Polynomial 0 - 9 Degree
Digital Filter - Digital Filter u=input y=output (ARMA)
Discrete System - B(q-1)/A(q-1) in terms of the Z transform
Identification - FINDS B(q-1)/A(q-1) parameters of Z transform

EXTERNAL

External Calculated - Makes a Call to an External DLL

MATRIXES
matrix - Matrix = (matrix1) oper (matrix 2)
matrix Oper1,Oper2 - Scalar Operations with Matrix
matrix I(t-to)...I(t-tn) - matrix of a signal delayed in time
concat. matrix - concatenates two matrices
matrix of blocks - Each Row or Column is a parent Block
matrix - Matrix = operation(matrix1)
Matrix Element - Value from an element of a matrix

BOOLEAN

ON OFF - Boolean Switch

FFT

FFT - Complex Fast Fourier Transform

Annexe B - Langage de Programmation pour les pilotes des Cartes d'Acquisition

Avant de décrire les cinq instructions disponibles pour programmer les pilotes des cartes d'acquisition, les opérateurs (variables) disponibles sont d'abord définis:

LISTE DES OPERATEURS

number (nombre) - valeur constante entrée par le clavier.

Port - valeur à lire d'un registre d'une carte ou à écrire vers le registre d'une carte, dépendant selon qu'il est employé dans le calcul du résultat ou comme résultat lui-même.

Value (valeur) - entrée pour le pilote quand une sortie doit être envoyée par la carte ou sortie du pilote quand une entrée doit être lue de la carte.

Channel (canal) - canal de la carte d'où une valeur doit être lue ou canal où une valeur doit être écrite vers la carte.

Geral, Gain, Var1, Var2 - variables qui peuvent être employées pour garder des résultats intermédiaires.

LISTE DES INSTRUCTIONS

WAIT (attendre) - arrête l'exécution du programme pendant un certain nombre de millisecondes.

RES:=op1,op2 - exécute l'opérateur 1 (op1) avec l'opérateur 2 (op2) et écrit le résultat dans la variable RES. L'Op1, op2 et RES sont choisis dans la Liste des Opérateurs. L'opération exécutée est choisie dans la Liste des Opérations.

op1, op2 ? RES - exécute op1 avec op2 et compare le résultat avec RES. Si la comparaison est vraie, alors le programme continue; si elle est fausse, alors il suspendra l'exécution pour le retard (*Delay*) défini, et ensuite il recommence. Si après le nombre défini de tentatives la comparaison est encore fausse, alors une erreur est affichée. L'op1, op2 et RES sont choisis dans la Liste d'Opérateurs. La comparaison exécutée est choisie dans la Liste des Comparateurs.

GOTO OFFSET - saute vers la ligne du programme en Offset.

BEEP f,t - donne un signal sonore avec la fréquence 'f' pour un temps 't'.

LISTE DES OPERATIONS

none (aucune) - aucune opération n'est exécutée.

NOT - opération de négation des bits de op1.

AND - opération de ET logique entre les bits de op1 et de op2.

OR - opération de OU logique entre les bits de op1 et de op2.

XOR - opération de OU EXCLUSIF logique entre les bits de op1 et de op2.

SHL - déplacement des bits de op1 vers la gauche selon le nombre de positions indiqué par op2.

SHR - déplacement des bits de op1 vers la droite selon le nombre de positions indiqué par op2.

+ - opération algébrique d'addition de op1 avec op2.

- - opération algébrique de soustraction de op1 avec op2.

* - opération algébrique de multiplication de op1 avec op2.

/ - opération algébrique de division de op1 par op2.

MOD - le reste de la division entière de op1 par op2.

LISTE DES COMPARATEURS

?? - toujours vrai

= - vrai si le résultat de l'opération de op1 par op2 est égal à RES.

<> - vrai si le résultat de l'opération de op1 par op2 est différent de RES.

>= - vrai si le résultat de l'opération op1 par op2 est plus grand ou égal que RES.

<= - vrai si le résultat de l'opération op1 par op2 est moindre ou égal que RES.

> - vrai si le résultat de l'opération op1 par op2 est plus grand que RES.

< - vrai si le résultat de l'opération op1 par op2 est moindre que RES.

Dans chaque ligne de code, il y a une valeur de décalage qui s'applique là où la variable PORT est employée. Elle indique le décalage par rapport à l'adresse de base du registre de la carte.

Annexe C - Contrôle d'Accès de l'Utilisateur

Un accès de login avec nom et mot-de-passe est disponible dans ACRUN pour donner plus ou moins de prérogatives (aussi appelé privilèges) à l'utilisateur, ou dans des termes d'usine, à l'opérateur. Le choix du mot-de-passe est fait, d'une façon 'traditionnelle', par l'instrumentiste et peut être modifié par l'utilisateur. D'autres méthodes existent comme '*System Generated*', '*Passphrases*', '*Cognitif Password*' comme décrit dans [ZVIRAN 93].

Chaque catégorie de fonction est normalement orientée vers un type d'utilisateur [DESPRÈS 91], donc dans ACRUN au lieu d'avoir des prérogatives pour l'utilisateur individuel, il y a une division selon les catégories d'utilisateurs. Les quatre niveaux (catégories) de privilèges réalisés et leurs emplois typiques sont:

ANYONE (Tout le Monde) - quand il n'y a personne identifié ou si un utilisateur ne requiert aucun privilège spécial. Dans ce dernier cas, il peut servir à identifier qui est en charge du programme pendant une certaine période de temps (utile dans la constatation des alarmes).

OPERATOR (Opérateur) - l'aptitude lui est habituellement donnée de constater la plupart des alarmes, d'écrire des rapports, et de visualiser les quelques objets sur les différents écrans disponibles.

SUPERVISOR (Superviseur) - il peut habituellement changer les consignes et les paramètres de commande, initie les procédures de sauvegarde ainsi que des procédures d'entrée et de sortie du programme. Peut créer des graphiques et constater toutes les alarmes.

INSTRUMENTALIST (Instrumentiste) - toutes les actions lui sont permises comme le changement des algorithmes de commande, la création de nouveaux objets et synoptiques, le changement de la fréquence d'échantillonnage, la définition des procédures des alarmes, et ainsi de suite. Il est le seul qui puisse ajouter des nouveaux utilisateurs, attribuer leur mot de passe initial et établir leurs prérogatives.

Les privilèges des utilisateurs sont décrits dans un ordre ascendant, donc un utilisateur OPERATOR a tous les privilèges de l'utilisateur ANYONE, le SUPERVISOR tous les privilèges de l'utilisateur OPERATOR, et le INSTRUMENTALIST tous les privilèges du SUPERVISOR.

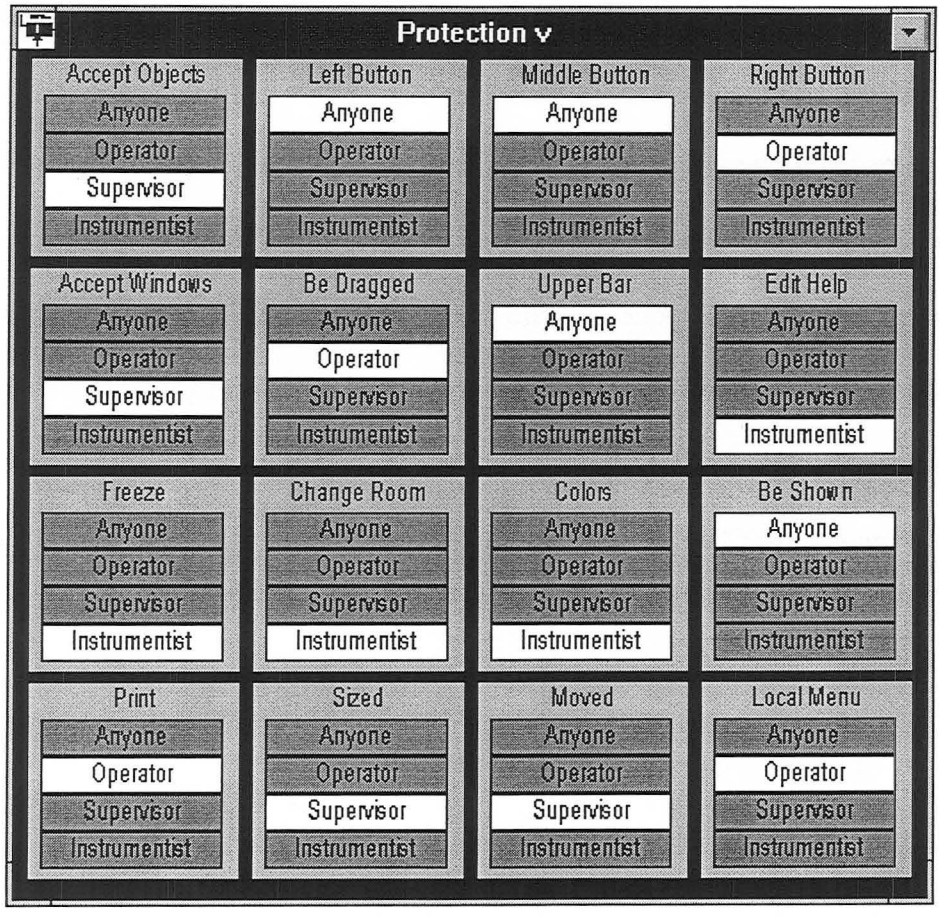


Figure 83 - Menu des Prérogatives d'un Objet

Il y a 16 actions distinctes sur chaque objet d'ACRUN qui peuvent être protégées par le Contrôle d'Accès d'Utilisateur.

Les actions qui peuvent être protégées sont:

Accept Objects (Accepter des Objets) - accepter des objets lâchés sur lui. La non accessibilité signifie que de nouveaux raccordements ne peuvent pas être établis ou cassés.

Left Button (Bouton de Gauche) - exécuter l'action correspondant au bouton gauche de la souris.

Middle Button (Bouton du Milieux) - exécuter l'action correspondant au bouton du milieux de la souris.

Right Button (Bouton de Droite) - exécuter l'action correspondant au bouton droit de la souris.

Accept Windows (Accepter des Fenêtres) - accepter les fenêtres lâchées sur lui.

Be Dragged (Être Glissé) - accepter d'être glissé pour être ensuite lâcher.

Upper Bar (Menu Supérieur) - être placé sur le menu de la barre Supérieure.

Edit Help (Modifier l'Aide) - rendre possible la modification du texte d'aide de l'objet.

Freeze (Congeler)- disposer pour ensuite détruire l'objet.

Change Room (Changer de Chambre) - déplacer l'objet vers une nouvelle chambre.

Colours (Couleurs) - changer les couleurs d'avant-plan et d'arrière-plan de l'objet.

Be Shown (Être Visualisé) - pouvoir être affiché sur l'écran.

Print (Imprimer)- permettre l'impression de l'objet.

Sized (Changer de taille) - changer la taille de l'objet.

Moved (Déplacer)- changer la position de l'objet.

Local Menu (Menu Local) - accéder au menu local de l'objet.

Comme tout dans ACRUN peut être considéré un objet, tout peut bénéficier de la protection décrite au-dessus. Une protection hiérarchique est en place, signifiant par exemple que si une chambre a les privilèges pour changer les Couleurs disponibles seulement aux instrumentistes, alors tous les objets dans cette pièce peuvent seulement avoir leurs couleurs changées par un Instrumentiste. Cela facilite la protection simultanée de l'accès à un grand nombre d'objets et est cohérent avec le monde réel: s'il n'est pas possible d'accéder à une chambre, alors il n'est pas possible d'accéder aux objets qui sont dedans.

