Intégration de fonctionnalités de prototypage dans ATLAS, un atelier de spécification, réalisation et validation d'applications de supervision embarquées

Bertrand Vilain, Thierry Poulain, Christophe Kolski

Publié dans : Automatique-Productique-Informatique Industrielle (APII), volume 28, n. 6, pp. 645-685, 1994.

Intégration de fonctionnalités de prototypage dans ATLAS, un atelier de spécification, réalisation et validation d'applications de supervision embarquées

Bertrand Vilain, Thierry Poulain, Christophe Kolski

Laboratoire d'Automatique et de Mécanique Industrielle et Humaine (L.A.M.I.H.)
URA CNRS 1775, B.P. 311, Le Mont Houy, 59304 Valenciennes Cedex, France

---

RESUME. Cet article propose une démarche itérative de développement d'applications de supervision, basée sur des méthodes du génie logiciel. Cette démarche est supportée par un atelier logiciel, appelé ATLAS. Celui-ci est destiné à la spécification, la réalisation et la validation de synoptiques industriels susceptibles d'être embarqués. ATLAS vise à faciliter le dialogue entre les différents intervenants du cycle de développement en offrant des fonctions de prototypage.

ABSTRACT. This paper presents an iterative method for supervision application development. Based on software engineering concepts, it is supported by ATLAS, a software engineering workbench for specification, development and validation of industrial control displays. These displays can be embarked. ATLAS aims at making easier the dialogue in the development team and provides prototyping functions.

MOTS CLES : Interface Homme-Machine, Spécification, Prototypage, Réalisation, Validation, Atelier Logiciel, Interface Embarquée, ATLAS, application de supervision.

KEY WORDS : Man-Machine Interface, Specification, prototyping, Development, Validation, Software Engineering Workbench, Embedded Interface, ATLAS, supervision application.

I. Introduction

L'automatisation progressive des procédés industriels entraîne des besoins croissants en logiciels et matériels de supervision capables de contrôler les processus de production, tout en pouvant être intégrés à des systèmes de gestion englobant la totalité des opérations d'une usine. Devant cette automatisation et l'introduction de l'informatique, d'importants changements dans le contenu et les méthodes de travail sont apparus. Selon Sperandio (1988), la place de l'homme dans le système, son rôle et ses qualifications professionnelles exigées ainsi que son degré d'autonomie s'en trouvent modifiés de façon majeure. En contrôle de procédé industriel automatisé, les nouvelles activités essentiellement mentales confiées aux opérateurs humains reflètent cette tendance : l'opérateur humain se voit confier, au travers de ces systèmes de supervision la gestion des dysfonctionnements et le suivi de son installation. En effet, l'ensemble des dysfonctionnements n'est pas modélisable de façon exhaustive et leur gestion se base souvent sur des techniques issues d'un savoir-faire typiquement humain (Sheridan, 1988, Millot, 1988 ; De Keyser et al., 1992 ; Johannsen, 1992).

L'homme se voit donc souvent confier le rôle de garant ultime de la sécurité du procédé lorsque celui-ci est déficient. Cependant, dans la salle de contrôle, il est la plupart du temps éloigné de la réalité du terrain et des installations, et travaille essentiellement à travers une interface graphique lui donnant une représentation plus ou moins réaliste du procédé à l'aide de modes de représentation spécifiques (des synoptiques, des courbes, des barre-graphes, des graphes de fluence, etc), selon différents niveaux d'abstraction (Rasmussen, 1986). L'opérateur humain prend alors ses décisions et effectue des actions sur le procédé tout en étant plongé dans un monde virtuel dans lequel l'ergonomie de conception et d'utilisation des outils graphiques mis à sa disposition est d'une importance fondamentale (Poulain et Kolski, 1992 ; Kolski, 1993).

La tendance actuelle des industriels est de se doter de logiciels de supervision du commerce. On en recense environ une centaine actuellement (Pradenc, 1992). L'époque de développement spécifique *complet* réalisé en interne par l'entreprise est révolue, l'industriel préférant maintenant configurer et programmer à partir d'une architecture définie par le constructeur son application de supervision. Dans cette optique, un des soucis des industriels est de produire des applications de supervision de qualité et à moindre coût; cet aspect financier n'apparaît pas seulement durant les phases de développement mais aussi lors des phases d'exploitation du produit développé. C'est pourquoi la pratique du Génie Logiciel est une alternative indispensable pour remplir ces objectifs de coûts. Cependant, il s'avère que, dans les procédés complexes, les logiciels permettant de réaliser des applications de supervision sont rapidement inefficaces et source de problèmes ergonomiques et/ou techniques, s'ils ne s'accompagnent pas d'une méthodologie concrète facilitant à l'équipe de développement un travail coopératif intégrant une démarche de prototypage (Bodker et Gronbaek, 1991 ; Poulain et al., 1993).

Dans ce contexte, cet article a pour objectif de décrire l'intégration de fonctionnalités de prototypage dans un atelier logiciel, appelé ATLAS, destiné à la spécification, la réalisation et la validation de synoptiques industriels[1]. Ces synoptiques sont susceptibles d'être embarqués. Cet atelier a été développé conjointement par la CSEE[2], 3IP [3] et le L.A.M.I.H. dans le cadre d'un projet ANVAR, le L.A.M.I.H. ayant été chargé de l'étude, du développement et de l'intégration des fonctionnalités de prototypage dans l'atelier.

Cet article se compose de trois parties. Dans la première, nous décrivons le principe des applications de supervision visées. Dans la seconde, nous proposons une démarche de développement d'applications de supervision utilisant le prototypage. Enfin, dans la dernière partie, nous expliquons l'application de la démarche à l'aide de l'atelier ATLAS en nous focalisant sur les fonctionnalités de prototypage.

II. Description des applications visées

Dans les applications de supervision, l'opérateur humain est le plus souvent éloigné de la réalité du terrain et des installations. Il travaille essentiellement en salle de contrôle à travers une interface graphique homme-machine lui fournissant une vision plus ou moins réaliste du procédé. Cette interface comprend des vues dites "synoptiques" accessibles sur un ou plusieurs écrans de visualisation, qui s'insèrent dans des applications de supervision.

Il s'agit dans un premier temps de préciser ce que nous entendons par "vues synoptiques". Il sera ensuite plus facile de décrire une application de supervision, ainsi que la manière dont est modélisé un composant d'une vue pour les applications visées.

II.1. Les vues synoptiques

L'interface graphique homme-machine en salle de contrôle est composée d'ensemble de vues, appelées de façon générale "synoptique". Ce terme de synoptique n'est pas utilisé dans son sens restrictif, mais il représente un ensemble de vues dites "synoptiques" constituées de plusieurs vues graphiques (Taborin, 1989) (voir figure 1).

---

1 Cet atelier se situe dans l'esprit des projets PTA (Poste de Travail de l'Automaticien) et BASE-PTA (Base d'Application et Standard d'Echange pour le Poste de Travail de l'Automaticien) qui ont débouché sur le projet de norme AFNOR Pr Z 68-901 en janvier 1992. ATLAS ne se calque pas sur le modèle conceptuel BASE-PTA proposé par la norme, mais vise à intégrer dans un atelier unique un ensemble d'outils adaptés aux besoins de l'équipe de développement, permettant de déboucher sur la réalisation de vues synoptiques susceptibles d'être embarquées.

2 CSEE, ZA Courtaboeuf, BP80, 91943 Les Ulis Cédex

3 Société pour l'Innovation, l'Informatique Industrielle et la Productique (3IP), 104, rue Castagnary, 75015, Paris.

En général, ces vues sont construites à partir de symboles graphiques interconnectés ou non, correspondant par exemple à des vannes, des cuves, des conduits, des moteurs, etc. Ces symboles représentent une partie de l'installation réelle, et leur dynamique permet de rendre compte des changements d'états de celle-ci. On a souvent aussi recours à des fonctions graphiques permettant d'afficher des valeurs numériques, des messages, des barre-graphes, des courbes, etc, afin de faciliter les prises de décision de l'opérateur et lui permettre d'effectuer des actions de correction, de configuration, d'optimisation ou de reprise suite à un dysfonctionnement. La figure 2 donne un exemple de vue graphique, valable pour les applications visées, où l'animation revient à simuler graphiquement la propagation de matière ou d'énergie à partir d'un ensemble de données provenant de capteurs.

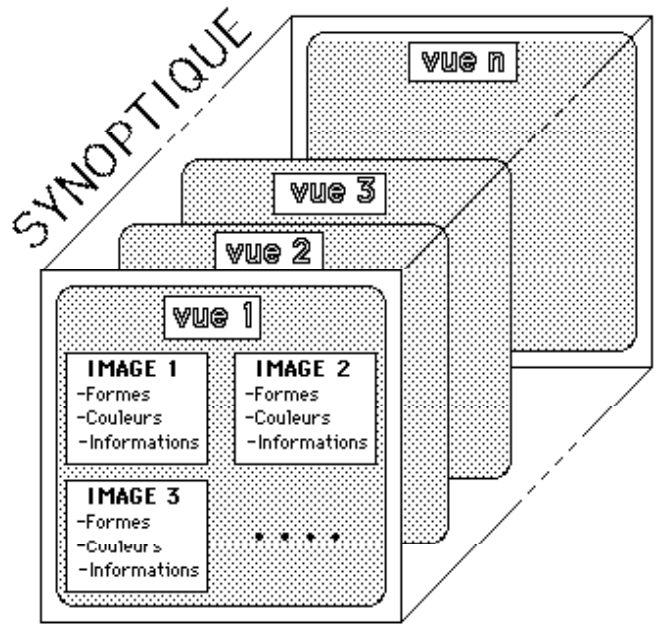

Figure 1 : Définition d'un synoptique (Taborin, 1989)

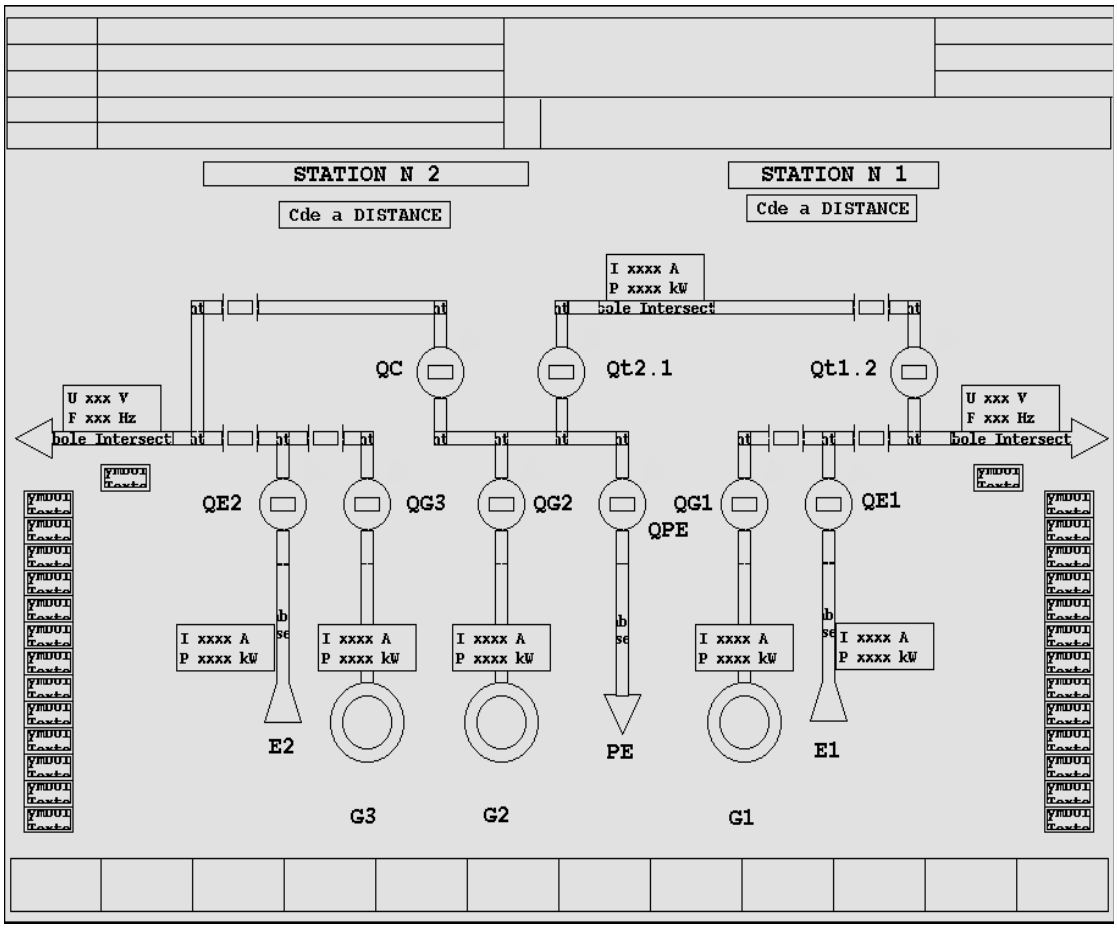

Figure 2 : Exemple de vue de supervision pour les applications visées

Une vue est composée de deux parties distinctes :

- *La partie statique* : appelée encore fond de plan, elle est constituée d'informations non modifiables, restant en permanence à l'écran. Son but est d'aider l'opérateur dans ses tâches d'identification et d'interprétation des informations dynamiques constituant la vue considérée. Par exemple, le repère d'une courbe appartient à la partie statique d'une vue.

- *La partie dynamique* : appelée aussi partie animée, elle est consituée de fonctions dites d'animation. Les informations dynamiques permettent de suivre les évolutions quantitatives et qualitatives des variables du procédé. Ainsi, elle peuvent exprimer des variations et des tendances au cours du temps. Une courbe, un barre-graphe ou un symbole capable de changer de forme, de couleur et/ou de position appartiennent à la partie dynamique d'une vue graphique.

Après avoir défini la notion de synoptique, la structure et l'architecture des applications de supervision visées peuvent être maintenant présentés.

II.2. Principe et architecture d'une application de supervision

La figure 3 résume l'architecture logicielle d'une application de supervision telle qu'elle a été

spécifiée en début de projet. Elle a servi de base au développement de l'atelier logiciel décrit plus loin. On y voit apparaître quatre grandes fonctions logicielles permettant à l'opérateur de superviser son procédé :

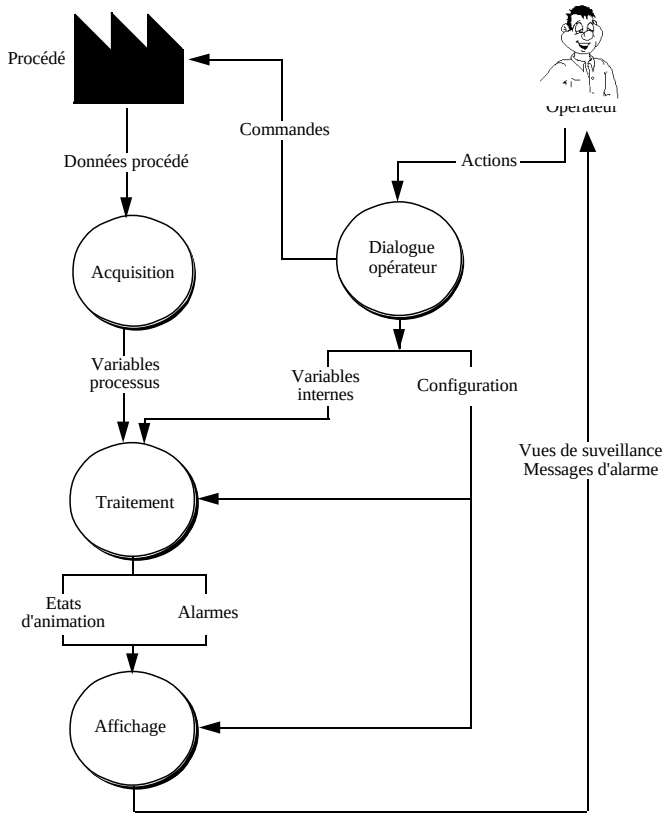

Figure 3 : Fonctions logicielles d'une application de supervision

- *Acquisition* : cette fonction a pour but de recueillir et de filtrer les informations provenant du procédé.