Finalement, un historique de tous les utilisateurs qui sont entrés dans le système est disponible aux instrumentistes. Il consiste en périodes de temps où les utilisateurs ont été en charge du programme et il contient les noms des utilisateurs, la date et le temps de leur entrée (*login*)et de leur sortie (*logout*). Les échecs des accès (quand le nom et le mot-de-passe ne correspondent pas) sont aussi enregistrés pour que des multiples tentatives d'accès non autorisées puissent être détectées et enquêtées.

La réalisation du contrôle d'accès individualisé par des objets peut être employée pour supporter des vues multiples ou des interfaces multiples, par exemple en montrant ou en cachant des objets dépendant des privilèges des utilisateurs [HAILPERN 90]. Le contrôle d'accès et la sécurité dans le travail en réseau peut impliquer davantage de développements à ACRUN parce que le partage des clés cryptographiques doit être réalisé. Pour ces questions et d'autres en relation avec la sécurité, voir les bonnes références bibliographiques dans [LU 90].

Annexe D - Programmation Orientée Objet

Le langage informatique Pascal employé pour développer ACRUN n'est pas un standard (comme Fortran ou C) et a été développé par Borland International. En tant que langage classique de programmation fonctionnelle, Pascal a été étendu avec des aptitudes orientées objet pour que les programmeurs puissent employer les grands avantages qu'il offre sans sacrifier l'expérience déjà acquise.

Sans avoir la prétention de donner un cours préliminaire de programmation orientée objet (pour cela voir [BUDD 91]), l'objectif est de souligner certaines techniques employées dans le développement d'ACRUN.

Un objet est composé de données (variables) et de méthodes (fonctions et procédures) pour agir sur ces données. Chaque objet appartient à une Classe d'Objet caractérisée par ses données, méthodes et dans beaucoup de cas par une Classe d'Objet Parent de laquelle il hérite des données et des méthodes. Un objet peut accéder simultanément à ses méthodes et à ses propres données ainsi qu'à celle de la classe Parent (s'il y en a une), de la classe Parent de la classe Parent, et ainsi de suite.

Les noms des données d'une classe Parent ne peuvent pas être répétées dans la classe Enfant (celle qui hérite du Parent). Cependant, les noms de méthode d'une classe Parent peuvent être remplacés par une méthode de l'Enfant.

Cela est une caractérisation simpliste des objets dans Borland Pascal. Voyons un exemple d'application d'une manière qui donne toute sa puissance à la réalisation d'ACRUN.

La Classe fondamentale d'objet dans le programme est le Rectangle qui peut être résumé par:

```
RECTANGLE = OBJECT OF
                    xi,yi,xf,yf:LONGINT;          (* variables *)
                    forColor, bakColor : LONGINT; (* variables*)
                    DRAW;                         (* méthode*)
                    SAVE;                         (* méthode*)
                    LOAD;                         (* méthode*)
                    MOVE(dx, dy : LONGINT);       (* méthode*)
                END;
```

La procédure (méthode) DRAW peut être quelque chose comme

```
PROCEDURE RECTANGLE.DRAW;
BEGIN
    FILL_RECTANGLE(xi,yi,xf,yf,bakColor);
END;
```

La procédure SAVE peut être quelque chose comme

```
PROCEDURE RECTANGLE.SAVE;
BEGIN
```

```
    WRITELN(file,xi,' ',yi,' ',xf,' ',yf,' ',bakColor);
END;
```

Et la procédure MOVE peut être quelque chose comme

```
PROCEDURE RECTANGLE.MOVE;
BEGIN
    ERASE_SCREEN;
    xi:=xi+dx;  yi:=yi+dy; xf:=xf+dx; yf:=yf+dy;
    DRAW;
END;
```

Dans un programme, une variable 'BLOCK' peut être déclarée du type RECTANGLE et le dessin de l'objet à l'écran sera simplement 'BLOCK.DRAW'.

Certains avantages peuvent déjà être énoncés dans l'utilisation de la programmation objet:

- toutes les méthodes réalisées doivent résider dans le même fichier de définition de l'objet. Cela oblige le programmeur à garder le projet ordonné et concentre les caractéristiques d'un objet toutes ensemble.
- la façon d'appeler une méthode d'un objet est plus explicite que le traditionnel DRAW(Block) où block semble être une variable indépendante de la procédure.
- à l'intérieur d'une méthode, les méthodes et variables de l'objet sont directement accessibles, ce qui conduit à un programme plus clair (aucun besoin de écrire 'rect.xi:=rect.xi+dx' ou même 'WITH rect DO xi: =xi+dx').

D'autres classes d'objet existent dans ACRUN qui héritent de la classe RECTANGLE, de toutes ses données et de ses méthodes, mais ajoutent des données supplémentaires et remplacent certaines méthodes.

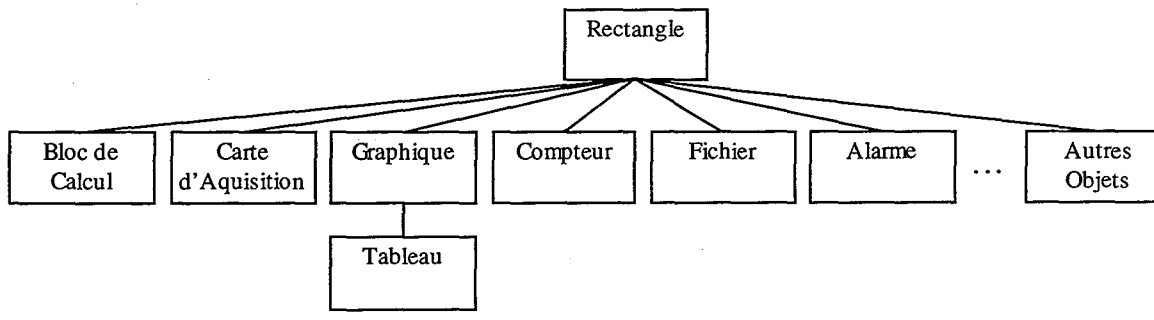


Figure 84 - Hiérarchie des Classes d'Objet

Voyons comment la Classe Compteur peut être réalisée:

```
METER = OBJECT (RECTANGLE) OF
            value : REAL;
            DRAW;
            SAVE;
            LOAD;
            INPUT_VALUE;
      END;
```

La procédure DRAW peut être réalisée de la façon suivante:

```
PROCEDURE METER.DRAW;
BEGIN
    FILL_RECTANGLE(xi,yi,xf,yf,bakColor);
```

```
    SET_COLOR(forColor);
    DRAW_TEXT((xi+xf) DIV 2, (yi+yf) DIV 2, REAL_TO STRING(value));
END;
```

Et la procédure SAVE peut être, elle, de la façon suivante:

```
PROCEDURE METER.SAVE;
BEGIN
    INHERITED SAVE;   (* appelle la méthode SAVE de la Classe Parent *)
    WRITELN(file,value);
END;
```

A partir de cet exemple, il est facile de comprendre comment la programmation objet fonctionne. L'objet Enfant hérite de toutes les données des Parents et peut les employer comme si elles étaient ses propres données. L'objet Enfant peut outrepasser les méthodes des Parents si nécessaire. Dans DRAW, comme le Parent ne représente évidemment pas la valeur réelle de l'Enfant, celui-ci a pris à sa charge de s'afficher lui-même à l'écran. Au contraire, dans la méthode SAVE de l'objet Enfant, il compte sur son parent pour sauvegarder les données héritées et complète l'opération en sauvegardant ses données spécifiques (la valeur réelle).

Une nouvelle méthode a été ajoutée, INPUT_VALUE, qui est spécifique à l'objet Compteur.

Enfin la méthode MOVE du Parent Rectangle n'a pas été remplacée, parce qu'elle accomplit bien sa fonction, même pour l'objet enfant Compteur. Voyons comment cela se passe:

1. Quand le sous-programme responsable pour redessiner l'écran donne l'ordre MOVE(10,20) à l'objet Compteur, il ne trouve pas en soi de méthode MOVE, donc il passe l'ordre à son Parent Rectangle.
2. Celui-ci, ayant une méthode MOVE, l'exécute en effaçant l'écran, augmentant xi et xf de 10, augmentant yi et yf de 20, et exécutant l'ordre DRAW.
3. Même si cet objet compteur exécute une méthode de la classe Rectangle (son parent) l'objet est encore le Compteur, donc comme la classe Compteur possède une méthode MOVE, elle est exécutée pour afficher l'objet sur les nouvelles coordonnées.

Comme l'ordre MOVE peut être employé avec succès par les objets qui héritent de Rectangle, il ne requiert pas d'être réécrit.

En ayant dans ACRUN beaucoup d'objets de classes différentes héritant de Rectangle (le Tableau hérite du Graphique qui lui hérite du Rectangle), un exemple sera maintenant donné sur la réalisation du noyau du programme. Il traite seulement avec des objets Rectangle donc il y a une liste de pointeurs avec tous les objets dans ACRUN qui sont des instances des Enfants (descendants) de Rectangle.

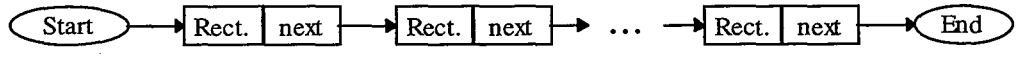


Figure 85 - Liste des Pointeurs Contenant des Objets Rectangle

Pour redessiner l'écran, il suffit de voyager à travers la liste et d'exécuter pour chaque objet l'ordre DRAW. La bonne méthode sera appelée en fonction de la Classe d'objet. Le même procédé est employé lors de la sauvegarde de tous les objets sur le disque (remplacer DRAW par SAVE) ou quand tous les objets se déplacent en même temps.

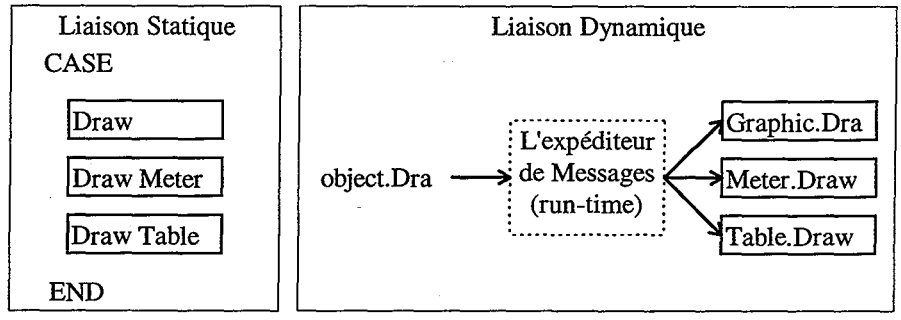
Quand un nouvel objet est créé, il suffit de l'ajouter à la liste et, automatiquement, il s'intègre avec les autres. Si une nouvelle classe d'Objet doit être créée, elle doit hériter de la classe Rectangle, définir ses données et ses méthodes, et cela est suffisant. Aucun changement dans le noyau du programme n'est nécessaire.

Comparaison du noyau du sous-programme d'affichage dans la programmation fonctionnelle et dans la programmation orientée objet:

Programmation Fonctionelle	Programmation Orientée Objet.

WHILE DataList NOT EMPTY DO
BEGIN
CASE TYPEOF(data) OF
Graphic:DRAW_GRAPHIC(data);
Meter:DRAW_METER(data);
...
Table:DRAW_TABLE(data);
END;
GET NEXT data;
END;
	WHILE ObjectList NOT EMPTY DO
	BEGIN
	object.DRAW;
	GET NEXT object;
	END;

Figure 86 - Liaison Statique par rapport à la Liaison Dynamique des Sous-Programmes



Le compilateur orienté objet réalise une procédure pour relier dynamiquement le sous-programme appelé lors de l'exécution du programme. Quand le message DRAW est envoyé (le sous-programme est appelé), il y a un expéditeur de message (attribution dynamique du sous-programme) qui délivre le message à l'objet approprié.

Une caractéristique principale des objets est qu'ils sont 'sociables'. Cela veut dire qu'ils peuvent communiquer aisément entre eux. Dans la littérature et dans certaines mises en oeuvres cela se traduit par l'envoi et la réception de messages. Cela pourrait être fait dans ACRUN en réalisant une méthode dans l'objet Rectangle appelé RECEIVE_MESSAGE et chaque classe d'enfant pourrait outrepasser cette méthode pour déchiffrer et exécuter les messages qui ont du sens pour eux. Le grand avantage du passage de message est qu'une seule méthode est requise, mais il y a un prix à payer: déchiffrer le message consomme beaucoup de temps et il ne pourrait y avoir aucune vérification des types de variables qui seraient associées avec les messages.

Dans les Blocs de Calcul où le temps de calcul peut être critique, ce passage de message ne serait pas adéquat; donc beaucoup de méthodes ont été réalisées pour échanger des données et communiquer entre les objets. Par exemple, la méthode RESET a été

réalisée et chaque objet peut répondre à cet ordre de façon appropriée (par défaut dans l'objet Rectangle, la méthode RESET est inopérante). La méthode NOME_RECT attend que l'objet renvoie son nom (un Bloc de Calcul peut renvoyer 'Sinus A' et un objet AGENDA peut renvoyer 'Réunions d'Affaires'). L'important est que l'objet Rectangle dispose de ces méthodes, et que des réponses par défaut soient toujours fournies (le compilateur oblige au respect de ces règles).

Un objet Graphique peut demander à un autre objet ses valeurs à des temps N, N+1, N+2 et représenter les valeurs qui sont renvoyées. Il peut aussi demander à l'objet ses unités et pour l'estimation, les valeurs minimum et maximum qu'il va atteindre. S'il le demande à un Bloc de Calcul, il reçoit des réponses 'intelligentes' mais s'il demande les mêmes choses à un objet Mère, le manque de réponse utile peut vouloir dire à l'objet Graphique qu'il n'y aucun intérêt à représenter cet objet. Ainsi les classes d'objets sont complètement indépendantes et c'est par la communication que les objets peuvent interagir entre eux de manière constructive.

La façon dont que chaque Classe d'objets est liée aux autres est spécifique à elle-même. Pour représenter des raccordements, le noyau d'ACRUN doit seulement demander à chaque objet une liste de tous ses objets d'entrée et une autre liste de tous ses objets de sortie. Le format des listes est standard dans ACRUN, mais la façon dont chaque objet réalise intérieurement les connexions ne présente aucun intérêt pour le noyau. Chaque classe d'objet a la liberté totale de choisir les structures de données qui lui conviennent le mieux. A titre de curiosité la liste de la vraie définition de Rectangle (tipoRectangulo) est présentée ci-dessous avec les données et les définitions des méthodes qui la

constitue. On peut en effet qu'il hérite de la classe tipoLinha (TypeLigne) qui hérite de la classe RECT qui, elle, a les cordonnées xi, yi, xf et yf.

```
rect=OBJECT(raizDasRaizes)
      xi,yi,xf,yf : tipoInt;
      PROCEDURE save;VIRTUAL;
      PROCEDURE load;VIRTUAL;
    END;

tipoLinha=OBJECT(rect)
        corTinta:TipoColorRef;
        ecran,paiDesteObjecto:PonteiroTipoRectangulo;
        CONSTRUCTOR iguala(linha:tipoLinha);
        PROCEDURE soIguala(linha:tipoLinha);
        PROCEDURE igualaDim(VAR linha:tipoLinha);
        PROCEDURE igualaDimRel(xIni,yIni,dx,dy:tipoInt);
        CONSTRUCTOR inicia(a,b,c,d:tipoInt;corT:tipoColorRef);
        PROCEDURE load;VIRTUAL;
        CONSTRUCTOR iniciaRel(x,y,dx,dy:tipoInt;corT:tipoColorRef);
        DESTRUCTOR destroi; VIRTUAL;
        PROCEDURE save;VIRTUAL;
        PROCEDURE desenha; VIRTUAL;
        PROCEDURE ajustaCoordenadas;VIRTUAL;
        PROCEDURE pedeTamanhoMinimo(VAR largXmin,largYmin:tipoInt);VIRTUAL;
        PROCEDURE moveuCoordenadas;VIRTUAL;
        PROCEDURE amplia(a:tipoInt);
        PROCEDURE Translada(dx,dy:tipoInt);
        FUNCTION meioX:tipoInt;
        FUNCTION meioY:tipoInt;
        FUNCTION compX:tipoInt;
        FUNCTION compY:tipoInt;
        PROCEDURE ordena;
        FUNCTION regiaoClip:hRgn;VIRTUAL;
      END;

tipoRectangulo=OBJECT(tipoLinha)
            corRect:tipoColorRef;
            xVariavel,yVariavel,visivel:tipoBool;
            podeDesenhar:BOOLEAN;
            win_window:tipoWnd;
            ultValorCalculado:tipoComplexo;
            data:tipoData;
            regiaoVisivel:hRgn;
            hMetafile:tHandle;
            bitmapRect:hBitmap;
            soDestroiSeZero:tipoInt;
            vaiSerDestruido:BOOLEAN;
            inicioCadeiaWnd:hWnd;
            numeroDistingueRect:tipoInt;
            previlegioRect:LONGINT;
            serTipoOle:tipoBool;
            cadeiaOndeEstaRect:pointer;
            win_rectDono:ponteiroTipoRectangulo;
            apontaLugarParaMetafile:ponteiroTipoLinha;
```

```
Xarrumado,Yarrumado,Xusado,Yusado:tipoInt;
alterouTendoQueSerRedesenhado:BOOLEAN;
CONSTRUCTOR constroi;
CONSTRUCTOR inicia(a,b,c,d:tipoInt;corT,corF:tipoColorRef);
DESTRUCTOR destroi;VIRTUAL;
PROCEDURE load;VIRTUAL;
FUNCTION podefazerSaveEmDisco:BOOLEAN;VIRTUAL;
CONSTRUCTOR iniciaRel(x,y,dx,dy:tipoInt;corT,corF:tipoColorRef);
CONSTRUCTOR iguala(rect:tipoRectangulo);
PROCEDURE igualaTodasAsCoordenadas(rect:tipoRectangulo);
PROCEDURE soIguala(rect:tipoRectangulo);
PROCEDURE save;VIRTUAL;
 FUNCTION nomeDoFicheiroEmDisco(vaiEscreverEmDisco:BOOLEAN)
                                                    :STRING;VIRTUAL;
PROCEDURE setup;VIRTUAL;
PROCEDURE coordenadas(VAR a,b,c,d:tipoInt);
PROCEDURE modifica(dxi,dyi,dxf,dyf:tipoInt);
FUNCTION cadeiaRect:pointer;
PROCEDURE desenha;VIRTUAL;
PROCEDURE desenhaDeNovo;VIRTUAL;
PROCEDURE desenhadeNovoComLigacoes;
PROCEDURE repintaRect(preDesenhaEmBitmap:BOOLEAN);
PROCEDURE desenhaRect;
FUNCTION serUsado:BOOLEAN;
PROCEDURE actualizaSemEscreverNoEcran;VIRTUAL;
FUNCTION actualiza(soSaberSeActualiza:BOOLEAN):BOOLEAN;VIRTUAL;
FUNCTION calcula:BOOLEAN;VIRTUAL;
FUNCTION reset:BOOLEAN;VIRTUAL;
FUNCTION comecaUso:BOOLEAN;VIRTUAL;
PROCEDURE acabaUso;VIRTUAL;
FUNCTION retira:BOOLEAN;VIRTUAL;
PROCEDURE pontoDeLigacao(VARx,y:LONGINT;output:BOOLEAN;
                param:tipoInt; outroRect:ponteiroTipoRectangulo);VIRTUAL;
PROCEDURE apagar;VIRTUAL;
FUNCTION nomeRect:st255;VIRTUAL;
FUNCTION soNomeRect:st255;VIRTUAL;
FUNCTION menuLocal(opcao:tipoInt):STRING;VIRTUAL;
PROCEDURE menuRect(VAR accoes:tipoAccoes);VIRTUAL;
FUNCTION menuRectPrinc(VAR accoes:tipoAccoes):BOOLEAN;
PROCEDURE editaEstaOrdem(ordem:tipoInt);VIRTUAL;
PROCEDURE editaDoRectangulo(pr:ponteiroTipoRectangulo;
                                        ordem:tipoInt);VIRTUAL;
FUNCTION valorRect(ordem:tipoInt):tipoREAL;VIRTUAL;
FUNCTION stringRect(ordem:tipoInt;VARs:STRING)
                        :ponteiroTipoSTRING;VIRTUAL;
FUNCTION stringDoRectangulo(pr:ponteiroTiporectangulo;
      ordem:tipoInt;VAR s:STRING):ponteiroTipoSTRING;VIRTUAL;
FUNCTION ordemDoRectangulo(pr:ponteiroTiporectangulo;
                        ordem:tipoInt):tipoInt;VIRTUAL;
FUNCTION ordemDestaData(VAR d:tipodata):tipoInt;VIRTUAL;
FUNCTION DadosLidosRect:tipoInt;VIRTUAL;
FUNCTION bitmapDeMinhasWin:hBitmap;VIRTUAL;
FUNCTION janelaDonaMinhasWin:ponteiroTipoRectangulo;
PROCEDURE inverteQuadro;
PROCEDURE inverteCheio;
PROCEDURE colocaDentroRect(VAR r:tipoRectangulo);
FUNCTION area:LONGINT;
```

```
FUNCTION mostraNaEstrutura:BOOLEAN;VIRTUAL;
FUNCTION devePertencerAlistaDeAmostragem(varprioridade:tipoInt)
                                                    :BOOLEAN;VIRTUAL;
FUNCTION matrizRect:POINTER;VIRTUAL;
FUNCTION paramRect:POINTER;VIRTUAL;
FUNCTION apontaControlo(VAR estratCon:tipoInt):POINTER;VIRTUAL;
PROCEDURE MudaParamRect;VIRTUAL;
PROCEDURE Quadro(tipo:tipoInt);
PROCEDURE QuadroCheio(tipo:tipoInt);
PROCEDURE apagouRectangulo(pr:ponteiroTipoRectangulo;comando:WORD;
                                                redesenha:BOOLEAN);VIRTUAL;
PROCEDURE alterouVariavelNesteEndereco(enderVariavel:POINTER);VIRTUAL;
FUNCTION podeDesapagar(VARlistaDosQueVaoSerDesapagados:listaDeRect)
                                                    :BOOLEAN;VIRTUAL;
PROCEDURE apagouWindow(window:hWnd;redesenha:BOOLEAN);VIRTUAL;
PROCEDURE rectFicouInvisivel(pr:ponteiroTipoRectangulo);VIRTUAL;
FUNCTION rectQueORepresentaNoEcran:ponteiroTipoRectangulo;VIRTUAL;
FUNCTION podeDesenharNoEcran:BOOLEAN;
FUNCTION seuTipoVisivel:BOOLEAN;VIRTUAL;
FUNCTION pertenceAlistaAusar:BOOLEAN;VIRTUAL;
PROCEDURE ficouUltimo;VIRTUAL;
PROCEDURE JuntouParaEcran;VIRTUAL;
FUNCTION ligarRect(pr:ponteiroTipoRectangulo):BOOLEAN;VIRTUAL;
PROCEDURE armazenaRect(VAR fil:TEXT);VIRTUAL;
PROCEDURE desenhaRapido;VIRTUAL;
 FUNCTION modificouEsteRect(pr:ponteiroTipoRectangulo)
                                                    :BOOLEAN;VIRTUAL;
PROCEDURE  rectangulo;
 PROCEDURE retiraDaListaDeAmostragemOsQueEntender
                                    (VARlistaAmos:listaDeRect);VIRTUAL;
PROCEDURE listaDeRectInputs(VAR juntaNestaLista:listaDeRect);VIRTUAL;
PROCEDURE listaDeRectInputsComMesmoDelay
                                (VAR juntaNestaLista:listaDeRect); VIRTUAL;
PROCEDURE listaDeRectOutputs(VAR juntaNestaLista:listaDeRect);VIRTUAL;
PROCEDURE listaDeRectOutputsComMesmoDelay
                                (VAR juntaNestaLista:listaDeRect); VIRTUAL;
 PROCEDURE listaDeFilhos(VAR juntaNestaLista:listaDeRect;
                                funcaoControlo:tipoInt);V IRTUAL;
FUNCTION proprioOuFilhoNestePonto(x,y:tipoInt)
                                    :ponteiroTipoRectangulo;VIRTUAL;
PROCEDURE ficaUsado(fica:BOOLEAN);VIRTUAL;
PROCEDURE libertaMemoria;VIRTUAL;
PROCEDURE colocaRect(apaga:BOOLEAN);
PROCEDURE darEsteObjecto(lista:listaDeRect);VIRTUAL;
PROCEDURE indicaJanelasQueFoiAlterado(wParam:WORD);
PROCEDURE desenhaOSeuIcone;VIRTUAL;
PROCEDURE inicializa(VAR sucesso:BOOLEAN);VIRTUAL;
PROCEDURE mudaParaData(VAR novaData:tipoData);VIRTUAL;
FUNCTION daMeMinimoMaximo(VAR minimo,maximo:tipoReal;
                                    previsivel:BOOLEAN):BOOLEAN;VIRTUAL;
FUNCTION daMeData(VAR estaData:tipoData):BOOLEAN;VIRTUAL;
FUNCTION casasDecimaisRect:tipoInt;VIRTUAL;
FUNCTION ultDadoLidoRect:tipoInt;VIRTUAL;
FUNCTION unidadesRect:STRING;VIRTUAL;
FUNCTION textoDeY(ordem:tipoInt):STRING;VIRTUAL;
FUNCTION maxBufferRect:tipoInt;VIRTUAL;
FUNCTION numDadosNoBufferRect:tipoInt;VIRTUAL;
```

```
  PROCEDURE colocaNestaPosicaoXY(x,y:tipoInt);
  FUNCTION bitmapDeTipoDeObjecto(paraLista:BOOLEAN):hBitmap;VIRTUAL;
  FUNCTION stringNestaData(VAR estaData:tipoData;
          VAR s:STRING):BOOLEAN;VIRTUAL;
  PROCEDURE editaEstaData(estaData:tipoData);VIRTUAL;
  PROCEDURE mudaObjectosQuePodemDependerDesteValor;
  FUNCTION telefoneRect:STRING;VIRTUAL;
  FUNCTION dragObjectoNestePonto(VAR
                accoes:tipoAccoes):ponteiroTipoRectangulo;VIRTUAL;
  PROCEDURE accaoDepoisDeSerCriado;VIRTUAL;
  FUNCTION temObjectosNoQuarto:BOOLEAN;
END;
```