- *Traitement* : c'est dans cette fonction que réside la logique permettant, d'une part, de passer d'un ensemble de données provenant du procédé à des états dits d'animation permettant d'animer le synoptique, et, d'autre part, d'engendrer les alarmes. Ce traitement s'effectue en trois étapes : la première consiste à évaluer chaque constituant instrumenté afin d'en déterminer ses états logiques. A partir du réseau des constituants, la seconde étape a pour objectif de propager l'ensemble des états logiques, et ceci pour des composants instrumentés ou non. Durant la troisième étape, on détecte les incohérences pouvant engendrer des alarmes. Prenons l'exemple de la figure 4 pour illustrer cette fonction. Cette figure montre un sous-ensemble d'un réseau dans lequel une pompe alimente en eau une installation par l'intermédiaire d'une vanne motorisée :

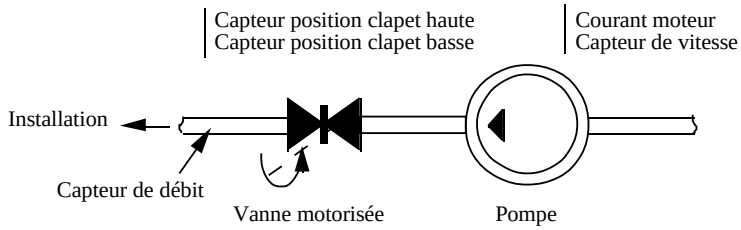

Figure 4 : Sous-ensemble d'un réseau

Lors de la première étape, on détermine les *états évalués* suivants :

- *A partir des informations provenant des capteurs de position du clapet de la vanne, position haute et position basse, on peut en déduire que la vanne est ouverte.*

- *Un capteur de vitesse sur la pompe indique qu'elle est en marche.*

La propagation permet de fixer les *états propagés* à partir des états évalués :

- *Par exemple, sachant que la pompe est reliée à la vanne, que la vanne est ouverte et que la pompe est en marche, un état propagé correspondant au fait que la vanne est en débit peut alors être déduit.*

La troisième étape a pour objectif de détecter les incohérences possibles :

- *par exemple, si le capteur de débit ne signale pas de fluide alors que la vanne est en débit, il y a une incohérence pouvant donner lieu à l'apparition d'une alarme.*

- *Affichage* : cette troisième fonction traduit graphiquement le résultat des fonctions précédentes et cela en modifiant l'aspect graphique des symboles présents à l'écran. En reprenant l'exemple précédent, l'attribut graphique associé à la couleur de la pompe peut être modifié pour visualiser le fait que celle-ci est en marche.

- *Dialogue "opérateur"* : cette dernière fonction gère les actions provenant de l'opérateur de supervision. Celles-ci peuvent correspondre à des commandes de changement de vues, mais aussi à des actions directes sur le procédé, comme la modification d'une consigne.

En conséquence, dans les applications visées, chaque composant d'une vue peut être modélisé des points de vue logique et graphique selon le principe décrit maintenant.

II.3. Modélisation des composants

Dans une vue, lorsque l'état d'une variable représentant un composant change, certains paramètres dynamiques de l'image associée doivent être modifiés. Le comportement dynamique du composant va ainsi montrer son changement d'état, par exemple en changeant de forme ou de couleur.

Les vues graphiques étant le reflet de l'installation réelle, le symbolisme de chaque constituant de l'application à superviser doit représenter de manière plus ou moins réaliste le composant réel. Pour homogénéiser les images des composants, il est fondamental de standardiser leur graphisme et leur comportement afin que l'opérateur humain qui doit superviser l'installation ne soit pas dérouté par les représentations d'une pompe, d'une vanne ou d'un moteur.

Ainsi, un composant affiché sur écran peut être créé par instanciation d'un composant type existant dans une bibliothèque. Ce composant sera associé à un type d'aspect logique -ou sémantique- et un type d'aspect graphique (voir figure 5) :

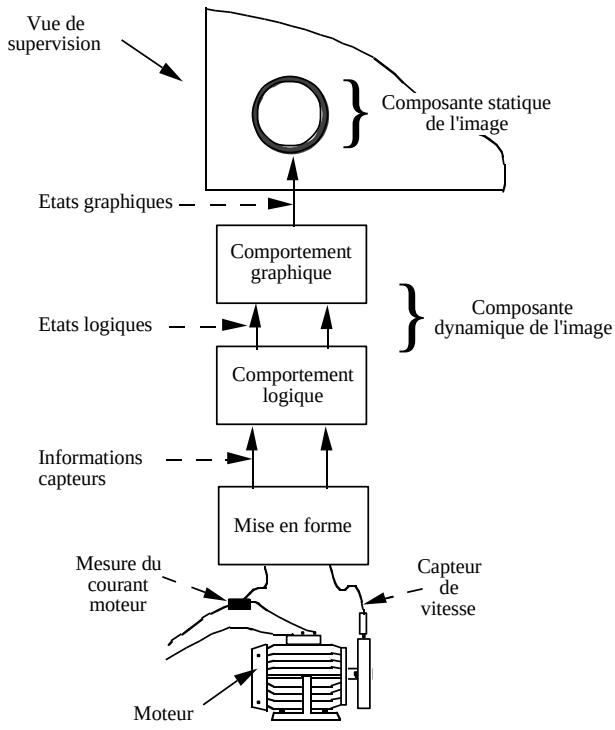

Figure 5 : Structure d'un composant

- *L'aspect logique* décrit le comportement logique du composant, dont l'état dépend d'un ensemble de variables du procédé. Par exemple, il est possible de caractériser une vanne motorisée appelée "vanne motorisée-14" par les informations suivantes reliées à la sémantique du composant : des capteurs de position haut et bas, et un capteur de débit. La règle *"si le capteur haut est vrai et le capteur de débit est vrai alors la vanne est en débit"* contribue à la modélisation logique du comportement de la vanne considérée. A partir de la sémantique de ce composant et de la modélisation complète du réseau, il est possible à partir de l'algorithme de propagation de déterminer l'ensemble des informations dans le réseau et donc de connaître le comportement dynamique de l'installation complète.

- *L'aspect graphique* décrit la manière dont le composant se comporte sur l'écran en fonction des différents états logiques. Par exemple, les variables "capteur de débit" et "capteur haut" sont reliées au comportement logique du composant "vanne motorisée-14". Grâce à ces deux variables, une variable interne appelée "défaut" a été définie dans l'aspect logique comme suit : *"si le capteur haut est vrai et le capteur de débit est inférieur à un dixième de bar alors la variable défaut est vraie"*. Cet état logique "défaut" peut conditionner le graphisme du composant affiché sur la vue par une règle du type : *"si la vanne est en défaut, alors sa couleur de fond est rouge"*. Pour gérer les autres états graphiques possibles, un ensemble de règles doit être défini.

Pour les applications de supervision telles que nous les avons décrites, nous avons mis au point une démarche de développement faisant l'objet de la partie suivante.

III. Démarche de développement de synoptiques industriels

La spécification, la réalisation et l'évaluation des systèmes homme-machine font l'objet actuellement de nombreux travaux de recherche, conduisant à des méthodes, des techniques, des outils et des modèles opérationnels ou en cours de validation. La littérature technique offre dans ce domaine de nombreuses synthèses et classifications. Par exemple, Daniellou (1986) et Millot (1990) décrivent des méthodes globales de conception de système homme-machine en salle de contrôle. Hancock et Chignell (1989) et Kolski et al. (1992) proposent des méthodes de conception d'interfaces "intelligentes". Taborin (1989) et Kolski (1993) recensent les différents types d'outils d'assistance aux opérateurs dans les salles de contrôle de procédé industriels.

En se focalisant sur les applications de supervision de type synoptiques industriels, on constate que leur développement s'appuie sur une grosse composante logicielle. Il convient donc de s'intéresser à l'apport du génie logiciel à ce domaine, avant de proposer une démarche de développement de synoptiques industriels.

III.1. Apport du génie logiciel

Une application de supervision est dotée d'un ensemble de fonctionnalités concrétisées à l'aide de programmes informatiques. Afin d'optimiser les facteurs coûts et productivité lors de leur développement, le génie logiciel offre des perspectives prometteuses (Poulain, 1994). A ce sujet, différents modèles représentant le cycle de vie d'un système sont préconisés en génie logiciel (voir à ce sujet Sommerville, 1988, Partridge, 1990, ou Jaulent, 1990). Nous nous attardons sur le modèle spirale (Boehm et al., 1984), qui contrairement au modèle cascade (Waterfall) (Boehm, 1982) et au modèle en "V" proposé par l'Association Française pour le Contrôle Industriel de la Qualité (voir Jaulent, 1990), facilite une démarche itérative de conception. En effet, pour parvenir à une application de supervision adaptée aux besoins informationnels des opérateurs humains, il est souvent nécessaire de proposer et évaluer plusieurs versions des vues en simulation ou sur site réel avant la validation des vues synoptiques.

Le modèle spirale ne suppose pas la définition *complète* des besoins dès le début du projet, tout en étant basé sur *l'analyse du risque*. A ce sujet, Scapin et al. (1988) montrent que certaines décisions de conception sont le fruit de compromis entre plusieurs critères. On peut donc évaluer l'impact des différents choix de conception avec les experts de l'application et les utilisateurs du système et ainsi analyser le risque qu'ils amènent.

Le modèle de la figure 6 reprend les points suivants : objectifs, contraintes, alternatives, risques, méthodes de résolution du risque, résultats de la résolution du risque, plan pour la phase suivante, engagement (humain, matériel, logiciel, financier…). Le modèle Spirale comporte une composante radiale représentant les coûts cumulés pour mener à bien les différentes étapes ainsi qu'une composante angulaire indiquant les progrès réalisés à la fin de chaque étape du projet. Il décrit le développement comme un processus itératif selon quatre phases (une par cadran) :

- *Détermination des objectifs, des alternatives et des contraintes* : chaque cycle de la spirale commence par l'identification (i) des objectifs que doit remplir le sous-ensemble du produit à réaliser, que ce soit sous les aspects fonctionnels, de performances, liés aux fonctionnalités, à la facilité à s'adapter ou à changer, etc, (ii) des différents moyens possibles pour réaliser le sous-ensemble du produit, comme l'achat de logiciels existants, la réalisation, la réutilisation, etc, et (iii) les différentes contraintes imposées par l'application des alternatives : coûts supplémentaires, organisation à mettre en place, matériels à prévoir, etc.

- *Evaluation des solutions* : cette étape consiste à évaluer les alternatives relatives aux objectifs et aux contraintes. Pour mener à bien cette étape, on peut réaliser des prototypes, des simulations ou encore effectuer des essais et utiliser toute autre méthode permettant d'évaluer le risque engendré par chaque alternative.

- *Développement et vérification du produit* : Suite à l'étape précédente, deux cas de figure apparaissent : soit il reste des risques et dans ce cas il faut alors poursuivre la spirale, ce qui fait

l'objet de la phase suivante, soit on aboutit à un prototype opérationnel qui servira de base aux étapes traditionnelles du modèle cascade qui sont : (1) la conception détaillée, (2) le codage du logiciel, (3) les tests unitaires, (4) l'intégration et les tests du logiciel, (5) les tests de recette et enfin (6) l'installation du logiciel.

- *Planification des phases suivantes* : cette étape consiste à fixer les différents moyens à mettre en œuvre pour poursuivre la spirale, tels que les ressources humaines ou matériels.

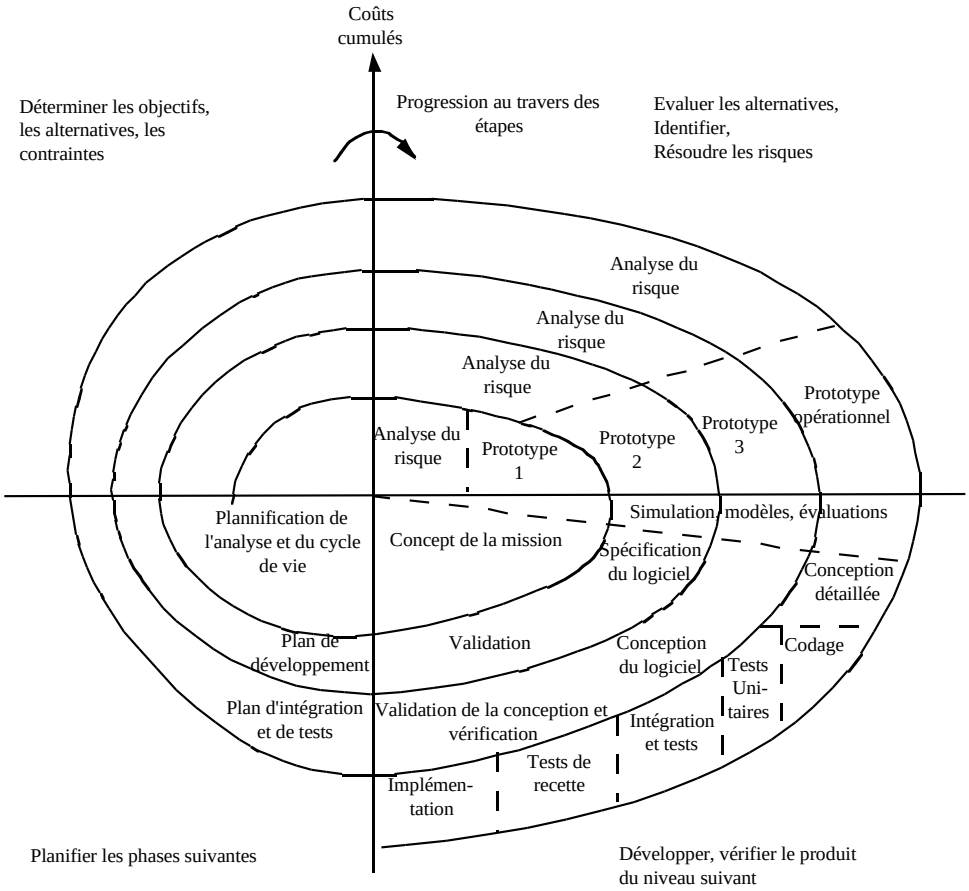

Figure 6 : Modèle spirale (Boehm et al., 1984)

Le modèle spirale est adapté au développement d'applications de supervision, car il permet d'évaluer tout au long des étapes les différents choix de conception. Dans un domaine où l'objectif final est de fournir une interface graphique appropriée, l'utilisation du prototypage de vues synoptiques comme moyen d'analyser le risque, peut faciliter en plus le dialogue entre les différents intervenants du processus de développement. Pour ces raisons, nous avons adapté la spirale de Boehm sous la forme d'une démarche de développement d'applications de supervision faisant l'objet de la partie suivante.