Annexe E - Réseaux de Petri Utilisant des Objets Agent

Le choix de réaliser des réseaux de Petri dans ACRUN a été retenu parce que c'est, à notre avis, la meilleur méthode formelle et la plus répandue pour préciser et analyser des systèmes temps réel. Les réseaux de Petri conviennent très bien pour la simulation et l'exécution, car ils dispensent le promoteur de passer d'un modèle à l'autre [JANKOANO 91]; ils sont donc bien adaptés à notre philosophie: intégrer le développement de la commande et sa mise en oeuvre. Les autres méthodes existent comme les machines d'état, d'algèbre de procédé et de tableaux d'états [HEITMEYER 94].

Les avantages des réseaux de Petri peuvent être résumés [CHAO 94] comme suit:

1. aptitude à réaliser une représentation graphique précise,
2. disponibilité de descriptions lisibles par les machines,
3. existence de techniques d'analyse pour des aspects de contrôle,
4. aptitude d'employer la méthodologie haut-vers-le-bas pour la conception,
5. possibilité d'automation de l'analyse et de la conception.

Les réseaux de Petri permettent entre autres de modéliser, d'analyser et de visualiser les évolutions avec parallélisme, synchronisation et partage de ressources.

Le placement automatique des réseaux de Petri peut être employé notamment pour des graphiques de supervision dans des systèmes flexibles temps réel [HEBRARD 92].

Le GRAFCET est un outil employé largement en France pour modéliser et décrire la partie de contrôle des systèmes discrets [DESCOTES 93]. Dans un système de production, il peut être employé pour réaliser le contrôle local conjointement avec des réseaux de Petri pour la coordination [SAYAT 93].

Les réseaux de Petri sont réalisés dans l'objet Agent. La fonctionnalité de l'Agent est d'exécuter une tâche donnée. La tâche peut être initiée par l'utilisateur ou par un autre objet, notamment un autre Agent.

Quand l'Agent est appelé par un autre objet, il y a un nombre ('1' par défaut) associé à cet appel qui est ajouté à son compteur. L'objet Agent a un seuil (également '1' par défaut) qui indique quand il devrait exécuter son action. Quand le compteur de l'objet Agent atteint la valeur du seuil, alors l'action est exécutée et le compteur est mis à '0'. L'action exécutée par un objet Agent peut inclure l'appel d'un autre objet Agent en lui passant un nombre à additionner à son compteur.

De cette façon simple, les réseaux de Petri sont réalisés dans des objets Agent qui peuvent être employés pour automatiser certaines actions, non seulement utiles dans l'automatisation industrielle, mais aussi dans des domaines comme celui de la gestion électronique de documents (pour le *groupware* et le *workflow*). Pour davantage d'information sur les réseaux de Petri et sa mise en oeuvre, voir [DICESARE 93].

A l'avenir, on peut faire appel à la mise en oeuvre des réseaux de Pétri conjointement à d'autres objets, dans beaucoup de domaines comme la validation de systèmes distribués [SOUISSI 93] qui seront discutés dans l'annexe prochaine.

Annexe F - L'Opération et Supervision en Réseaux

Ce qui a le plus grand potentiel pour révolutionner CAPE
(Computer Aided Process Engineering) jusqu'à la fin de la décennie,
ce sont peut-être les réseaux [PEKNY 91].

Windows dispose de trois variantes avec des aptitudes intrinsèques pour fonctionner en réseaux: Windows pour Workgroups, Windows 95 et Windows NT. Les trois réalisent le NetDDE qui permet que les données temps réel soient accessibles dans n'importe quel ordinateur du réseau. L'emploi du protocole de communication TCP/IP dans Windows95 et WindowsNT, qui améliore la sécurité du système, fait d'eux un bon choix pour de grandes applications industrielles.

Lors du travail en réseau, les problèmes de sécurité augmentent et des procédures de vérification doivent être réalisées [ROLIN 94]. La synchronisation d'horloges et les procédures d'attente sont compliquées à réaliser dans les réseaux [INGELS 93]. En réalisant une opération temps réel en réseau, des systèmes qui tolèrent des pannes peuvent être réalisés et essayés d'une façon pratique en utilisant des techniques avancées [KIM 89].

Vis-à-vis de tous les problèmes évoqués ci-dessus, ACRUN emploie le NetDDE de Windows pour réaliser ses fonctionnalités de réseau. Cela évite de connaître les différents protocoles de réseau qui existent et permet à l'application ACRUN d'être plus universelle (elle dépend seulement de l'API de Windows qui réalise le NetDDE).

Annexe G - L'Interface Homme-Machine d'ACRUN

"Quand les problèmes de productivité de logiciel peuvent être résolus économiquement, les applications d'opération des procédés peuvent espérer avoir des interfaces multimédia impliquant l'animation en haute résolution et de l'interaction contrôlée par la voix, par l'écriture, et par des équipements conventionnels" [PEKNY 91].

Certes, l'emploi extensif de la programmation orientée objet et la puissance croissante des ordinateurs a permis une évolution importante dans l'interface homme-machine et une grande simplification pour l'utilisateur. Dans ACRUN, un souci principal a été de considérer l'interface homme-machine comme le ciment pour la réalisation de l'intégration totale. Comme le logiciel est la base pour l'intégration logique de divers composants, l'interface homme-machine est cruciale pour l'étape plus décisive qui concerne l'intégration: permettre à l'utilisateur l'exploitation de la fonctionnalité disponible.

Comme la convivialité est un problème clé dans le développement des logiciels industriels [CHABRIER 94], la facilité d'emploi et la productivité sont les facteurs principaux pour juger une interface utilisateur. Si pour les utilisateurs initiaux la facilité peut être prépondérante, cependant la productivité est considérée comme un avantage compétitif. Ces deux facteurs sont souvent contradictoires dans leurs besoins, par exemple des actions utilisant des hiérarchies multiples de menus logiques peuvent être faciles à apprendre et intuitives à mémoriser, mais peuvent prendre beaucoup de temps dans leur exécution.

Les programmes sont jugés par leurs caractéristiques et leur facilité d'utilisation, comme dans la compagnie IBM où l'importance donnée à chacun de ces aspects est la même [SHACKEL 86].

Il existe certaines normes dans la communauté du logiciel, et, particulièrement, dans les systèmes d'exploitation et graphiques Windows. Ces normes ne sont pas statiques comme le montre la dernière version de Windows 95 (en fait, une de ses caractéristiques

principales est la nouvelle interface utilisateur). Le bouton droit étant employé comme un contrôleur d'objet a depuis longtemps été employé dans ACRUN (et dans d'autres logiciels) et fait partie maintenant aussi de la philosophie de Windows 95. Le 'Glisser - Lâcher' et la sélection de plusieurs objets en même temps avec la souris sont des caractéristiques aussi communes d'ACRUN et de Windows.

ACRUN n'emploie pas le menu principal que recommande Windows. Au lieu de la représentation iconique des objets, ACRUN les affiche en utilisant leurs données et leurs méthodes (procédures). Les icônes dans Windows, représentant des programmes ou données, ont toutes la même dimension et ne peuvent pas changer. Les images dans ACRUN, représentant des objets, peuvent modifier leur dimension et être placées n'importe où dans l'espace de travail sur l'écran. L'espace de travail peut lui-même changer de dimension et être déplacé d'une façon interactive. Cela est beaucoup plus puissant que dans les normes actuelles de Windows. Quelques-unes de ces caractéristiques sont attendues dans une version future de Windows basée sur OLE avec le nom de code 'Cairo' et prévue pour 1997.

La couleur, en tant que caractéristique importante des objets, a été largement négligée même dans des domaines comme la vision par ordinateur [WANG 94]. Dans l'interface homme-machine réalisée dans ACRUN, la couleur joue un rôle important non seulement dans la visualisation des objets, mais encore dans leur recherche.

L'interface résultante, nommée ORDER (Objects in Rooms and Desktop Ends disordeR) devrait devenir une caractéristique majeure d'ACRUN.

L'inconvénient majeur de l'interface développée est l'emploi intensif de la souris qui n'est pas toujours possible dans un environnement industriel. Cela tend à être résolu avec des écrans tactiles et avec beaucoup d'autres périphériques aujourd'hui déjà largement employés sur les ordinateurs portables.

Annexe I - Exemple de Système de Commande de Procédé

Pour illustrer davantage les besoins d'un système de commande de procédé mentionnés dans le chapitre I, un exemple est maintenant développé qui a concerné la mise au point d'une commande non linéaire d'un réacteur discontinu de polymérisation [Wang 95]. La démarche aurait été similaire pour une commande linéaire, tout en étant alors plus simple et classique.

Le réacteur chimique simulé réalise une polymérisation de styrène dans un domaine où il présente un comportement instable à cause de la forte exothermicité de la réaction. Durant l'opération, la température du réacteur doit suivre un profil donné a priori, qui pourrait être un profil optimal.

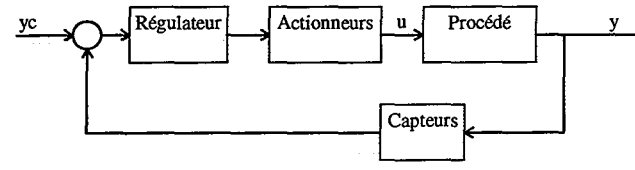


Figure 96 - Système de Contrôle par Bouclage d'un Procédé

La Figure 97 montre le système de contrôle et les connections des signaux. Le réacteur batch est le procédé, la variable contrôlée y(t) est la température de réaction (Tr) qui est mesurée par un thermocouple (capteur) entre l'ordinateur (contrôleur) par un convertisseur analogique/numérique (A/N). Le signal de contrôle ou la sortie u(t) du contrôleur, ici l'ouverture de la soupape à trois voies (actionneur) reliée à des échangeurs de chaleur chaud et froid, est donné par l'ordinateur à travers un convertisseur numérique/analogique (N/A). Le signal de référence ou consigne r(k) est emmagasiné a priori dans l'ordinateur ou calculé en ligne.

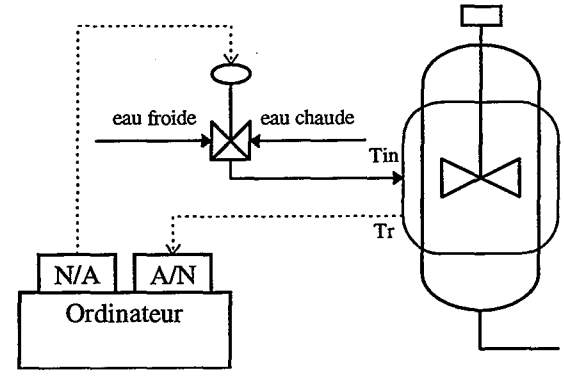


Figure 97 - Contrôle de Température dans un Réacteur Discontinu

Annexe H - Une Séance Avec ACRUN

La philosophie objet utilisé est réflétée dans l'intégration de la présentation dans ACRUN. Dans l'écran principal, on peut voir sur la droite les classes d'objets disponibles (dans le haut celles qui font parties de ACRUN et en bas celles qui sont importées des logiciels serveurs d'objets OLE). Sur la gauche, il y a des icones pour des fonction générales qui peuvent être utilisées pour tous les objets (changement de couleurs, privilèges d'utilisation, agrandissement à l'écran, etc.). Sur le haut, se trouve un menu qui peut changer pour des objets différents ou pour de la fonctionnalité génerale (sur l'image, on voit le menu correspondant aux Blocs de Calcul). En bas il y a la description de la position où se trouve le curseur de la souris et les actions correspondants aux trois boutons de la souris.

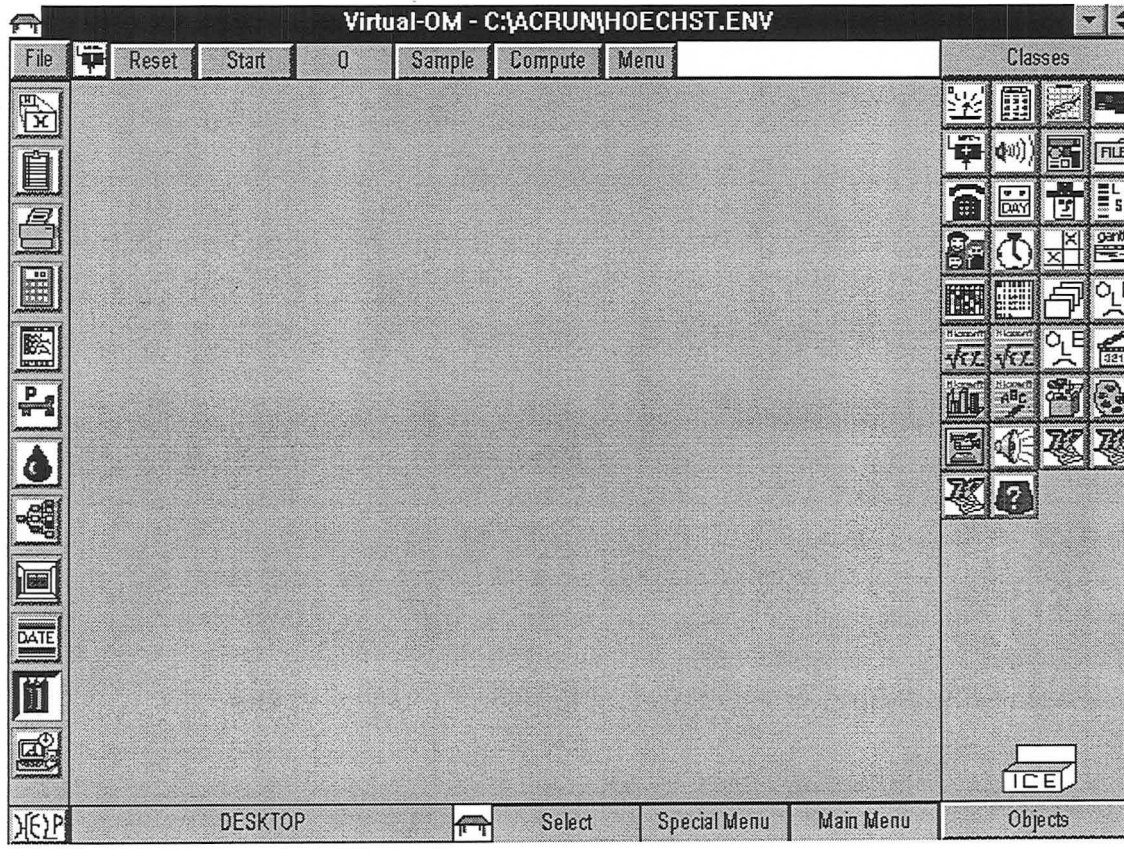


Figure 87 - Écran Principal de ACRUN

Pour créer un objet d'une classe (faire une instantiation), il suffit de faire un glisser - déplacer (Figure 88).

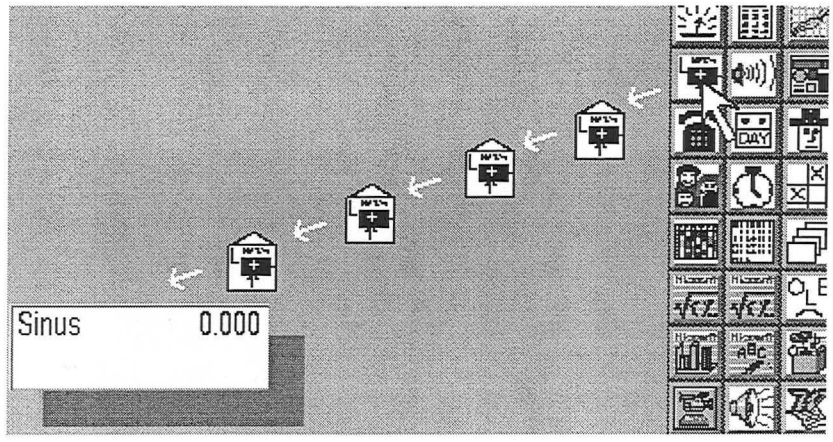


Figure 88 - Créer un Nouveau Objet

Pour connecter deux objets il suffit de faire glisser-déplacer du premier vers le second (Figure 89).

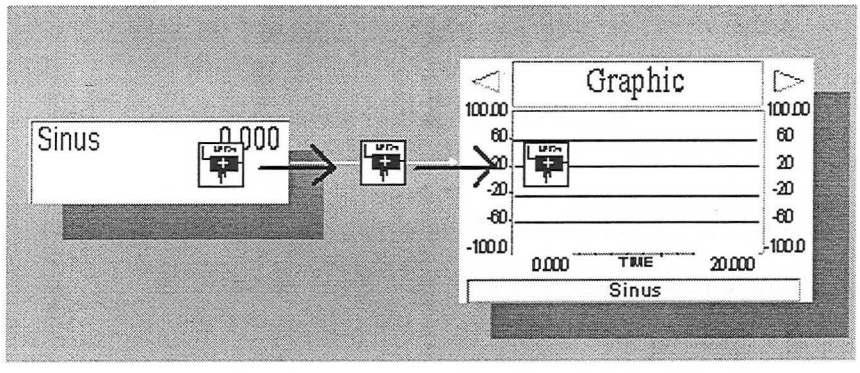


Figure 89 - Connecter un Bloc de Calcul à un Graphique

Pour initialiser une simulation, il suffit de cliquer sur le bouton *Compute*.

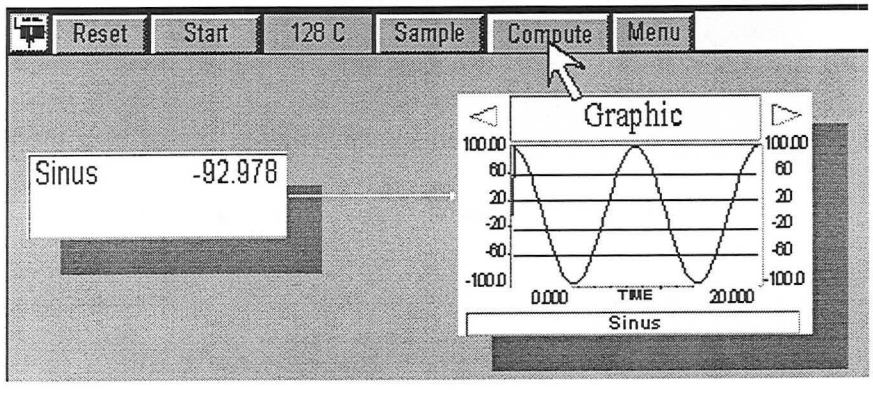


Figure 90 - Initialiser une Simulation

Pour changer les paramètres de l'acquisition, on clique avec le bouton droit de la souris dans le bouton *Sample*.