III.2. Démarche proposée

La démarche proposée vise à permettre aux développeurs de formaliser progressivement leurs spécifications et d'évaluer l'impact de leurs choix de conception. Ainsi, elle doit faciliter la stabilisation des différents problèmes tout au long de la conception, selon les principes introduits par Boehm dans le modèle spirale. Prévue pour faciliter la spécification, la réalisation et la validation des synoptiques industriels, elle comprend deux phases (figure 7) : (1) une première phase dite "descendante" qui, à partir des connaissances du procédé et des besoins informationnels des opérateurs, aboutit au dossier de définition de l'application de supervision, et (2) une phase dite "ascendante" qui à partir de ce dossier consiste à produire le logiciel qui sera implanté sur le site (Poulain, Germe et Kolski, 1993 ; Poulain, 1994). Ces deux phases seront expliquées ci-après. L'atelier logiciel ATLAS, auquel nous avons contribué et qui se base sur cette démarche, sera ensuite décrit.

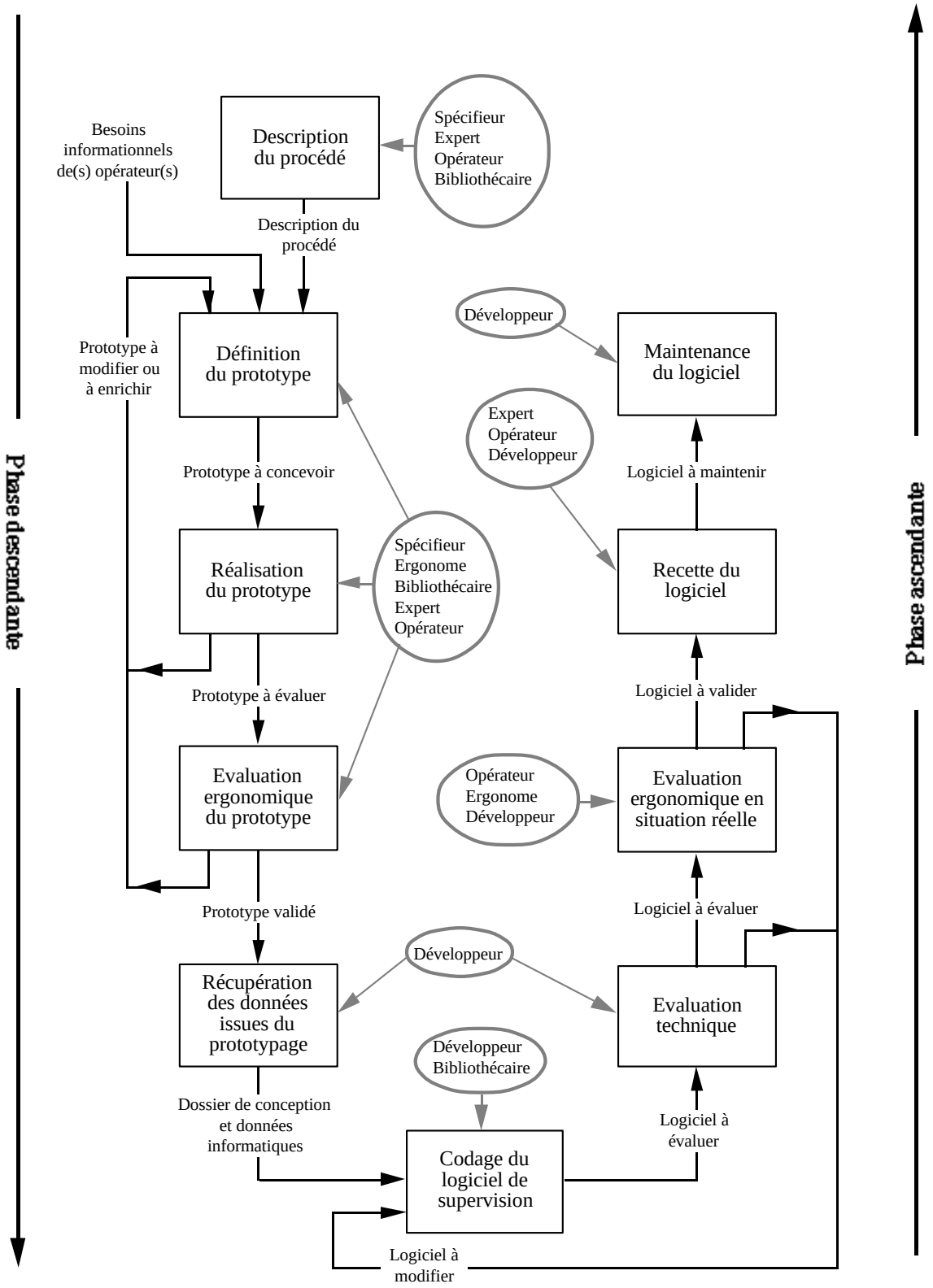

Figure 7 : Démarche de développement de synoptiques de supervision

Comme nous l'avons déjà vu, la conception et l'évaluation d'une application de supervision fait intervenir des personnes ayant des connaissances et des compétences différentes mais néanmoins

complémentaires. Dans ce but, il convient d'identifier dans un premier temps les intervenants impliqués dans la démarche proposée, afin de mieux les situer lors de la description des étapes de la démarche.

Ces rôles respectifs de ces personnes sont au nombre de six : (i) le spécifieur, (ii) l'expert, (iii) le bibliothécaire, (iv) l'ergonome, (v) l'opérateur et (vi) le développeur (figure 8). Le **spécifieur**, comme son nom l'indique a pour fonction de spécifier l'application qui sera réalisée. Pour ce faire il fait intervenir l'**expert** de l'installation pour laquelle on désire concevoir une application de supervision. L'expert a une connaissance profonde de la structure de l'installation ainsi que de son fonctionnement. Le rôle de **bibliothécaire** a également été introduite dans la démarche : son rôle est de gérer une bibliothèque de composants de base à partir de laquelle d'une part le spécifieur peut décrire l'installation et d'autre part le développeur peut concevoir et réaliser l'application de supervision. En effet le **développeur** a pour fonction de produire le logiciel de supervision qui sera effectivement implanté sur le site. Le développement fait également intervenir d'une part l'**opérateur**, c'est à dire la personne qui va utiliser l'application de supervision pour contrôler et commander l'installation qu'il a en charge, et d'autre part l'**ergonome** qui garantit la prise en compte des facteurs humains à certaines étapes de la démarche.

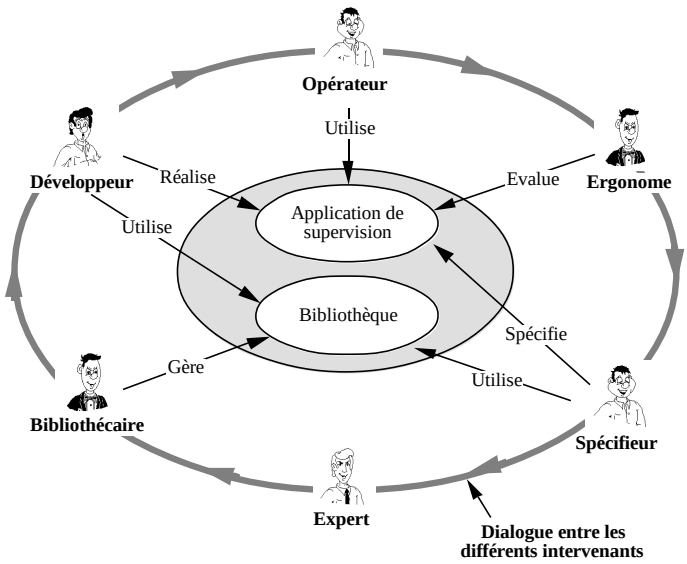

Figure 8 : Les différents intervenants de la démarche proposée (Poulain, 1994)

Après cette brève présentation des acteurs, les deux phases principales de la démarche peuvent être maintenant expliquées.

III.2.1. Phase descendante

La *phase descendante consiste* à élaborer et valider un prototype du synoptique à réaliser, à partir de la connaissance du procédé et du recensement des besoins informationnels des opérateurs dans la salle de contrôle. La fin de cette phase descendante aboutit à un dossier de spécification du logiciel de supervision. Cette phase comprend plusieurs étapes qui sont : (i) la description du procédé, (ii) la définition du

prototype, (iii) la réalisation du prototype, (iv) l'évaluation ergonomique du prototype, et (v) la récupération des données issues du prototypage. Celles-ci sont successivement décrites.

III.2.1.1. Description du procédé

La description du procédé revient à décrire et à modéliser le procédé à interfacer sous un angle fonctionnel (par exemple la mission du procédé) et sous un aspect structurel (les composants, les entrées, les sorties, etc.), et ceci selon différents niveaux d'abstraction. Cette étude permet au **spécifieur** de mieux appréhender d'une part le procédé à interfacer, mais aussi d'extraire la connaissance des **experts** du procédé ainsi que celle des **opérateurs** de l'installation.

Notre proposition réside dans l'utilisation dans cette étape d'un outil informatique permettant au spécifieur de "dessiner la structure de l'installation". Cette description doit pouvoir être exploitée par la suite pour construire les vues. Cet aspect sera abordé par la suite lors de la description d'un tel outil dit de prototypage. Il est à noter que lors de cette étape le spécifieur, dans la mesure où il assemble des composants, doit disposer d'une *bibliothèque* prédéfinie, créée et gérée par le **bibliothécaire**. La bibliothèque de composants peut reposer sur le principe de modélisation abordé précédemment, à savoir qu'un composant est caractérisé par un comportement logique et un comportement graphique. Il est à noter également que, dans cette première étape, il n'est pas indispensable de disposer d'un comportement logique tenant compte de l'instrumentation exacte du composant. Seul un comportement logique dit *simplifié* suffit pour décrire l'installation, et ceci en vue d'une première évaluation.

III.2.1.2. Définition du prototype

La définition du prototype consiste pour le **spécifieur** à définir l'ensemble des vues formant l'application de supervision, ainsi que leur dynamique et leur enchaînement. Ceci requiert, d'une part la connaissance du procédé à interfacer issue de l'étape précédente, et d'autre part une étude conduisant à recenser les besoins informationnels de l'opérateur. Ce recensement nécessite une analyse par **l'ergonome** de la tâche et de l'activité réelle ou future probable des opérateurs.

Dans le cas de la supervision des procédés, et en raison des contraintes économiques et sécuritaires souvent mises en jeu, il faut absolument spécifier l'ensemble des fonctionnalités des synoptiques. Il est donc indispensable de définir, non pas une maquette des vues utilisées par le logiciel de supervision, mais un prototype. Rappelons que la différence fondamentale entre une maquette et un prototype réside dans le fait qu'une maquette ne permet de tester qu'une partie des fonctionnalités, ce qui n'est pas le cas d'un prototype.

Dans notre cas, aucune fonctionnalité de l'outil informatique que nous proposons par la suite ne peut remplir cet aspect d'analyse conduisant à la définition des vues de supervision[4]. Lors de cette étape, le

4 Cet aspect donne lieu actuellement a de nombreuses recherches (Cf. par exemple Lind, 1994 ; Vilain et al, 1994) et n'a pu être considéré dans notre application.

spécifieur peut être assisté par l'ergonome qui assure la prise en compte des facteurs humains lors de la définition des vues. Cette prise en compte a pour objectif d'une part d'éviter des erreurs de présentation de l'information et d'autre part de faciliter, par le biais d'une analyse des tâches des opérateurs, la structuration des différentes vues en regard des objectifs de conduite assignés aux opérateurs.

En fonction des différentes données, dont il dispose en entrée de cette étape de définition, à savoir les données décrivant le procédé et les besoins informationnels des opérateurs, le spécifieur définit le nombre de vues ainsi que leur contenu.

III.2.1.3. Réalisation du prototype

Lors de cette étape, le **spécifieur** réalise un prototype des vues synoptiques à évaluer. Le type d'outil informatique mis en œuvre lors de cette étape conditionne l'utilisation ou non de cette démarche de conception. En effet, pour contribuer à une véritable démarche itérative, l'outil doit permettre une modification aisée et rapide du prototype et être facile d'utilisation (Keysons et Parsons, 1990).

Le prototype doit être facilement réalisé en disposant le plus interactivement possible sur l'écran les différents constituants de l'interface, à l'aide de la souris par exemple. De plus, l'outil informatique doit permettre de réutiliser les données graphiques saisies lors de l'étape de description du procédé, à savoir les différents constituants ainsi que leurs interconnexions.

Lors de cette étape le spécifieur peut être amené, en dialoguant avec les **experts**, les **opérateurs** et les **ergonomes,** à reboucler sur l'étape de définition du prototype de manière à modifier ou compléter ses spécifications (figure 7). La composante dynamique des vues peut ne pas être systématiquement prise en compte. En effet, seule une pré-évaluation statique des vues peut être effectuée. De ce fait, pour parfaire la validation, l'étape suivante d'évaluation ergonomique du prototype est réalisée.

Mais, afin de pouvoir procéder à l'évaluation dynamique lors de l'étape suivante, il faut au préalable procéder à la génération du code exécutable de l'application de supervision à partir des données de spécification saisies lors de cette étape de réalisation. Cette génération, dans la mesure où l'outil à développer doit contribuer à l'utilisation du prototypage, doit être transparente pour le spécifieur et par conséquent être automatique.

III.2.1.4. Evaluation ergonomique du prototype

Cette étape, effectuée en situation simulée, a pour objectif d'évaluer ergonomiquement avec les **opérateurs** les aspects statiques et dynamiques des vues synoptiques. Cette évaluation doit s'effectuer grâce à un ensemble de situations prédéfinies : la simulation doit alors permettre de raccorder le logiciel de supervision à des variables simulées du procédé dont le comportement peut être facilement défini dans des scénarios pré-établis, comme par exemple des variations pseudo-aléatoires. Ces scénarios peuvent être construits en modifiant pas à pas les variables du procédé dans des fichiers de simulation.

Il est important de souligner que cette simulation ne sera pas effectuée à partir d'une modélisation précise du procédé et cela pour deux raisons principales : (i) une telle modélisation fait intervenir des compétences spécifiques, par exemple la physique qualitative, que ne possèdent souvent pas les différents intervenants et (ii) le temps nécessaire pour réaliser un modèle cohérent, qui dans certains cas se chiffre en plusieurs mois voire plusieurs hommes-année suivant la complexité de l'installation, serait préjudiciable à l'utilisation du prototypage. Ce deuxième point vient en effet en contradiction avec d'une part une des caractéristiques du prototypage qui est d'effectuer une évaluation rapide du logiciel à produire, et d'autre part avec le souci de réduire les coûts de développement. L'inconvénient d'une telle approche et que l'on ne peut valider que l'aspect *contrôle* des vues et non l'aspect *commande*, dans la mesure où les actions des opérateurs ne peuvent être prises en compte étant donné le caractère figé des fichiers de simulation.