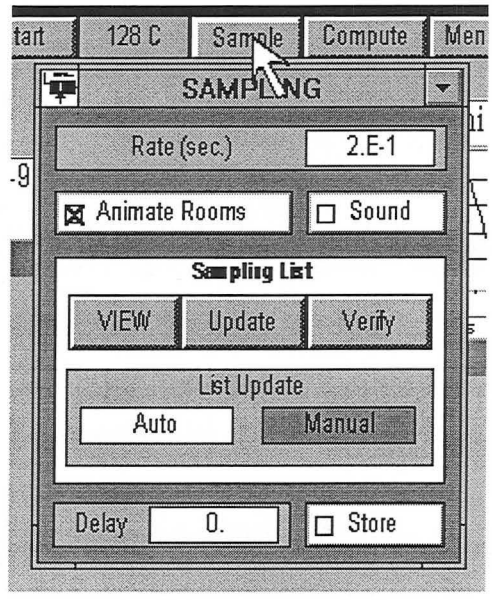


Figure 91 - Paramètres d'Échantillonnage

Pour commencer l'acquisition en temps réel, on clique sur le bouton *Start* qui, ensuite, se transforme en *Stop* et qui sert à interrompre l'acquisition.

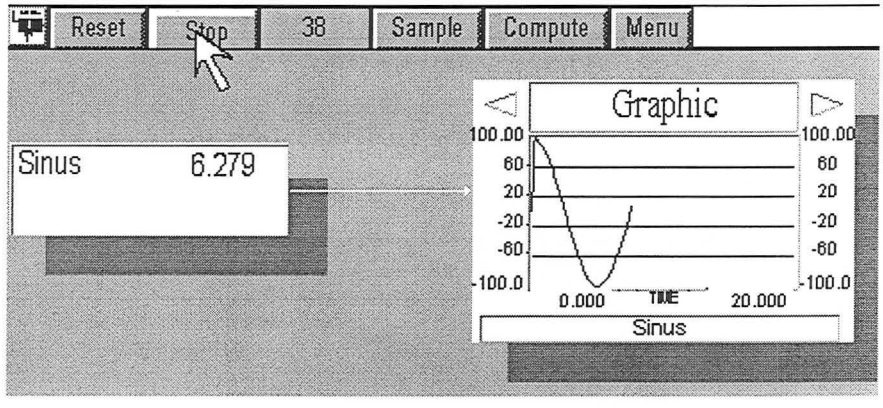


Figure 92 - Initialiser/Arrêter une Acquisition

Chaque objet a un menu local qui est accessible en cliquant sur lui avec le bouton droit de la souris. Ce menu est spécifique pour chaque classe d'objet.

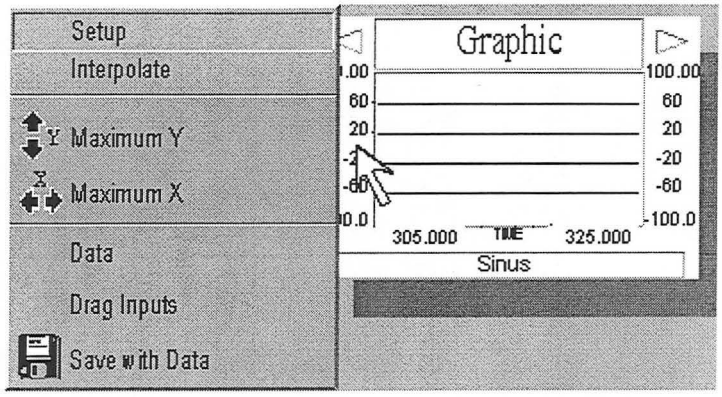


Figure 93 - Menu Local des Objets

Chaque objet a un menu qui est accessible en cliquant sur son ombre avec le bouton droit de la souris. Ce menu est le même pour toutes les classes d'objet.

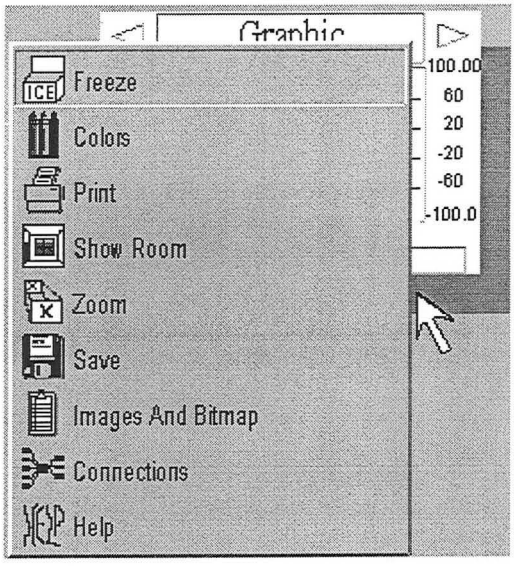


Figure 94 - Menu Objet Général

Par le menu local de l'objet Carte d'Aquisition, on peut accéder au menu de développement des pilotes pour les cartes d'acquisition et de contrôle.

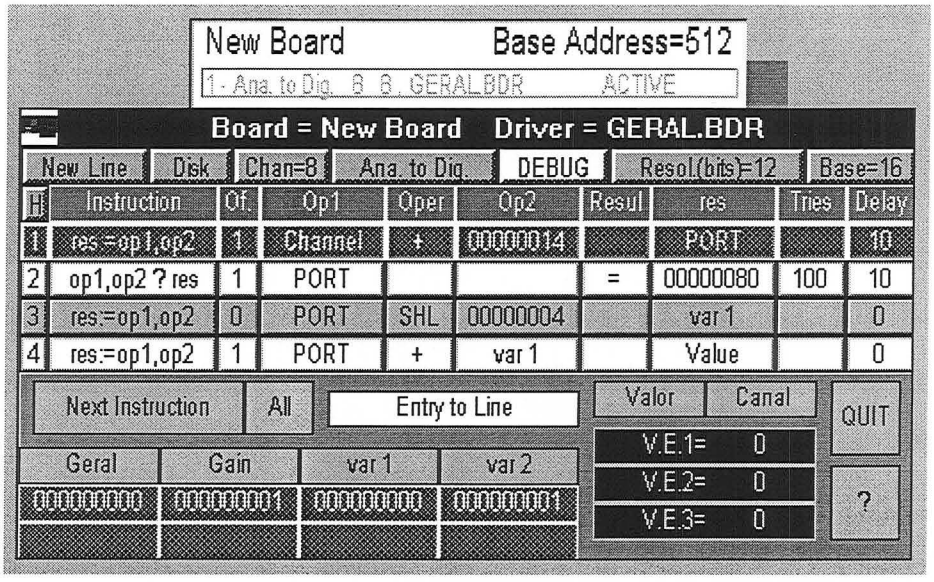


Figure 95 - Dévelopment d'un Pilote de Carte d'Acquisition

Développement

1. Traitement du Signal - le signal de la température doit être filtré avant l'identification afin de réduire l'effet des bruits de mesure et de transmission (on utilise des filtres récursifs). En commande linéaire, une fonction de transfert serait identifiée sur la base de la séquence d'entrées-sorties. En commande non linéaire, il est nécessaire d'estimer les états à partir des mesures et des entreés. Dans les deux cas, il est préférable de travailler avec des signaux pré-filtrés.

2. Identification - En commande linéaire adaptative, à chaque période d'échantillonnage, la fonction de transfert discrète du procédé est identifiée par une méthode récursive. Dans la commande non linéaire réalisée, l'observateur d'états est un filtre de Kalman étendu basé sur le modèle du procédé qui utilise les mesures pour estimer les états non mesurés qui sont ensuite injectés dans la loi de commande non linéaire. Afin de tenir compte des erreurs du modèle, de plus, certains paramètres importants du procédé sont estimés, comme le coefficient d'effet de gel et le coefficient de transfert de chaleur.

3. Contrôleur - En commande linéaire, l'algorithme de commande peut être un simple PID, une commande par modèle interne, ou une commande adaptative (comme la commande prédictive généralisée). La loi de commande non linéaire est développée dans le cadre de la géométrie différentielle, et revient grossièrement à rechercher la dérivation d'ordre n de la sortie faisant apparaître linéairement l'entrée.

4. Simulateur du Procédé - Le modèle du procédé comprend essentiellement un système d'équations différentielles ordinaires, représentant les bilans de matière incluant la cinétique réactionnelle, et les bilans de chaleur décrivant les variations de température au sein du réacteur et dans la double enveloppe. En commande linéaire, le modèle est utile seulement en simulation pour remplacer le procédé. En commande non linéaire, le modèle sert de base au développement de la commande, des observateurs d'états et des estimateurs de paramètres.

5. L'Affichage de Données - la température de référence r(t) et la température de la réaction y(t) sont toujours affichées sous forme d'une courbe en fonction du temps, de même que la sortie du contrôleur u(t) (pourcentage d'ouverture de la soupape à trois voies).

Mise en Oeuvre

1. Signaux d'Acquisition et de Contrôle (interface entrée/sortie) - au moins un convertisseur A/N (pour traduire la valeur analogique du capteur à une valeur numérique qui est utilisée par l'ordinateur) et un convertisseur N/A (pour traduire la valeur numérique de l'ordinateur à une valeur analogique qui sert comme entrée à l'actionneur) sont nécessaires et disponibles sur une carte d'acquisition insérée dans l'ordinateur.

2. Interface homme-machine - de nombreuses simulations, puis des expériences sont nécessaires avant que le système de contrôle du réacteur ne soit complètement opérationnel. Le schéma de commande par blocs comprend ainsi des blocs d'entrée, le bloc régulateur, le bloc observateur, le bloc estimateur, les blocs capteurs, les blocs de consignes et de sorties. L'influence de certains paramètres d'entrée ou de sortie de ces blocs a pu être visualisée en cours de simulation afin de mettre au point la commande. Les synoptiques du réacteur batch peuvent afficher l'indication des valeurs des variables dans leur place correspondante (température de l'échangeur froid, chaud, de la double enveloppe, du réacteur). Le contrôle par l'opérateur de l'ouverture de la soupape à trois voies est aussi possible soit par une boîte de dialogue ou un bar-graph.

3 . Alarmes et Sécurité - une alarme peut être déclenchée si la température dépasse une certaine valeur, soit en raison d'un disfonctionnement de l'équipement, soit en raison de la chaleur de réaction insuffisamment dissipée. La sécurité peut être conçue pour limiter la possibilité de changement des paramètres du système à un certain type d'utilisateur (par exemple, à l'instrumentiste).

4. Exécution temps réel - Sur un réacteur discontinu de ce type, une période d'échantillonnage de l'ordre de cinq secondes est suffisante. Les mesures principales sont la température de la double enveloppe et surtout la température au sein du réacteur qui est la sortie commandée. Le système ne présente pas de retard véritable. La période d'échantillonnage est suffisamment élevée pour permettre l'estimation des états et des paramètres et le calcul de la loi de commande. Ces actions sont prioritaires par rapport à d'autres événements tels que l'affichage de données, l'exécution d'autres programmes tournant concurremment dans le même ordinateur, ou toute autre opération de sauvegarde et de maintenance de l'ordinateur.

A travers un exemple d'automatisation en génie chimique comme le réacteur discontinu de polymérisation, il est possible de mieux appréhender les termes généraux employés dans le chapitre I et partout, dans ce travail, où les questions de mise en oeuvre sont abordées. Néanmoins, notre but concerne essentiellement les problèmes d'intégration utilisant des techniques du génie logiciel et non des problèmes d'ingénierie chimique ou d'automatique.

BIBLIOGRAPHIE

[ALTY 95] J. L. Alty, M. Bergan, Multimedia Interfaces for Process Control: Matching Media to Tasks, Control Engineering Practice, Vol.3(2) pp.241, 1995

[AUDSLEY 94] N.C. Audsley, A. Burns, M. F. Richardson, A. J. Wellings, STRESS: A Simulator for Hard Real-Time Systems, Software - Practice and Experience Vo.24(6), pp.543, 1994

[BACKSH 94] B. R. Backsh, G. S. Stephanopoulos, Representation of Process Trends - IV. Induction of Real-Time Patterns from Operating Data for Diagnosis and Supervisory Control, Computer and Chemical Engineer Vol.18(4) pp.303, 1994

[BARKER 95] H. A. Barker, Open Environments and Object-Oriented methods for Computer-Aided Control System Design, Control Engineering Practice, Vol.3(3) pp.347, 1995

[BARTHES 94] J.-P. A. Barthès, Developing Integrated Object Environments for Building large Knowledge-Based Systems, International Journal of Human-Computer Studies Vol.41(1-2) pp.33, 1994

[BELL 92] Doug Bell, Ian Morrey, John Pugh, Software Engineering: A Programming Approach - 2nd ed., Prentice Hall International, 1992

[BENSON 89] R. S. Benson, Process Systems Engineering: Past, Present and a Personal View of the Future, Computer and Chemical Engineer Vol.13 pp.1193, 1989

[BENSON 92] R. S. Benson, Computer Aided Process Engineering - an Industrial Perspective, Computer and Chemical Engineer Vol.16(4) pp.S.1, 1992

[BENSON 94] R. S. Benson, The Process Industry requirements of Advanced Control techniques: Chalenges and Opportunities, ADCHEM'94 pp.215, 1994

[BERRY 94] D.C. Berry, Involving Users in Expert System Development, Expert Systems Vol.11(1) pp.23, 1994

[BLAKELY 94] J. A. Blakely, D. Fishman, D. Lomet, M. Stonebraker, D. Barabará, panel: the Impact of database research on Industrial Products (Summary), ACM SIGMOD Record Vol.23(3) pp.35, 1994

[BOOCH 94] G. Booch, Coming of Age in an Object-Oriented World, IEEE Software, Vol.11(6) pp.33, 1994

[BRAMS 83] G.W. Brams (ouvrage colectif), Réseaux de Petri: Théorie et Practique, Masson, 1983

[BROOKS 87] F. P. Brooks, No Silver Bullet: Essence and Accidents of Sofware Development, IEEE Computer Vol.20(4) pp.10, 1987

[BUDD 91] Timothy Budd, An Introduction to Object Oriented Programming, Addison Wesley Pblishing Company Inc, 1991

[BUDD 91] Timothy Budd, Introduction à la Programmation par Objets, Addison Wesley, 1991

[BURNS 92] A. Burns, J.McDermid, J. Dobson, On the Meaning of Safety and Security, The Computer Journal, Vol.35(1) pp.3, 1992

[CADOZ 94] C. Cadoz, Le Geste Canal de Communication Homme/Machine RAIRO Technique et Science Informatiques Vol.13(1) pp.31, 1994

[CENA 95] G. Cena, L. Durante, A. Valenzano, Standard Field Bus Networks for Industrial Applications, Computer Standards & Interfaces, Vol.17(2), pp.155, 1995

[CHABRIER 94] B. Chabrier, P. Franchi-Zannettaci, Delegation Semantique par Contraintes Réactives pour les Interfaces Graphiques, RAIRO Technique et Science Informatiques Vol.13(4) pp.539, 1994

[CHAO 94] D. Y. Chao, D. T. Wang, An Interactive Tool for Design, Simulation, Verification and Synthesis of Protocols, Software - Practice and Experience Vo.24(8), pp.747, 1994

[CHATTY 94] S. Chatty, De la Manipulation Directe à l'Animation: la Dynamique de l'Interface, RAIRO Technique et Science Informatiques Vol.13(1) pp.63, 1994

[CHEN 94] C. Chen, S. Lee, G. Santamarina, An Object-Oriented Manufacturing Control System, Journal of Inteligent Manufacturing Vol.5(5) pp.315,1994

[COCHLOVIUS 93] E. G. Cochlovius, K. Schnitzlein, The System Editor and Simulator SES: Applying a Digital Simulator in Chemical Engineering Education, Computer and Chemical Engineer Vol.17(8) pp.785, 1992

[CORRIPIO 88] A. Corripio, A. M. Sterling, Computer Integrated Manufacturing in the Process Industry, AIChe Annual Meeting Washington DC, Paper 36A, 1988

[CUI 93] X. Cui, K.G. Shin, Direct Control and Coordination Using Neural Networks, IEEE Transactions on Systems, Man, and Cybernetics, Vol.23(3) pp.686, 1993

[DANFORTH 88] S. Danforth, C. Tomlinson, ACM Computing Surveys Vol.20(1) pp.29, 1988

[DAVIS 94] J. F. Davis, Online Knowledge Based Systems in Process Operations: the Impact of Sctructure on Integration, ADCHEM'94 pp.109, 1994

[DEDENE 94] G. Dedene, M. Snoek, M.E.R.O.D.E.: A Model Driven Entity-Relationship Object-Oriented Development Method, Software Engineering Notes Vol.9(3) pp.51, 1994

[DESCOTES 93] B. Descotes-Genon, Z. S. Abazi, GRAFSUR, to make into account safety in Grafcet, RAIRO Automatique Productique Informatique Industriel Vol.27(1) pp.39, 1993

[DESPRES 94] F. M. Després, Automatisation de Systèmes de Production - du Besoin a l'Utilisation, Éditions Kirk, 1994

[DICESAR 93] F. Dicesar, G. Harhalakis, J. M. Proth, M. Silva, F. B. Vernadat, Practice of Petri Nets in Manufacturing, Chapman & Hall, 1993

[DRENICK 90] R. F. Drenick, Opportunities in Organizational Control, IEEE Control Systems Magazine, Vol.10(5) pp.27, 1990

[DUKE 95a] D.J. Duke, M. D. Harrison, Event Model of Human-System Interaction, Software Engineering Journal, Vol.10(1) pp.3, 1995

[DUKE 95b] D.J. Duke, M. D. Harrison, Mapping user Requirements to Implementation, Software Engineering Journal, Vol.10(1) pp.13, 1995

[GRAELLS 94] M. Graells, A. Espuna, L. Puigjaner, An Efficient Industry-Oriented Approach for the Scheduling with Intermediate Storage of Multipurpose Batch Chemical Plants, ESCAPE 4 ICHEM^E N°133 pp.17, 1994

[GUY 94] K.W.A. Guy, The Industrial Challenge: the process Industries in the 21st Century, ESCAPE 4 ICHEME N°133 pp.479, 1994

[HAILPERN 90] B. Hailpern, H. Ossher, Extending Objects to Support Multiple Interfaces and Access Control, IEEE Transactions on Software Engineering Vol.16(11) pp.1247, 1990

[HAKJUNG 94], J. Hakyung, I.-B. Lee, D. R. Yang, K. S. Chang, Optimal Scheduling for Serial Multi-Product Batch Processor with Variuous Storage Policies Under Consideration of Non-Zero Transfer Times and Setup Times, ESCAPE 4 ICHEME N°133 pp.79, 1994

[HARTSON 89] H.R. Hartson, D. Hix, Human-Computer Interface Development, ACM Computing Surveys Vol.21(1) pp.5, 1989

[HATTON 94] L. Hatton, A. Roberts, How Accurate is Scientific Software, IEEE Transactions on Software Engineering Vol.20(10) pp.785, 1994

[HE 94] J. He, Z. Mammeri, J.-P. Thomesse, Modelisation des techniques de Synchronisation d'Horloges, RAIRO Technique et Science Informatiques Vol.13(2) pp.185, 1994

[HEBRARD 92] A. Hébrard, J.P. Bourey, J.C. Gentina, RAIRO Automatique Productique Informatique Industriel Vol.26(3) pp.209, 1992

[HEITMEYER 94] C. Heitmeyer, U. Buy, Succedings of the Seventh International Workshop on Software Specification and Design - Real Time Systems, Software Engineering Notes Vol.9(3) pp.19, 1994

[HENDERSON 94] R. Henderson, B. Zorn, A Acomparison of Object-Oriented Programming in Four Modern Languages, Software - Practice and Experience Vo.24(11), pp.1077, 1994

[HOMAYOUN 94] N. Homayoun, P. Ramanathan, Dynamic Priority Schedulling of Periodic and Aperiodic Tasks in Hard Real-Time Systems, Real Time Systems Vol.6(2) pp.207, 1994

[HORN 93] F. Horn, J.B. Stefani, On Programming and Supporting Multimedia Object Synchronizaton, The Computer Journal, Vol.36(1) pp.4, 1993

[HSU 94] C. Hsu, L.Gerhardt, D. Spooner, A. Rubenstein, Adaptive Integrated Manufacturing Enterprises: Information Technology for the Next Decade, IEEE Transactions on Software Engineering, Vol.24(5) pp.828, 1994

[HURLEY 93] W. D. Hurley, A Process model for Interactive Systems, Journal of Systems Integration Vol.3(3/4) pp.251, 1993

[INGELS 93] P. Ingels, C. Maziero, M. Raynal, Un Noyau Réparti pour les Applications Fondées sur la Progression d'Un Temps Virtuel, Reseaux et Informatique Répartie Vol.3(2) pp.145, 1993

[JACOBSON 93] I. Jacobson, Is Object Technology Software's Industrial Platform?, IEEE Software Vol.10(1) pp.24, 1993

[JAFFE 91] M.S. Jaffe, N.G. Leveson, M.P.E. Heimdahl, B.E. Melhart, Software Requirements Analysis for Real-Time Process ControlSystems, IEEE Transactions on Software Engineering Vol.17(3) pp.241, 1991

[JOHNSON 88] R. E. Jonhson, B. Foote, Designing Reusable Classes, Journal of Object Oriented Programming Vol.1(2) pp.22, 1988

[KIM 90] W. Kim, Introduction to Object Oriented Databases, MIT Press, Cambridge, MA, 1990

[KIM 95] K. H. Kim, An Approach to Experimental Evaluation of Real-Time Fault Tolerant Distributed Computing Schemes, IEEE Transactions on Systems, Man, and Cybernetics, Vol.15(6) pp.715, 1989

[LABTECH 92] Laboratory Technology Ltd, Labtech/Control User Manual, 1992

[LANDAU 90] I. D. Landau, System Identification and Control Design, Prentice Hall, 1990

[LARMINAT 91] Ph. de Larminat, La Commande Robuste: un Tour d'Horizon, RAIRO Automatique Productique Informatique Industriel Vol.25(3) pp.267, 1991

[LEBLANC 94] L. LeBlanc, T. Jelassi, An Empiracal Assessment of Choice Models for Software Slection - A Comparison of the LWA and MAUT Techniques, Révue des Systèmes de Decision, Vol.3(2) pp.115, 1994

[LEVESON 90] N. G. Leveson, The Challenge of Building Process-Control Software, IEEE Software Vol.7(6) pp.24, 1990

[LEVESON 94] N.G. Leveson, M.P.E. Heimdhal, H. Hildreth, J.D. Reese, Requirements Specification for Process Control Systems, IEEE Transactions on Software Engineering Vol.20(9) pp.684, 1994

[LU 90] W.-P. Lu, M.K. Sundareshan, A Model for Multilevel Security in Computer Networks, IEEE Transactions on Software Engineering Vol.16(6) pp.647, 1990

[LURIE 95] R. E. Lurie, I.M. MacLeod, Digital Sampling Rate Reduction - A Guide for Practising Control and Instrumentation Engineers, Control Engineering Practice, Vol.3(2) pp.171, 1995