La mise en œuvre de ces scénarios fait intervenir les **ergonomes** qui ont pour rôle d'évaluer le prototype, à partir de situations caractéristiques, en collaboration avec les **spécifieurs** et les **opérateurs**. Cette phase d'évaluation peut faire intervenir plusieurs techniques telles que des interviews, des grilles d'évaluation ergonomique ou des questionnaires d'utilisation (Senach, 1990 ; Meinadier, 1991 ; Kolski, 1993). L'évaluation peut conduire à une modification de la définition ou de la conception du prototype (figure 7).

III.2.1.5. Récupération des données issues du prototypage

De façon générale, une vue de supervision est constituée d'une composante statique qui peut être assimilée à un simple "dessin", et d'une composante dynamique contenant l'aspect sémantique permettant d'animer les composants de la vue. Dans la phase de récupération, le **développeur** doit récupérer la composante statique des vues soit sous la forme de document ou encore dans certains cas, si l'outil graphique de prototypage est identique (ou compatible) à l'outil utilisé pour le développement, sous la forme de données informatiques. La composante dynamique n'est généralement pas récupérée complètement, dans la mesure où pour l'élaboration du prototype, on utilise des composants dits "simplifiés" qui ne valident que l'aspect graphique (animation du symbole) et non l'aspect logique, qui, quant à lui, est fonction des instrumentations réelles de l'installation.

Néanmoins, ce point est une motivation supplémentaire pour utiliser une telle démarche basée sur le prototypage. En effet la possibilité de récupérer le maximum de travail issu de la phase de prototypage par le **développeur** réduit les temps de développement, et par conséquent incite à utiliser la notion de prototypage pour spécifier les vues de supervision. De ce fait, l'outil informatique dédié au prototypage doit être compatible avec celui utilisé pour produire la version finale de l'application qui sera implantée sur le site.

III.2.2. Phase ascendante

La phase ascendante permet, à partir des spécifications établies et validées lors de la phase descendante, de concevoir l'application qui sera effectivement implantée sur le site. Cette phase comprend

plusieurs étapes, figure 7 : (i) le codage du logiciel, (ii) l'évaluation technique, (iii) l'évaluation ergonomique en situation réelle, (iv) la recette du logiciel, (v) la maintenance du logiciel. Celles-ci sont successivement décrites.

III.2.2.1. Codage du logiciel

Le codage[5] du logiciel consiste à développer dans sa version finale le synoptique de supervision pour ensuite l'implanter sur le site. De manière générale, le **développeur** peut reproduire les vues s'il ne dispose que de documents papiers ou s'il peut récupèrer les fichiers informatiques correspondants. Puis il modifie le comportement logique des constituants de la vue dans la mesure où, comme nous l'avons vu lors des étapes précédentes, une bibliothèque de composants simplifiés est utilisée. De plus le développeur doit préciser les liens avec l'installation, c'est à dire les moyens d'accès aux données provenant du procédé (adresses, bus de communication).

Pour le codage, une option intéressante réside dans la génération automatique de l'application. En effet, à partir des caractéristiques des différents composants de la vue (équation logique et graphique, données procédé…) et d'un squelette d'application dédié à un type de matériel spécifique, il est possible de produire automatiquement par l'intermédiaire d'un générateur de code l'application finale. Cette option a été retenue dans l'atelier ATLAS qui sera décrit par la suite.

III.2.2.2. Evaluation technique

Avant de placer l'application développée en situation réelle de fonctionnement, il convient d'effectuer une validation technique. Cette étape comme dans la majorité des développements informatiques s'avère indispensable. En effet il paraît inconcevable de ne pas procéder à une étape de test du logiciel, avant de le placer en situation réelle de fonctionnement. Cette validation technique, menée par le **développeur**, consiste à tester les différentes fonctions informatiques, de manière à identifier et corriger les erreurs logiciels, et cela dans le but d'obtenir un produit techniquement fiable. Cette étape étant classique en génie logiciel, elle ne sera pas détaillée ici.

III.2.2.3. Evaluation ergonomique en situation réelle

Une fois le logiciel de supervision validé techniquement, celui-ci est implanté sur le site. Néanmoins il convient de procéder de nouveau à une étape d'évaluation ergonomique en situation réelle, faisant intervenir les **ergonomes**, les **opérateurs** et les **développeurs**. Cette étape se justifie dans la mesure où la première évaluation ergonomique s'est faite en situation simulée et ne permet donc pas de valider complètement le logiciel. Deux raisons principales justifient cet aspect. La première est que, dans une situation simulée, les opérateurs n'ont pas forcément la motivation nécessaire pour effectuer une validation, et leurs actions n'ayant aucune conséquence leur attitude n'est pas forcément conforme à la réalité du

5 En raison de la démarche utilisée, la conception de la version finale est limitée au codage car on récupère la structure informatique "conçue" lors du prototypage.

terrain (absence de stress) (Cf. Van Daele, 1992). La seconde raison est que la première évaluation s'effectue dans un environnement réduit et donc les interactions avec les autres dispositifs présents dans la salle de contrôle, comme les imprimantes, les autres moyens d'affichage, ou les informations provenant d'un rondier n'ont pu être évalués. Comme pour l'évaluation en situation simulée, de nombreuses techniques d'évaluation peuvent être utilisées (Wilson et Corlett, 1990).

III.2.2.4. Recette du logiciel

Cette étape classique du génie logiciel a pour but de vérifier l'adéquation de l'application de supervision aux besoins du client. Cette étape fait donc intervenir ici les **experts,** les **opérateurs** et les **développeurs**. Dans la mesure où la démarche proposée incite les différents acteurs du processus de conception à dialoguer, la recette du logiciel doit se trouver inévitablement facilitée. En effet, dès les premières étapes de la démarche, le spécifieur soumet pour évaluation les spécifications aux futurs exploitants (opérateurs) ainsi qu'aux experts, et de ce fait une détection précoce des problèmes (inadéquation avec les besoins des opérateurs, choix des modes de représentation, etc) a été rendue possible. [6]

III.2.2.5. Maintenance du logiciel.

Cette dernière étape consiste pour le **développeur** à maintenir le logiciel réalisé. A ce sujet, comme le rapportent de nombreux auteurs tel Sommerville (1988), les coûts de maintenance peuvent être jusqu'à deux à trois plus élevés que les coûts de développement dans le cas de systèmes complexes. D'une manière générale, on distingue trois types de maintenance, la maintenance corrective consistant à corriger les erreurs logicielles, la maintenance évolutive permettant de faire face à de nouveaux besoins des utilisateurs et la maintenance adaptative conditionnée par l'utilisation de nouveaux matériels ou logiciels (Jaulent, 1990). L'expérience montre alors que le coût de maintenance est principalement dû à des changements dans la définition des besoins. Notre approche reposant sur le prototypage doit conduire à la réduction des coûts du projet, dans la mesure où elle permet de mieux définir le besoin des opérateurs avant le codage réelle du logiciel et donc permet de réduire la maintenance, aussi bien corrective qu'évolutive.

III.2.3. Conclusion sur la démarche proposée

La démarche proposée repose sur le prototypage pour évaluer les choix de conception dès les premières étapes de développement des applications de supervision selon une démarche itérative. Cette approche présente l'avantage d'établir progressivement les spécifications et surtout de les évaluer au fur et à mesure, facilitant ainsi la détection précoce de problèmes de conception. De plus la démarche contribue à instaurer un dialogue entre les différents intervenants du processus de conception. Par conséquent, une telle approche permet de confronter les opinions et facilite le transfert de connaissance d'un domaine à l'autre.

---

6 Comme nous le verrons par la suite (Cf. IV.1), c'est à la fin de cette étape, dans le cas d'applications embarquées où le code doit ensuite être écrit en mémoire non volatile, que l'on procède au "claquage" de la mémoire.

Néanmoins une telle approche ne peut être applicable que si l'on dispose d'un outil informatique facile d'utilisation pour concevoir le prototype. C'est pourquoi les six premières étapes de la démarche ont fait l'objet d'une informatisation sous la forme d'un atelier logiciel appelé ATLAS, présenté maintenant.

IV. Application de la démarche à l'atelier ATLAS

La démarche présentée précédemment a fait l'objet d'une formalisation et d'une intégration dans un atelier de génie logiciel, appelé ATelier Logiciel d'Animation de Synoptiques (ATLAS), initialement créé pour le compte de la Marine Nationale. Cet atelier est destiné à la **spécification**, la **réalisation** et la **validation** de synoptiques industriels. Ces synoptiques, qui doivent refléter le plus fidèlement possible le procédé, comportent des vues de supervision de type **réseaux** (électriques, hydrauliques) ; ils font partie d'applications de supervision susceptibles d'être embarquées, à bord de navires par exemple comme c'est le cas pour la Marine Nationale. L'une des contraintes initiales de l'atelier était de faire en sorte qu'il soit au maximum utilisable par des personnels non informaticiens.Cet atelier fait partie d'une nouvelle génération de progiciels de supervision visant à faciliter à l'équipe de développement des synoptiques un travail coopératif exploitant au maximum les compétences de chaque intervenant, par l'intermédiaire d'un outil commun permettant une démarche de prototypage.

Dans un premier temps, nous présentons l'environnement logiciel et matériel d'ATLAS. Puis, nous expliquons l'architecture logicielle permettant de développer les applications de supervision. Par la suite, les modules principaux de développement sont expliqués.

IV.1. Environnement logiciel et matériel d'ATLAS

L'atelier ATLAS utilise l'environnement matériel suivant, figure 9 : **une machine de développement**, à savoir une station de travail SUN/SPARC, sert de support pour la spécification de l'application, la réalisation des synoptiques industriels et leur validation, (2) **une machine cible,** encore appelée Unité de Traitement et de Visualisation (UTV), exécute le code résultant de la phase de développement ; celui-ci est téléchargé via le réseau Ethernet de la SUN vers l'UTV. Après validation, ce code correspondant à l'application de supervision réellement embarquée est "claqué" en mémoire morte dans la machine cible. En effet, pour des considérations techniques et sécuritaires, toute application est stockée en mémoire, la machine cible n'étant pas dotée de mémoire de masse.

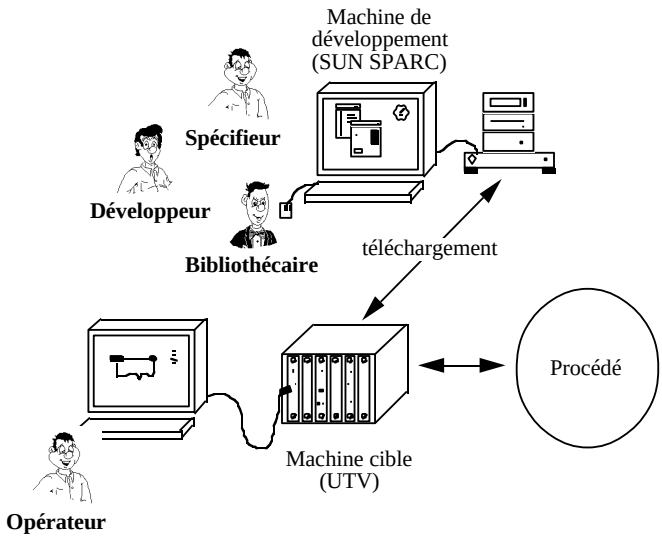

Figure 9 : Structure générale d'ATLAS

Cette configuration a été retenue dans la mesure où les applications destinées à la marine sont implémentées dans du matériel dit "endurci" répondant à des normes de fonctionnement particulières. Dans cette optique, une machine cible spécifique a été réalisée pour répondre aux critères militaires. C'est donc sur la machine cible que seront mis en oeuvre les tâches de l'application de supervision expliquées dans le paragraphe I.2 (figure 3), en l'occurence l'acquisition des données, le traitement logique des données[7], les affichages et le dialogue avec l'opérateur. Elles se concrétisent par des processus communiquant entre eux par des tables de données. Ces différents processus sont construits autour du système d'exploitation VxWorks (dont le fournisseur est WindRiverSystems) qui intègre un noyau temps réel. Les outils logiciels de base suivants ont été utilisés dans l'atelier :

- *TeleUse* : ce générateur d'interfaces sous X-MOTIF, commercialisé par la société Alsys, a permis essentiellement de décrire l'interface homme-machine d'utilisation de l'atelier sur la machine de développement.

- *DATAVIEWS* : cet outil est utilisé pour la description des *synoptiques* sur la machine de développement, et pour leur animation graphique sur la machine de développement et la machine cible. DATAVIEWS, distribué par la société Unitechnic, est constitué d'un éditeur graphique DVdraw, permettant de créer des objets graphiques auxquels on peut associer un comportement dynamique, et d'une bibliothèque de fonctions (DVtools), permettant également la création d'objets mais aussi la gestion des événements et des données influant sur l'aspect graphique des objets (DVtools est interfacée avec le langage C). Concrètement, DVdraw permet au bibliothécaire d'enrichir la bibliothèque graphique des composants, tandis que DVtools a été utilisé d'une part pour concevoir un éditeur dédié pour réaliser les vues de prototypage, et d'autre part

7 Il est à noter qu'en ce qui concerne la fonction de traitement, un algorithme de base a été développé par la société 3IP. Celui-ci assure la détermination des *états évalués*, leur propagation pour fixer l'ensemble des *états propagés* et la détection des *incohérences*. Les différents états évalués sont calculés à partir de règles spécifiques aux composants et à leur instrumentation, appelées encore *types de forme logique*.

pour produire le code destiné à la machine cible. Contrairement à TeleUse, DATAVIEWS n'est pas conçu initialement pour créer des Interfaces de dialogue. [8]

- *ORACLE*, SGBD servant au stockage des données décrivant l'application.

IV.2. Architecture logicielle de la machine de développement

L'atelier ATLAS a été doté d'un ensemble de fonctionnalités permettant de produire une application de supervision. Celles-ci sont accessibles au travers de trois interfaces distinctes, l'une destinée au bibliothécaire, l'une au spécifieur et la dernière au développeur, figure 10. Les informations manipulées par ces intervenants sont respectivement gérées par le gestionnaire de bibliothèque, le gestionnaire de prototypage et le gestionnaire de développement. Toutes les fonctionnalités relatives au prototypage sont en grisé sur la figure.

Le développeur conçoit son application de supervision à partir d'une part d'une bibliothèque de composants gérée et construite par le bibliothécaire, et d'autres part des vues synoptiques réalisées par le spécifieur. Une fois toutes les données spécificiant l'application de supervision saisies par le développeur (Cf. IV.4), le code destiné à la machine cible peut être généré puis téléchargé. Dans le but de tester techniquement l'application de supervision, un simulateur dédié à la machine cible est disponible. Ces différents points sont abordés ci-après au travers des trois gestionnaires, en détaillant particulièrement celui permettant le prototypage.

8 Le lien entre TeleUse et DATAVIEWS s'effectue en intégrant les vues DATAVIEWS dans l'arborescence X-MOTIF.

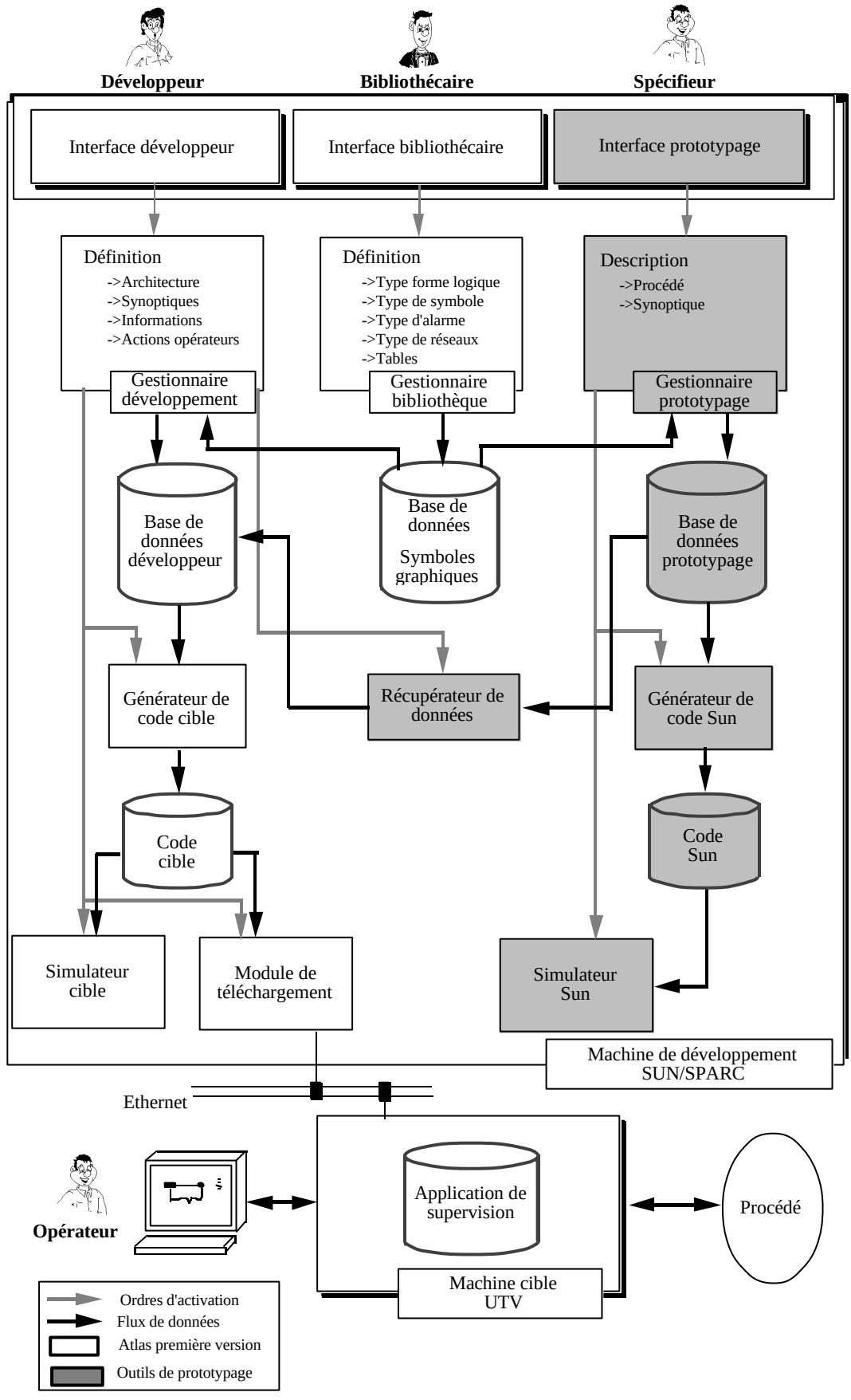

Figure 10 : Atelier ATLAS

IV.3. Le gestionnaire de bibliothèques

L'atelier repose sur la notion de bibliothèque de composants dont le comportement est décrit et testé unitairement. Ainsi, un procédé est décrit à partir d'un ensemble de composants élémentaires reliés entre eux pour former un réseau. Les synoptiques sont construits à partir des différents éléments composant le procédé (vannes, cuves par exemple) et d'éléments complémentaires de visualisation d'informations tels des courbes ou des barre-graphes.

Nous avons vu que le bibliothécaire est l'utilisateur qui a en charge la bibliothèque. Il est le garant du bon fonctionnement de celle-ci et la délivre au spécifieur et au développeur. La bibliothèque est composée de deux classes de données : les données référençables (les tables) et les données instanciables (les types). Ces données sont toutes accessibles facilement par l'intermédiaire de l'interface homme-machine du gestionnaire de bibliothèque (figure 11). Les tables correspondent à un ensemble d'informations élémentaires nécessaires à la construction de la bibliothèque et donc d'une application, selon les principes avancés en II.2 ; celles-ci sont stockées et gérées par le système de gestion de base de données Oracle.

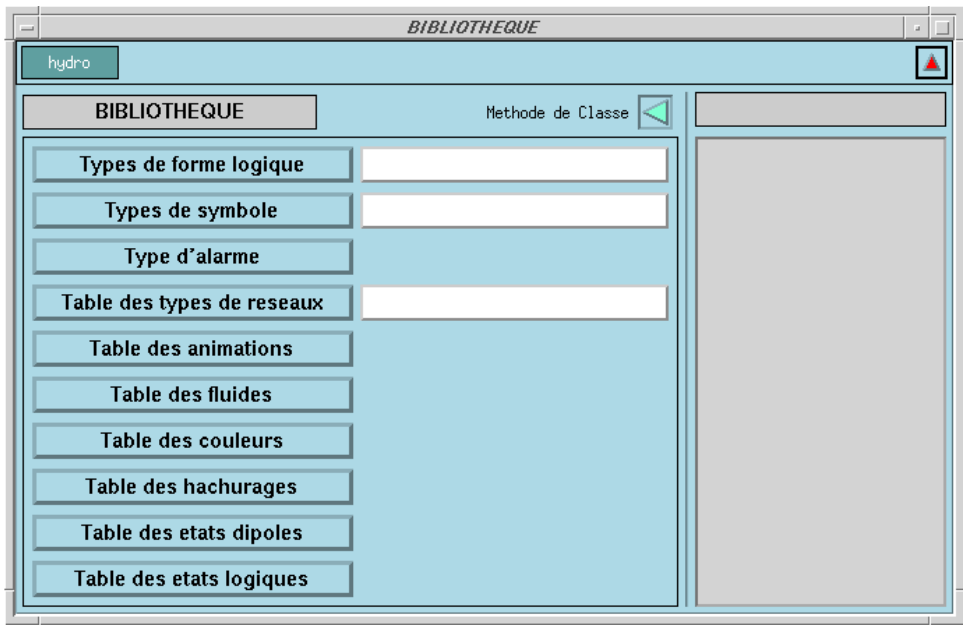

Figure 11 : Interface du gestionnaire de bibliothèque

Les types utilisés pour engendrer les données de l'application sont :

- **Types de formes logiques :** un type de forme logique a pour but de décrire le comportement logique d'une entité élémentaire. Ce comportement se traduit par un ensemble d'informations "procédé" influant sur ce comportement, un ensemble de variables décrivant les états logiques déduits et un ensemble de règles logiques décrivant la façon de déduire les états logiques à partir des valeurs provenant des capteurs du procédé. Le bibliothécaire décrit les équations logiques à partir d'une interface spécifique présentée en figure 12. Les équations logiques sont construites à

l'aide d'un pseudo-langage qui est compilé pour produire un source en langage C. Ces différentes règles sont utilisées par la tâche de traitement présente dans la machine cible.

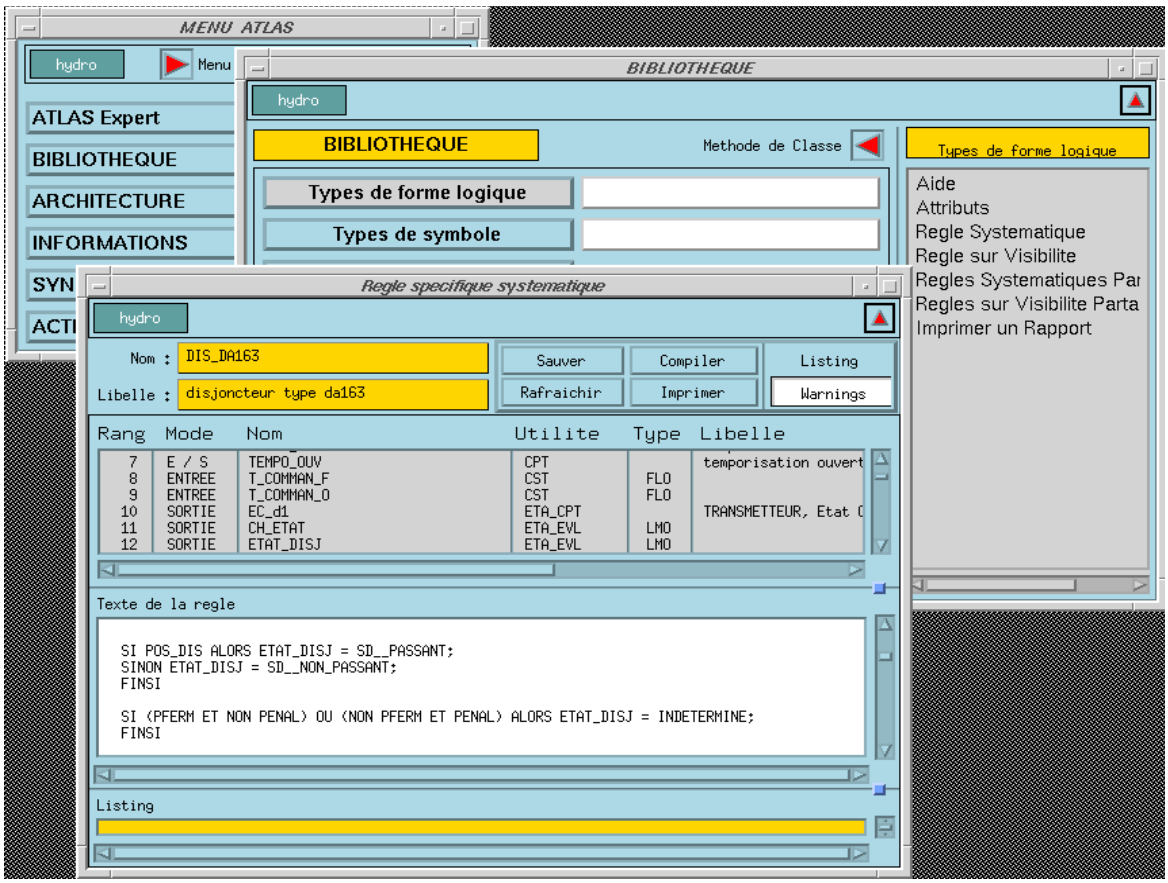

Figure 12 : Editeur des règles logiques

- **Types de symboles :** un type de symbole correspond à un comportement graphique d'une entité élémentaire. A un type de symbole peuvent être attachés plusieurs types de formes logiques. Ce comportement graphique se caractérise par un ensemble de règles de comportement permettant de commander les animations à partir des variables d'entrée (états logiques). De la même manière que pour les règles décrivant le comportement logique, le bibliothécaire utilise un pseudo-langage destiné à produire un source en langage C, figure 13. En parallèle à l'écriture des équations d'animation, le bibliothécaire dispose de l'éditeur DVdraw de Dataviews pour dessiner le symbole graphique et également lui associer des comportements d'animation selon les variables, tels que changer de couleur, apparaître, disparaître, etc, figure 14.

- **Les types d'alarmes :** à chaque composant du procédé, une alarme peut être associée. Celle-ci peut être activée à la suite d'apparitions d'états d'incohérences issus de la phase de propagation. Par exemple, l'évaluation des états logiques du composant "Vanne" indique que celle-ci est fermée; néanmoins, après propagation des différents états logiques des constituants du réseau, on détermine que la vanne est en débit. De plus, un ensemble d'alarmes peut être activé suite à

l'apparition d'un défaut sur un composant, par exemple si le courant absorbé par un moteur est supérieur à la normale.