[MARQUARDT 92] W. Marquardt, An Object Oriented Representation of Structured Process Model, Computer and Chemical Engineer Vol.16 Suppl. pp.S.329, 1992

[MATLAB 92 b] Andrew Grave, Man J. Laub, John N. Little, Clay M. Thompson, Control System Toolbox User's Guide, July 1992

[MATLAB 92a] Matlab Reference Guide, The Mathworks Inc, August 1992

[MCCROSKEY 91] P. S. McCroskey, J.M. Wassick, Integration of Model Development & Advanced Process Control, COPE 91 pp.293, 1991

[MELLIER 95] P. Mellier, F. Grize, OVIDE: a Tool for Data Acquisition and Validation, ACM SIG Plan Notices Vol. 30(1) pp.53, 1995

[MESSINA 94] G. Messina, G. Tricomi, Software Standardization Integrating Industrial Automation Systems, Computers in Industry, Vol.25(2) pp.113, 1994

[MOTARD 89] R. L. Motard, Integrated Computer Aided Process Engineering, Computer and Chemical Engineer Vol.13 pp.1199, 1989

[NEELAMKAVIL 90] F. Neelamkavil, O. Mullarney, Separating Graphics From Application in the Design of User Interfaces, The Computer Journal, Vol.33(5) pp.437, 1990

[NUMAO 95] M. Numao, An Integrated Scheduling/Planing Environment for Petrochemical Production Processes, Expert Systems with Applications Vol.8(2), pp.263, 1995

[PANGALOS 93] G. J. Pangalos, Designing the User Interface, Computers in Industry, Vol.22(2) pp.193, 1993

[PEKNY 91] J. Pekny, V. Venkatasubramanian, G. V. Reklaitis, Prospects for Computer-Aided Process Operations in the Process Industry, COPE 91 pp.435, 1991

[PERRIS 94] T. Perris, Specific Challenges to the CAPE Community, ESCAPE 4 ICHEM^{E} N°133 pp.507, 1994

[PREECE 91] P.E. Preece, M.H. Kift, D.M. Grills, A Graphical user Interface for Computer Aided Process Engineering Design, COPE 91 pp.209, 1991

[PREECE 94], P. Preece, T. Ingersoll, J. Tong, Specification (Data) Sheets - Picture a Database, ESCAPE 4 ICHEM^{E} N°133 pp.467, 1994

[PUIGJANER 91] L. Puigjaner, A Espuna, I. Palou, J. Torres, Aprototype Computer Integrated Manufacturing Modular Unit for Education and Training in Batch Processing Operations, COPE 91 pp.427, 1991

[RAVN 93] A. P. Ravn, h. Rischel, K. M. Hansen, Specifying and Verifying Requirements of Real-Time Systems, IEEE Transactions on Software Engineering Vol.19(1) pp.41, 1993

[RENGASWAMY 94] R. Rengaswamy, V. Venkatasubramanian, ADCHEM'94 pp.120, 1994

[RIPPIN 91] D.W.T. Rippin, Education and Training in Computer Apllications, COPE 91 pp.279, 1991

[ROLIN 94] P. Rolin, L. Mé, J. V. Gomez, Securité des Systèmes Informatiques - des Systèmes Centralizés aus Reseaux, Reseaux et Informatique Répartie Vol.4(1) pp.31, 1994

[RUSSEL 94] D. W. Russel, Software Engineering and production Monitoring Systems, Journal of Systems Integration Vol.4(3) pp.243, 1994

[RYCKBOSCH 92] J. Ryckbosch, Le Logiciel Scientifique - Conception par Objets Tec&Doc - Lavoisier, 1992

[SASSEN 94] J.M.A. Sassen, E.F.T. Buiel, J.H. Hoegee, International Journal of Human-Computer Studies Vol.40(5) pp.895, 1994

[SAYAT 93] B. sayat, Pierre Ladit, Control Specification Of a prodection System Using Grafcet, RAIRO Automatique Productique Informatique Industriel Vol.27(1) pp.53, 1993

[SCHACKEL 86] B. Schackel, IBM make usability as Important as Functionality, The Computer Journal, Vol.29(5) pp.475, 1986

[SCHMID 94] U. Schmid, Monitoring Distributed Real-Time Systems, Real Time Systems Vol.7(1) pp.33, 1994

[SILLY 94] M. Silly, Un Algorithme d'Ordonnancement de Tâches Sporadiques pour les Systèmes Temps Réel, RAIRO Automatique Productique Informatique Industrielle Vol.28(2), 1994

[SIMULINK 93] Simulink User's Guide for Microsoft Windows, The Mathworks Inc, April 1993

[SINGPURWALLA 91] N. D. Singpurwalla, Determining an Optimal Time Interval for Testing and Debugging Software, IEEE Transactions on Software Engineering Vol.17(4) pp.313, 1991

[SOLHEIM 93] J. A. Solheim, J. H. Rowland, An Empirical Study of Testing and Integration Strategies Using Artificial Software Systems, IEEE Transactions on Software Engineering Vol.19(10) pp.941, 1993

[SOUISSI 93] Y. Souissi, Compositions de Réseaux de Petri e Validation Modulaire de Systèmes Distribué, Reseaux et Informatique Répartie Vol.3(3) pp.187, 1994

[STANLEY 91] G. M. Stanley, F. E. Finch, S.p Fraleigh, An Object-Oriented Graphical Language and Environment for Real-Time Fault Diagnosis, COPE 91 pp.265, 1991

[SUH 90] N. P. Suh, University-Industry Interaction for the Inovation of New Products and Processes: A U.S. Model,Robot. Comp. Integrat. Manuf., Vol.7(1) pp.15, 1990

[TANKOANO 91] J. Tankoano, D. Boudebous, J. C. Derniame, Petri-S: Un Simulateur de Systèmes de production Automatisée Décrits à l'Aide de Reseux de petri Interpretés Colorés, RAIRO Automatique Productique Informatique Industriel Vol.25(1) pp.1, 1991

[THOMAS 92] I. Thomas, B.A. Nejmeh, Definitions of Tool Integration for Environments, IEEE Software, Vol.9(2) pp.29, 1992

[THOMESSE 93] J.-P. Thomesse, Le Reseau de Terrain FIP, Reseaux et Informatique Répartie Vol.3(3) pp.287, 1993

[VICENTE 92] K. J. Vicente, J. Rasmussen, Ecological Interface Design : Theoretical Foundations, IEEE Transactions on Systems, Man, and Cybernetics, Vol.22(4) pp.589, 1992

[VIRRANTALO 94] A. M. Virrantalo, An Object-Oriented Taxonomy of Declarative Process Knowledge, Computer and Chemical Engineer Vol.18 Suppl. pp.S.737, 1994

[WANG 94] Y. S. Wang, B.J. Griffiths, B.A. Wilkie, P.A. Silverwood, P. Norgate, Complex and Coloured Object Inspection, Computers in Industry, Vol.25(2) pp.125, 1994

[WANG 95] Z.-L. Wang, Commande Non Linéaire de Réacteurs de Polymérisation, Thèse de Doctorat I.N.P.L., Nancy, 1995

[WELKE 94] R. J. Welke, The Shifting Software Development Paradigm, Data Base, Vol.25(4) pp.9, 1994

[WHITELY 92] J. R. Whiteley, J. F. Davis, Knowledge-Based Interpretation of Sensor Patterns, Computer and Chemical Engineer Vol.16 pp.329, 1992

[WILLIAMS 94] T. J. Williams, P. Bernus, J. Brosvic, D. Chen, G. Doumeingts, L. Nemes; J.L. Nevins, B. Vallespir, J. Vlietstra, D. Zoetekouw, Architecturs for Integrating Manufacturing Activities and Enterprises, Computers in Industry, Vol.24(2-3) pp.111, 1994

[YUAN 94] X. Yuan, M.C. Saksena, A. K. Agrawala, A Decomposition Approach to Non-Preemptive Real-Time Schedulling, Real Time Systems Vol.6(1) pp.7, 1994

[ZHU 94] J. J. Zhu, Motivation-by-Challenge for teaching Control Systems, IEEE Control Systems Magazine, Vol.14(5) pp.64, 1994

[ZOMAYA 94] A.Y. Zomaya, Reinforcement Learning for the Adaptive Control of Nonlinear Systems, IEEE Transactions on Systems, Man, and Cybernetics, Vol.24(2) pp.357, 1994

[ZVIRAN 93] M. Zviran, W.J. Haga, A Comparison of Passwords Techniques for Multilevel Authentication Mechanisms, The Computer Journal, Vol.36(3) pp.227, 1993

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur BOURSEAU Patrick, Maitre de Conférence, ENSCP Paris,
Monsieur JOULIA Xavier, Professeur, ENSIGC Toulouse.

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur CARDIGOS DOS REIS Francisco

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

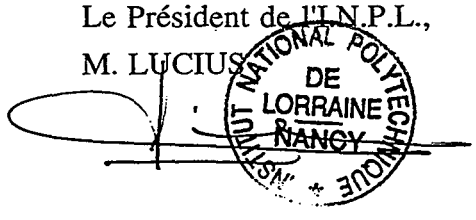
à soutenir devant l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :
"ACRUN-Contribution de la technologie orientée objet à l'Intégration Logicielle de l'Acquisition et de la Commande".

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"GENIE DES PROCEDES"**

Fait à Vandoeuvre le, **8 Février 1996**
Le Président de l'I.N.P.L.,
M. LUCIUS



TEL. 33/83.59.59.59
FAX. 33/83.59.59.55

Résumé

La conception orientée objet permet d'intégrer des outils possédant des fonctionnalités distinctes dans un même environnement. Le projet ACRUN a concerné la réalisation d'un logiciel de haut niveau d'intégration permettant d'effectuer toutes les étapes nécessaires à la commande d'un procédé. Cela inclut la conception du pilote de la carte d'acquisition, la simulation par blocs, la conception d'un système de commande. Le logiciel est basé sur les quatre vecteurs suivants d'intégration : présentation, données, procédé et contrôle. La programmation peut être réalisée à l'aide d'outils graphiques ou de manière dynamique avec des sous-programmes conventionnels en Fortran, C ou Pascal. Le logiciel peut aussi bien fonctionner en simulation que en ligne sur un procédé réel. Dans le cas de l'acquisition de données, le logiciel est utilisé sur un site industriel, au Portugal. En ce qui concerne la commande, un chercheur du groupe a utilisé ce cadre logiciel pour développer en simulation une commande multivariable non linéaire d'un réacteur de polymérisation incluant un observateur d'états et un estimateur de paramètres. En conclusion, il s'agit d'un outil perfectible, mais puissant, moderne, intégré, bien adapté à la commande de procédés.

Abstract

ACRUN - Contribution of Object Oriented Technology to Computer Integration of Acquisition and Control

Object oriented programming allows to integrate tools having different fonctionnalities in a unique environment. ACRUN project concerned the production of a code having a high degree of integration and allowing to realize all the steps necessary in a given process control design . This includes the creation of the data acquisition card driver, the simulation based on the block concept, the design of the control system. The code is based on the following four vectors on integration : data, process and control. Programming can be done either by graphical tools, or dynamically by conventional subroutines written in Fortran, C or Pascal. The program can work as well in simulation as on a real process. In the case of data acquisition, it is now used on an industrial site in Portugal. Concerning process control, a different searcher of the group has used this programming framework to develop in simulation the multivariable non linear control of a styrene polymerization reactor, including a state observer and a parameter estimator. In conclusion, it is a perfectible tool, powerfurl, modern, fully integrated and well adapted to process control.