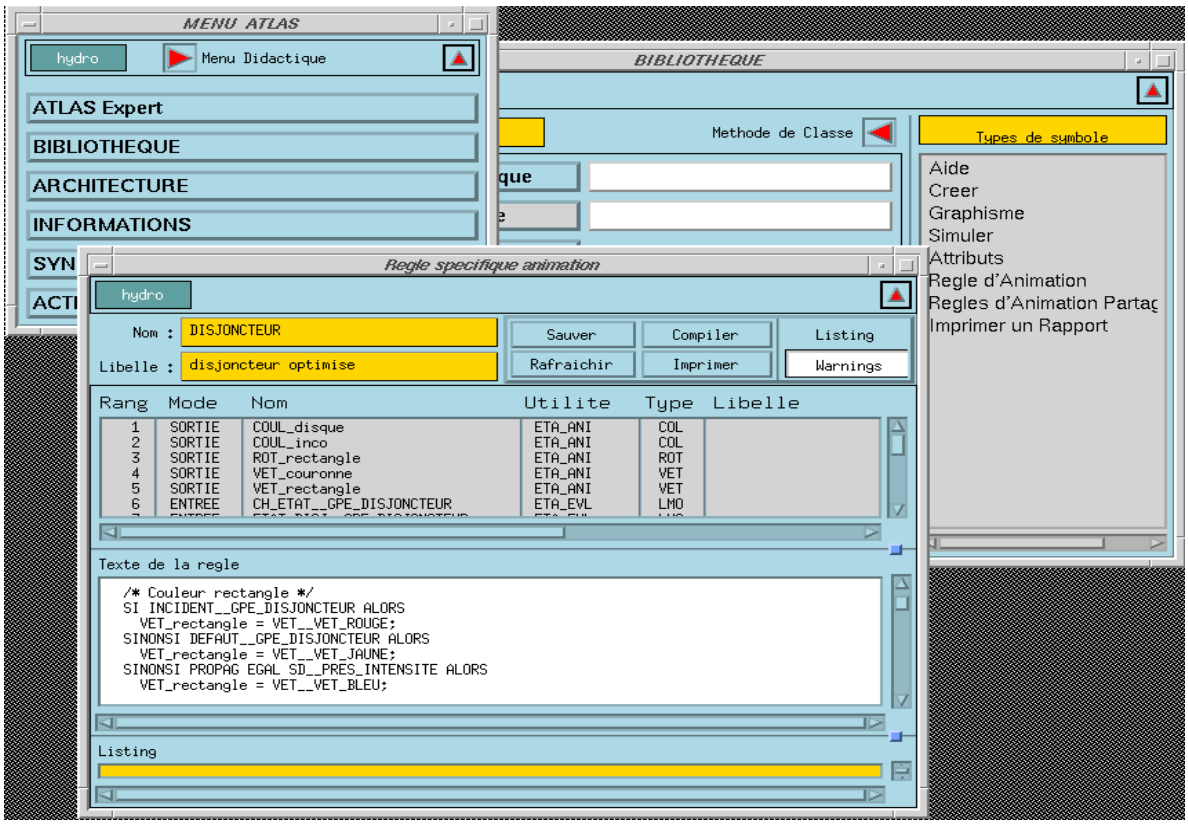

Figure 13 : Editeur des règles d'animations

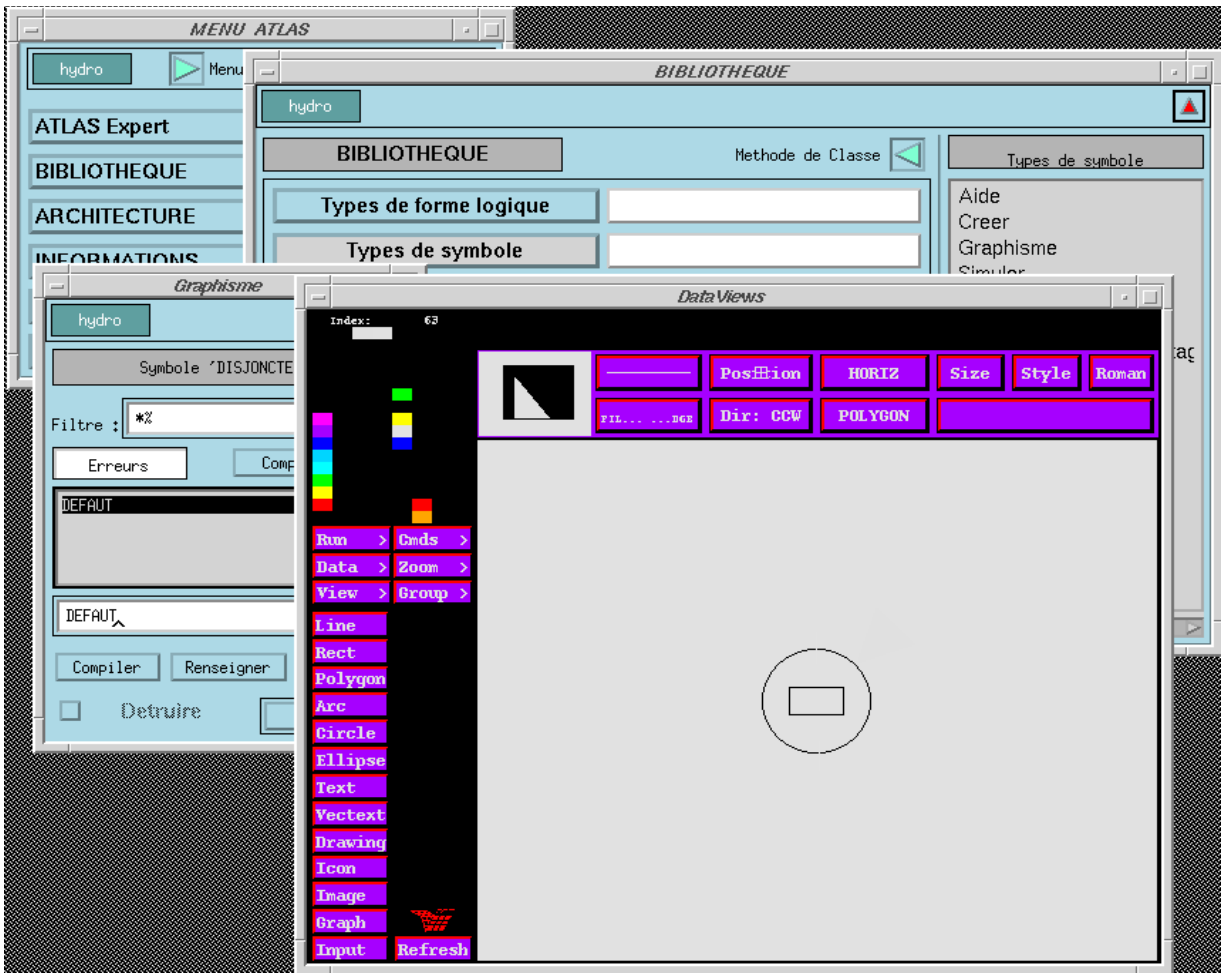

Figure 14 : Dessin d'un composant à l'aide de l'éditeur DVdraw

La bibliothèque contient également plusieurs tables accessibles par des masques de saisie développés sous SQLForms. Les tables principales sont les suivantes :

- **La table des types de réseaux :** pour chaque application, un réseau de propagation modélisant les différents liens entre les composants est construit. Ces liens sont des entités physiques non instrumentées telles que des conduites ou des barres de connexions électriques. On ne peut pas, par conséquent, déduire leurs états à partir d'informations “procédé”. Ces états sont évalués par l'algorithme de propagation d'état. De plus, celui-ci permet de déterminer d'autres états logiques associés aux composants de l'installation. Il est à noter qu'un algorithme de réseau spécifique au type de réseau à superviser est développé. Cet algorithme sert de base pour la tâche de traitement présente dans la machine cible.

- **La table des animations** : Cette table regroupe l'ensemble des animations utilisables dans une application par un type de symbole par exemple un symbole peut changer de couleur lorsqu'il entre en rotation, etc. Celles-ci sont exprimées sous forme de règles (Cf. IV.3).

- **La table des fluides :** Cette table contient les types de fluide qui peuvent circuler dans les symboles. Un fluide est défini par une animation particulière (couleur, hachurage, etc) et une dynamique des couleurs des différents fluide a été introduite pour permettre une visualisation facile des déplacements des fluides dans les tuyaux.

- **La Table des Couleurs** : Cette table intègre la définition des couleurs utilisables dans une application.

- **La Table des Hachurages** : Cette table intègre la définition des hachures qui permettent de visualiser le fluide passant dans les tuyaux.

- **La Table des états dipôles** : Cette table regroupe l'ensemble des synonymes associés aux différents états que peut prendre un dipôle type. Cette notion de dipôle est utilisée par l'algorithme de propagation, où chaque constituant est assimilé à un dipôle pouvant être un consommateur ou un producteur d'énergie ou de matière. Par exemple, le synonyme de Hors_Energie dans le cas d'un circuit hydraulique sera Hors_Débit, ou Hors_Tension dans le cas d'un circuit électrique. Cette notion de dipôle est utilisée par l'algorithme de propagation pour évaluer les états des différents composants.

Une fois la bibliothèque construite et validée, le spécifieur peut utiliser ses différentes entités et réaliser une ou plusieurs versions des synoptiques à l'aide du gestionnaire de prototypage. Ceci possède l'avantage de standardiser les composants de la bibliothèque d'une part, et de faciliter leur réutilisation d'autre part.

IV.4. Le gestionnaire de prototypage

Le spécifieur est l'utilisateur de l'atelier qui est en rapport avec les experts du procédé et avec les opérateurs de l'application de supervision. Sa fonction est de spécifier l'application, en liaison avec un spécialiste de la communication homme-machine de manière à ce qu'elle soit la plus ergonomique possible. Sa tâche est assistée par les fonctionnalités de prototypage qui prennent en compte les premières étapes de la démarche : la description du procédé, la réalisation et l'évaluation ergonomique d'un prototype, ainsi que la récupération des données de prototypage. Rappelons que seule l'étape de définition du prototype ne peut pas être assistée informatiquement, puisqu'elle correspond à une étape d'analyse conduisant le spécifieur, à partir de la description du procédé et d'un recensement des besoins informationnels des utilisateurs finaux, à définir des vues de prototypage. Dans le but d'assurer les fonctionnalités de prototypage, quatre modules principaux ont été prévus (figure 10). Le premier a été réalisé par le L.A.M.I.H. et intégré à l'atelier. Les trois suivants sont en cours de conception par 3IP :

- *Le gestionnaire de prototypage* : il permet d'abord au spécifieur de décrire les installations graphiquement. A partir de cette description, il lui est ensuite possible de réaliser rapidement des vues de supervision aboutissant à un prototype de l'application de supervision. C'est sur les

fonctionnalités de ce gestionnaire que nous nous focaliserons dans la suite de cet article.

- *Le récupérateur de données de prototypage :* l'équipe de développement ne sera encline à utiliser la démarche que si le travail fourni dans les premières étapes peut être récupéré, au moins en partie. Dans cette optique, un module de récupération génère, dans la base dite "développeur", des données à partir du prototype réalisé par le spécifieur. Cette récupération est réalisée à l'aide d'enchaînements automatiques de scripts SQL. Ceux-ci permettent essentiellement de recopier les informations liées aux types de symboles, et aux réseaux des constituants, dans la mesure où toutes les données relatives aux formes logiques ont été simplifiées de manière à procéder à une simulation (voir III.2.1.1.).

- *Le générateur de code Sun* : il a pour objectif de produire une application de supervision, destinée à s'exécuter sur la machine Sun. Cette option a été choisie dans la mesure où, dans la pratique, le spécifieur ne dispose pas forcément d'une machine cible pour prototyper ses vues. Cela s'explique par le fait que le spécifieur n'est pas basé sur le même lieu géographique que les développeurs. D'ailleurs la solution alternative qui consistait à fabriquer une machine cible dédiée exclusivement à la simulation du prototype s'avère prohibitive en raison de son coût.

- *Le simulateur sur machine Sun* : une fois les différentes vues créées, le spécifieur active dans un premier temps le générateur de code machine Sun. Puis, dans un second temps, il valide à l'aide du simulateur ses vues en situation simulée. Le simulateur présente les mêmes fonctionnalités que le simulateur destiné à la machine cible. En effet, le spécifieur simule les informations provenant des capteurs du procédé, ce qui a pour conséquence de modifier le comportement dynamique des vues de prototypage. Dans ce cadre, le spécifieur a la possibilité d'enregistrer des séquences de stimuli, puis de les rejouer par la suite. Cette notion de scénarii peut être utilisée en collaboration avec des spécialistes des facteurs humains, pour définir des situations caractéristiques en vue de l'étape d'évaluation ergonomique du prototype en situation simulée.

Ainsi, le gestionnaire permet la description du procédé ainsi que la conception et l'évaluation des vues de prototypage. De plus il est doté d'un module de récupération de données assurant le lien entre l'atelier ATLAS et l'outil de prototypage intégré dans celui-ci. Les données utiles à la génération de l'application de supervision à évaluer sont stockées dans une base de données "prototypage" et gérées à l'aide du gestionnaire de base de données Oracle (figure 10) .

Les deux fonctionnalités principales du gestionnaire de prototypage sont donc (i) la description du procédé et (ii) la description des vues de supervision. Elles sont successivement décrites ci-dessous.

IV.4.1. Fonctionnalité de description du procédé

Dans un premier temps, avant de construire progressivement les vues de supervision, le spécifieur procède à la description du procédé à l'aide d'un éditeur spécifique. La méthode retenue s'inspire de la

méthode diagrammes-blocs (voir Fadier, 1990). Elle repose sur un formalisme graphique, qui consiste à décrire le procédé et ses divers constituants, selon une démarche descendante, en partant du système global et en arrivant par affinements successifs aux composants élémentaires. Cette description met en évidence les liens entre les différents sous-systèmes et les composants.

Dans la mesure où le spécifieur décrit en final son installation à partir de composants de base qu'il assemble, l'utilité de la bibliothèque de composants est énorme. Comme nous l'avons signalé précédemment, la construction des vues peut s'inspirer fortement de la description architecturale de l'installation. De ce fait, nous avons pris l'option de construire une seule bibliothèque de composants qui sera utilisée à la fois pour la description de procédé et la description des vues. Cette bibliothèque doit permettre de modéliser le comportement graphique et logique de chaque constituant.

Elle présente des avantages qui sont à la fois d'ordre ergonomique et informatique. En effet, l'utilisation d'une bibliothèque offre la possibilité, une fois **validée**, de **standardiser** l'aspect graphique des composants (utilisation de symboles identiques). D'un point de vue ergonomique, cet aspect permet d'uniformiser les interfaces produites, et évite de disposer de différentes représentations graphiques pour un même composant. De plus, d'un point de vue informatique, cela permet de réduire les coûts de développement en offrant la possibilité de **réutiliser** des développements antérieurs. En effet, la composante graphique est réutilisable dans la majorité des cas, tandis que la composante logique est fonction de la technologie utilisée (Cf. II.3). Par exemple, un disjoncteur aura toujours la même représentation graphique, tandis que son comportement logique sera dépendant de l'instrumentation mise en œuvre, à cela près que l'on pourra toujours simuler les états logiques du disjoncteur (fermé/ouvert).

Dans un premier temps, le spécifieur a la possibilité de décrire globalement son installation (niveau 0 de la figure 15). Ce premier niveau de description consiste à identifier les différents sous-systèmes et/ou composants élémentaires (voir l'exemple en figure 16). A chaque sous-système identifié, un niveau de description inférieur peut être associé. Celui-ci permet de recenser les différents composants élémentaires le constituant. Les sous-systèmes et les composants de base sont dotés de points de connexions représentant les entrées et les sorties par lesquelles la matière ou l'énergie circulent. Ces différentes connexions permettent de construire au fur et à mesure le réseau des constituants. De plus, la connexion à chaque sous-système provoque la création d'une entrée/sortie. Celle-ci est visualisée d'une part sur le niveau contenant le sous-système et d'autre part sur le niveau qui lui correspond.

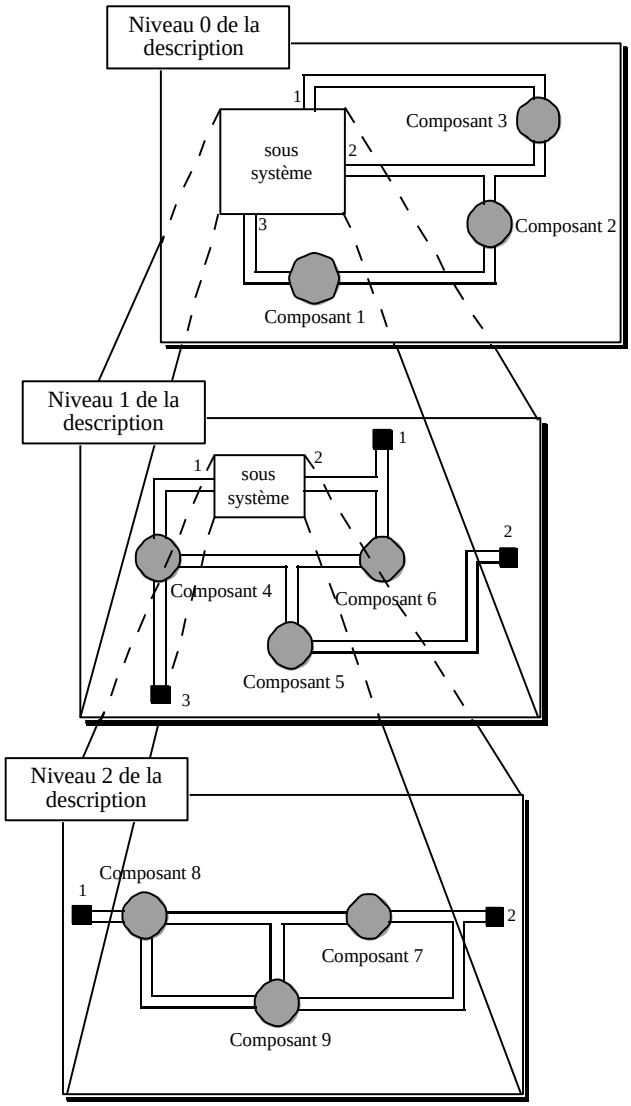

Figure 15 : Principes de description du procédé

Trois principales règles pour la description du procédé ont été émises : (i) l'instance d'un composant ne peut apparaître qu'une seule fois dans la description, (ii) seuls les composants élémentaires peuvent apparaître dans la description du procédé (pas d'afficheurs numériques par exemple) et (iii) la description est complète si et seulement si le réseau est entièrement défini et fermé. En effet, la première règle se justifie par le fait qu'un composant n'existe physiquement qu'une seule fois et donc il ne peut figurer de façon multiple dans la description. La description du procédé ne peut faire intervenir que les composants élémentaires de l'installation; par composants élémentaires on entend les composants par lesquels la matière ou l'énergie transitent : pompes, disjoncteurs, moteurs, vannes, etc. De plus dans la mesure où la description du procédé consiste à décrire les différents liens fonctionnels entre les composants élémentaires, il convient de relier tous les composants, c'est-à-dire toutes les entrées et sorties associées aux différents composants, et les derniers niveaux de description ne peuvent contenir de sous-systèmes. En effet, les sous-systèmes ne correspondent qu'à une vue abstraite englobant les constituants physiques de l'installation.

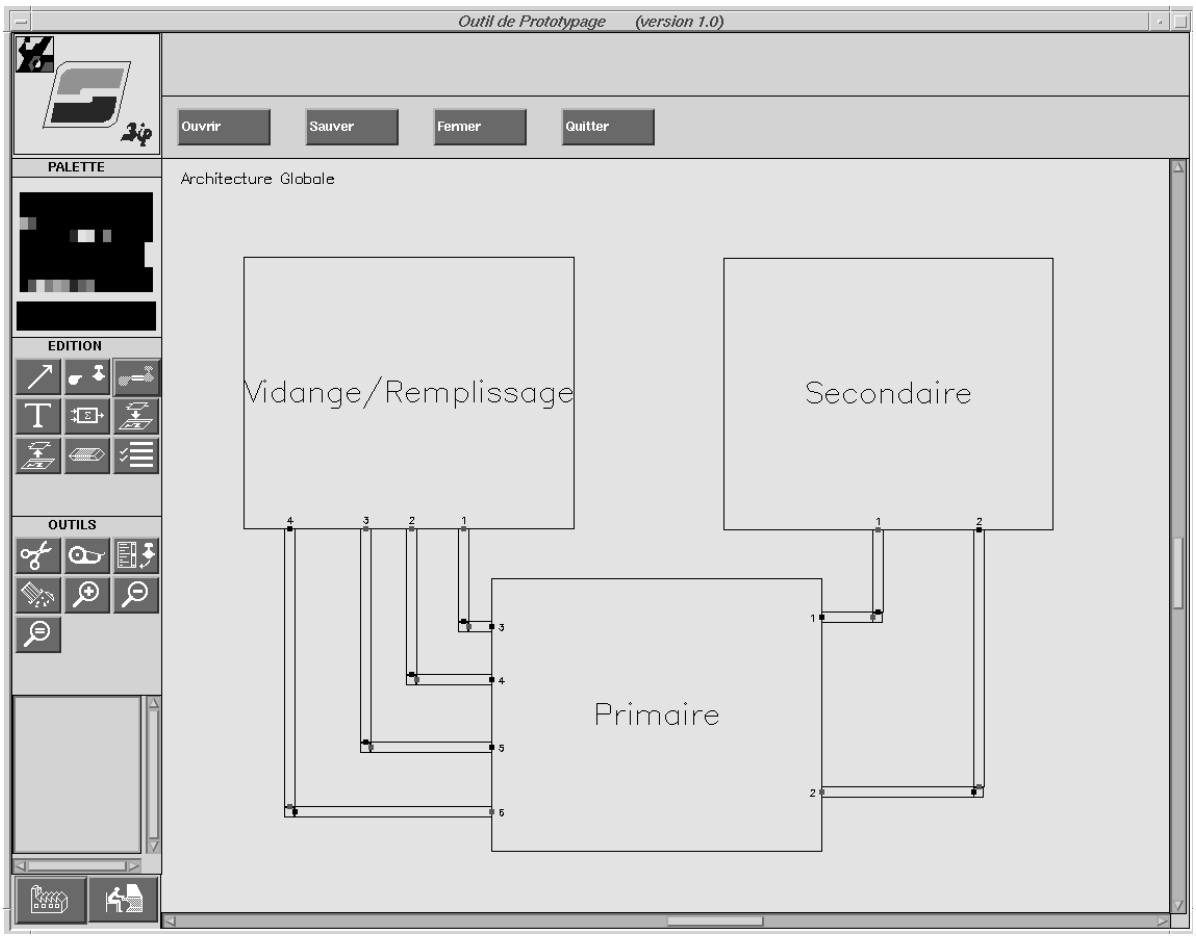

Figure 16 : Niveau 0 d'un procédé en cours de description par le spécifieur

Le spécifieur dispose d'un ensemble de fonctions facilitant la description du procédé, telles que :

- **Couper/coller :** cette fonction permet au spécifieur de déplacer des composants et/ou des systèmes sur les vues graphiques associées à la description du procédé. Il est à noter qu'il ne doit pas être possible de copier de composant, dans la mesure où un composant ne peut exister qu'une seule fois sur l'installation (cf. les règles énumérées précédemment).

- **gestion automatique des entrées/sorties** : la connexion à un sous-système provoque la création d'une part d'un point de connexion sur le sous-système proprement dit et d'autre part la création d'une entrée/sortie au niveau inférieur symbolisant la liaison du niveau supérieur sur le sous-système. De la même manière, la suppression d'une connexion provoque la disparition d'une entrée/sortie au niveau inférieur.

- **gestion des données de prototypage :** le spécifieur gère les données correspondant à la description du procédé : les liens entre les constituants, les constituants eux-mêmes, les données procédé qui leur sont associées. Ces données sont utiles pour produire le code nécessaire à une évaluation en dynamique et pour servir de base au codage du logiciel de supervision.

Toutes ces fonctionnalités permettent donc au spécifieur de décrire le procédé pour lequel on veut concevoir une application de supervision. Un exemple de page-écran de description d'un procédé fait l'objet de la figure 17. A partir de cette description, le spécifieur va concevoir ses vues de supervision. Pour cela, il doit disposer d'un ensemble de fonctions. Celles-ci sont présentées dans le paragraphe suivant.

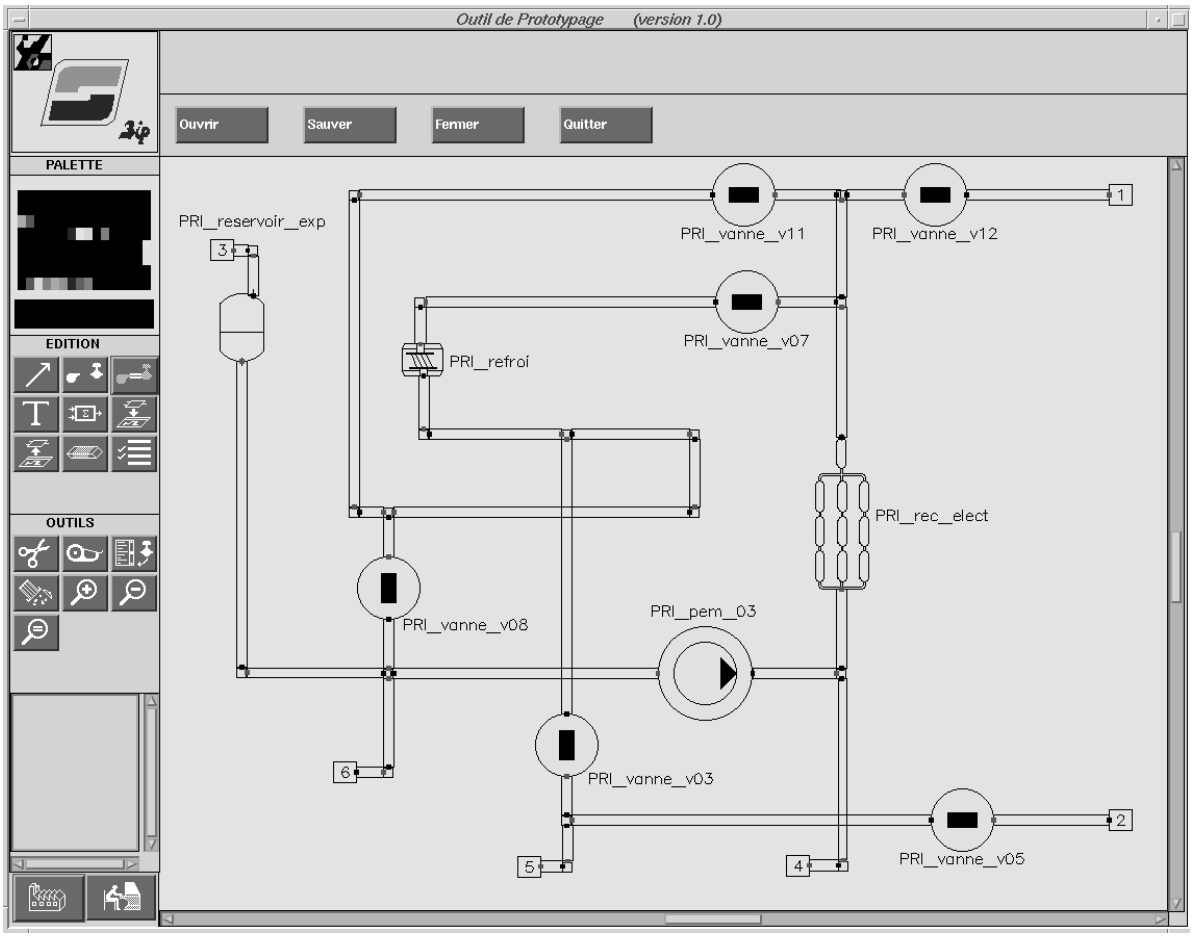

Figure 17 : Exemple de procédé en cours de description

IV.4.2. Fonctionnalités de description de vues graphiques de supervision

La description des vues graphiques consiste à partir de la description du procédé et des besoins informationnels des opérateurs à prototyper les vues que l'on va évaluer. Avant d'aborder la méthode permettant de décrire les vues, il convient de définir les différents éléments constituant une vue. En effet, deux principaux types de constituants visualisant l'état du procédé sont utilisés : (i) **les composants élémentaires** et (ii) **les informations complémentaires**. Les composants élémentaires, comme nous l'avons défini précédemment, correspondent aux composants physiques du procédé. Les informations complémentaires correspondent dans notre cas à la mise en forme de données prélevées sur l'installation. C'est par exemple le cas des afficheurs numériques, de messages d'alarmes, de courbes visualisant l'évolution temporelle d'une variable, ou encore de symboles permettant de synthétiser une ou plusieurs variables (Cf. Taborin, 1989), etc.

De la même manière que pour la description du procédé, la construction des vues repose sur une bibliothèque de composants. Néanmoins seuls les composants correspondant aux "informations complémentaires" doivent être disponibles. En effet, la description complète de l'installation ne peut être faite que lors de la description du procédé et cela en reliant les composants représentés par un symbole graphique. Il n'est donc pas cohérent de permettre au spécifieur de placer un nouveau composant élémentaire lors de la description des vues. Ces composants, correspondant aux informations complémentaires, sont également caractérisés par un comportement graphique et un comportement logique. Pour construire, selon la méthode présentée sur la figure 18, les vues de supervision, un ensemble de fonctions est disponible :

- **Récupération de la description du procédé :** la partie **synoptique** de la vue, représentée par les composants élémentaires constituant le réseau de l'installation, est réalisée par un *copier/coller* sur les vues de description du procédé vers les vues à prototyper. Il est à noter que l'on peut copier/coller autant de fois qu'on le désire un composant élémentaire. Cette récupération ne peut se faire qu'une fois l'installation décrite complètement et validée par les experts et les opérateurs. Cela est dû au fait qu'il doit y avoir cohérence des données saisies lors de la description du procédé avec celles récupérées par la fonction "copier/coller" pour créer la partie synoptique des vues.

- **Ajout d'informations complémentaires :** chaque vue peut être enrichie par des symboles graphiques complémentaires visualisant sous un format spécifique des données prélevées sur le procédé. Dans certains cas, il peut s'avérer utile de ne représenter sur une vue que des informations complémentaires par exemple des vues de tendance de type courbes.

- **facilité d'aménagement des vues** : l'outil de prototypage doit faciliter l'aménagement graphique des vues créées, et cela pour deux raisons principales : (i) afin d'inciter le spécifieur à utiliser l'outil en facilitant son travail et (ii) dans les séances d'évaluation menées avec les opérateurs et les ergonomes, il est important de pouvoir tenir compte rapidement de leurs remarques. Cette facilité d'aménagement passe par la possibilité de conserver les liens entre les symboles graphiques lors du déplacement des composants élémentaires. Ce point est important, dans la mesure où la majorité des vues comporte une partie synoptique.

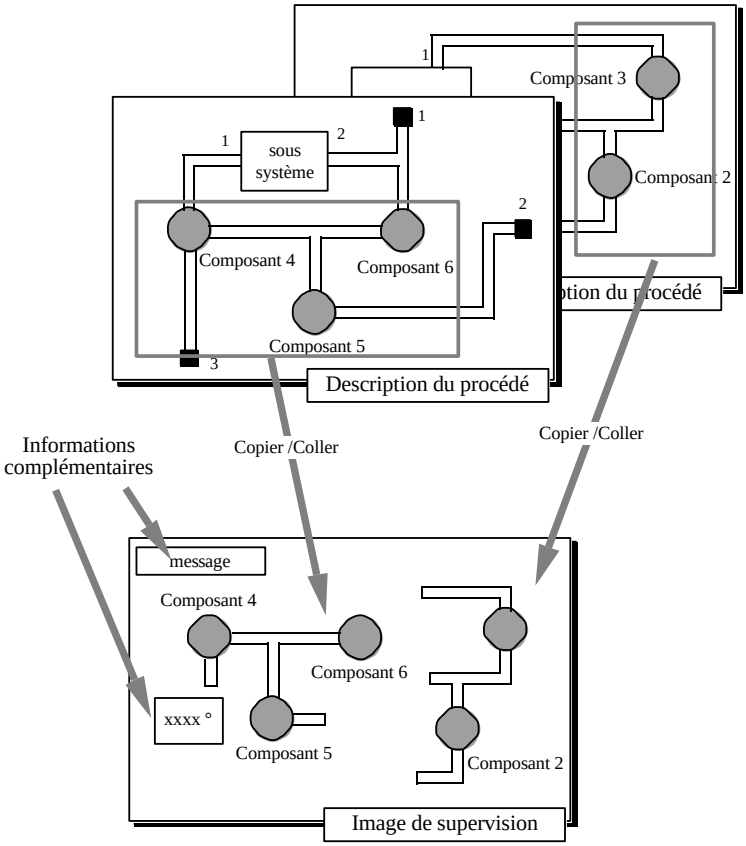

Figure 18 : Description des vues de supervision

- **Gestion des données du prototypage et production du code destiné à la simulation** : de la même façon que pour la description du procédé, il convient de mémoriser les données décrivant les vues. Cette mémorisation permet dans un premier temps de produire le code rendant possible l'évaluation en situation simulée des vues puis dans un second temps d'être récupérée pour les étapes suivantes de la démarche.

Un exemple de vue réalisée est visible en figure 19. Ainsi, les différentes vues sont créées en "récupérant" une partie de la description du procédé. De plus, elles peuvent être complétées par des informations complémentaires, comme le montre l'exemple de la figure où un barre-graphe indiquant la température dans le condenseur a été placé. Lors de la réalisation des vues de supervision, libre au spécificateur (i) de ne représenter qu'une partie "synoptique" - dans ce cas il ne récupère que des parties de description -, (ii) de "mixer" les parties "synoptiques" et les informations complémentaires, ou encore (iii) de n'utiliser que des informations complémentaires pour concevoir des vues de synthèse.

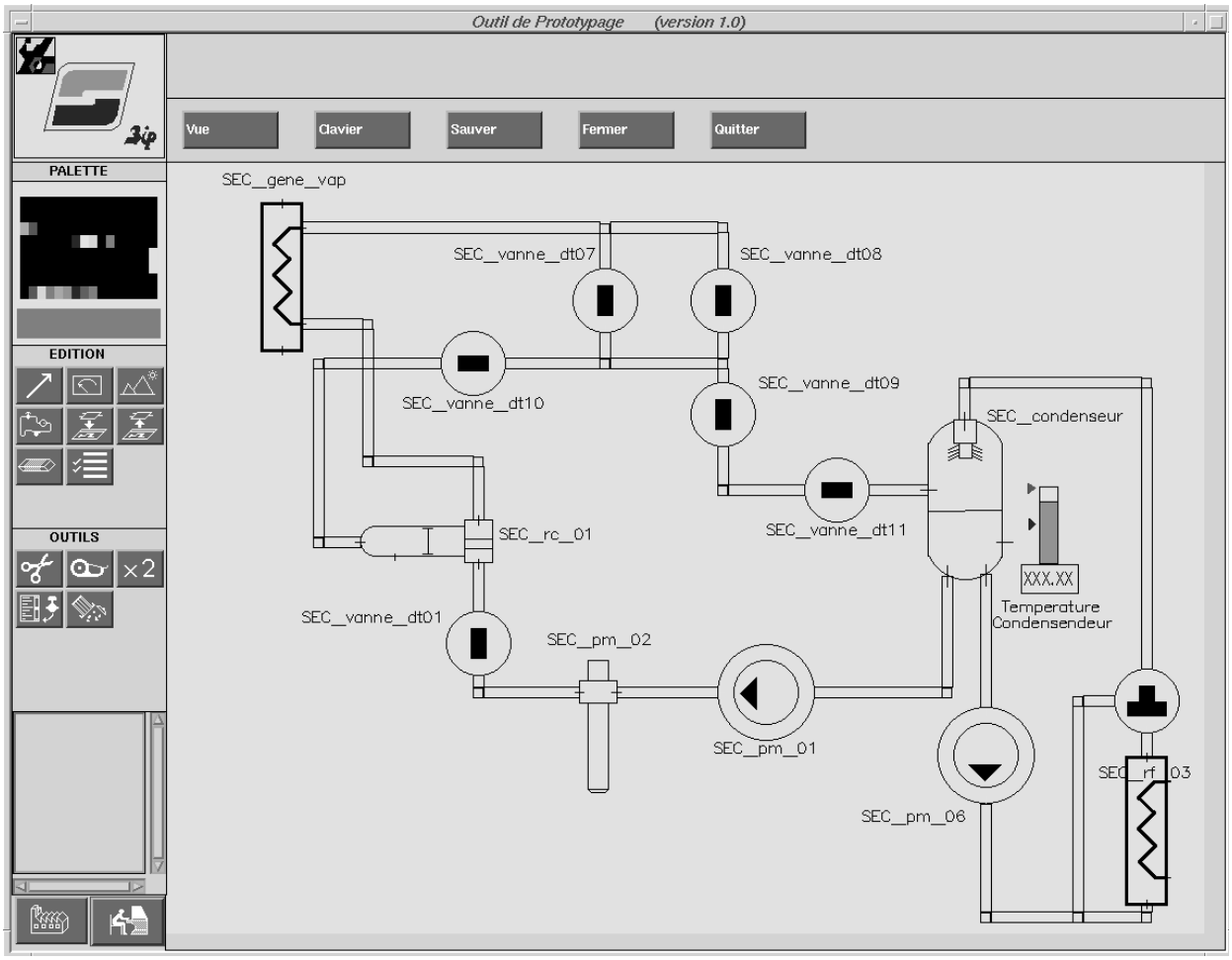

Figure 19 : Vue de supervision en cours de description

Lorsque le prototype des vues a été évalué et validé en collaboration avec les experts du procédé, les opérateurs et les ergonomes, le développeur peut se charger des dernières étapes de la démarche. Pour cela, il se sert du gestionnaire de développement. Il est à noter que dans ce cas, le gestionnaire de prototypage n'est qu'un outil supportant le démarche d'évaluation définie par l'équipe de développement (figure 8). Cette évaluation peut être menée selon différentes techniques (Cf III.2.1.4)

IV.5. Le gestionnaire de développement

A l'aide de fonctions de l'interface du gestionnaire de développement, le développeur a la possibilité de construire complètement une application de supervision. Pour cela, il est amené à spécifier les données de l'application selon quatre grandes classes : l'architecture de l'application, les synoptiques, les informations et les actions de l'opérateur.

- **L'architecture de l'application :** la description d'une application est décomposée selon trois aspects : (1) les blocs de traitements logiques contenant un réseau ainsi que les autres informations complémentaires telles que des compteurs numériques ou des barres-graphes, (2) les blocs de configuration, comprenant l'ensemble des blocs de traitement logique, (3) les organisations de

vues décrivant la structure physique des écrans de supervision.

- **Les synoptiques :** il s'agit pour le développeur de reprendre les vues réalisées par le spécifieur, pour fixer les différents types de forme logique réels. En effet, la phase de prototypage n'a pu être faite qu'à partir de formes logiques simplifiées (Cf III.2.1.5).

- **Les informations :** celles-ci correspondent aux systèmes de communication, pour l'acquisition des données, aux variables du procédé repérées par leurs adresses sur les bus de communication et aux moyens de filtrage des données.

- **Les actions opérateurs :** il s'agit de la description des différentes configurations du clavier fonctionnel (pour la Marine Nationale, un clavier fonctionnel spécifique a été conçu pour la machine cible). Pour chacune d'entre elles, on associe un ensemble de touches à partir desquelles des commandes sont activées. Celles-ci correspondent pour la plupart du temps à l'organisation des enchaînements des vues composant les différents synoptiques.

La spécification de ces données se fait grâce à l'enchaînement d'un ensemble de pages-écrans développées sous SQL-Forms, dans lesquelles il s'agit de compléter les données manquantes.

A partir du gestionnaire de développement, le développeur accède aux modules informatiques nécessaires à la production du code à partir des données stockées dans la base développeur. Ces différents modules sont au nombre de trois, figure 10 :

- **Le générateur de code cible** : celui-ci produit le code nécessaire à l'exécution d'une application de supervision. Cette application est constituée du cycle d'enchaînement suivant : l'acquisition, les traitements et l'affichage. Le code généré se présente sous la forme de fichiers sources écrits en langage C, répartis par type : (i) code d'enchaînement des structures de traitement logique et traitement d'animation, (ii) code d'animation, (iii) structure de données. La génération se fait à partir des informations stockées dans la base de données (ex : type de constituant : pompe, vanne...), et aboutit à un code exécutable sur la machine cible. Il utilise la chaîne de fabrication croisée du domaine public GNU qui génère du code 68030 à partir d'une SUN Sparc.

- **Le simulateur cible** : il a pour fonction de simuler les différents constituants des vues, par exemple la mise en marche d'un moteur, et par conséquent d'évaluer les vues de supervision avant de les télécharger sur la machine cible. Cette évaluation a pour seul objectif de valider l'aspect technique des vues. En effet, le développeur réalise complètement son application de supervision et la réalise, puis il valide ses développements, mais pas l'adéquation des vues avec les objectifs de conduite des opérateurs. Le simulateur cible est destiné à l'étape d'évaluation technique présentée dans la démarche proposée dans le chapitre précédent. Des séquences de stimuli peuvent être enregistrées pour créer des scénarios de simulation.

- **Le module de téléchargement** : Le code cible, une fois généré, est téléchargé via le réseau Ethernet par le module de téléchargement dans la machine cible. L'application de supervision pourra ainsi scruter les variables du procédé et animer les images de supervision.

V. Conclusion

Dans cet article, nous avons d'abord décrit une démarche de développement d'applications de supervision basée sur le prototypage. Cette notion est introduite dans les premières étapes de la démarche par la spécification, la conception et l'évaluation d'un ou plusieurs prototypes. La conception de ces applications implique des connaissances issues de domaines divers pour lesquelles se posent des problèmes de transfert. De ce fait, la démarche proposée fait apparaître à différents niveaux des personnes ayant des qualifications et des besoins différents, qu'ils soient les experts du domaine concerné ou les opérateurs qui vont avoir à utiliser les vues réalisées, ou encore toutes les personnes qui ont en charge le développement informatique (développeur, bibliothécaire), et aussi des spécialistes des facteurs humains tels que les ergonomes.

Une telle démarche ne peut être mise en œuvre que si l'on dispose d'un outil informatique prenant en charge une partie des étapes décrites. Un tel outil a été développé initialement par la CSEE (Compagnie des signaux et équipements électroniques) et 3IP (Société pour l'innovation, l'informatique industrielle et la productique) sous la forme d'un atelier de génie logiciel appelé ATLAS (Atelier Logiciel pour l'Animation de Synoptiques). Celui-ci permet, à partir d'une description du procédé, de concevoir des vues de supervision, susceptibles d'être embarquées.

Il est à noter que le travail des auteurs de l'article consistait dans un premier temps à proposer une démarche, et à réaliser dans un second temps l'outil informatique la concrétisant. Ainsi, dans le but de prendre en compte la fonctionnalité de prototypage dans l'atelier, un ensemble de modules a été prévu : (i) le gestionnaire de prototypage, (ii) le générateur de code Sun, (iii) le simulateur Sun et (iv) le récupérateur de données. Parmi les quatre modules informatiques cités, seul le gestionnaire de prototypage a fait l'objet de développements informatiques de la part du L.A.M.I.H. (Poulain et al., 1991, 1992 et 1993). Les trois autres modules sont actuellement en cours de conception par la société 3IP. Il faut noter qu'une version de base de l'atelier sans fonctionnalité de prototypage est actuellement disponible et utilisée par la Marine Nationale. La version avec fonctionnalité de prototypage devrait être opérationnelle dans le courant de l'année 1994.

Remerciements

Les auteurs remercient vivement l'ensemble des ingénieurs de la CSEE et 3IP ayant contribué à la réalisation de l'atelier, ainsi que les rapporteurs anonymes de la revue APII qui ont permis d'améliorer cet article. Un grand merci à Monsieur Didier Marsollier de la société 3IP pour le temps passé à la lecture critique de cet article.

Références bibliographiques

AFNOR, 1992, Représentation des systèmes de contrôle et de commande des systèmes automatisés, contribution à l'intégration des outils XAO du Génie Automatique : modèle conceptuel Base-PTA. Projet de norme française : BASE-PTA, version A1, janvier 1992.

BODKER S., GRONBAEK K., 1991, Cooperative prototyping : users and designers in mutual activity, *International Journal of Man-Machine Studies*, 34 (3), 453-478, March.

BOEHM B.W., 1982, Les facteurs du coût du logiciel. *Techniques et Sciences Informatiques*, 1 (1), pp. 5-24.

BOEHM B.W., GRAY T.E., SEEWALDT T., 1984, Prototyping versus specifying : a multiproject experiment. *IEEE transactions on Software Engineering*, 10 (3), May.

DE KEYSER V. et co-auteurs, 1992, The Nature of Human Expertise, *Rapport intermédiaire établi dans le cadre de la convention RFO/AI/18*, Université de Liège, Faculté de Psychologie et des sciences de l'éducation, 189 pages, Mars.

DANIELLOU F., 1986, L'opérateur, la vanne, l'écran : l'ergonomie dans les salles de contrôle, Montrouge, ANACT, collection "Outils et Méthodes".

FADIER E., 1990, Fiabilité Humaine : Méthodes d'analyse et domaines d'applications, In *Les Facteurs humains de la fiabilité dans les systèmes complexes*, J. LEPLAT et G. de TERSSAC éditeurs, Edition Octarés, Marseille.

HANCKOCK P.A., CHIGNELL M.M., 1989, Intelligent interfaces : Theory, Research and Design, North-Holland.

JAULENT P., 1990, Génie logiciel : les méthodes, 2° édition. Armand Colin, Paris.

JOHANNSEN G., 1992, Towards a new quality of automation in complex man-machine systems. *Automatica*, vol. 28 (2), pp. 355-373, March.

KEYSON D.K., PARSONS K.C., 1990, Designing the user interface using rapid prototyping, *Applied Ergonomics*, 21(3), 207 - 211, september.

KOLSKI C., 1993, *Ingénierie des interfaces homme-machine, conception et évaluation*. Editions Hermès, Paris, 372 pages, août.

KOLSKI C., TENDJAOUI M., MILLOT P., 1992, Methodology for designing Man-Machine "intelligent" interfaces : The "Decisional Module of Imagery" as a case study, *International Journal of Human Factors in Manufacturing*, vol. 2 (2), pp. 155-175.

LIND M. 1994, Modelling goals and functions of complex industrial plants, Applied Artificial Intelligence, 1994

MEINADIER J.P., 1991, L'interface utilisateur, pour une informatique plus conviviale. Dunod, Paris.

MILLOT P., 1988, Supervision des procédés automatisés et ergonomie, Editions Hermes, Paris.

MILLOT P., 1990, Coopération homme-machine : exemple de la téléopération, *Actes des Journées du GR Automatique*, Strasbourg, 17-19 Octobre.

PARTRIDGE D., 1990, *Apports de l'intelligence artificielle au génie logiciel.* Edition Masson, Paris.

POULAIN T., 1994, Contribution du génie logiciel à la conception et l'évaluation d'applications de supervision. Thèse de doctorat soutenue le 25 février 1994 à l'université de Valenciennes.

POULAIN T., GERME J.P., KOLSKI C., 1993, Un atelier de génie logiciel pour la spécification, la réalisation et l'évaluation de synoptiques industriels embarqués. *Génie Logiciel et Systèmes Experts*, n° 31, pp. 58-70.

POULAIN T., KOLSKI C., 1992, Monde réel et monde virtuel, la problématique du contrôle de procédé par un opérateur humain, *Actes de la Conférence Informatique 92 "L´interface des mondes réels et virtuels"*, EC2, Montpellier, 23-27 Mars.

POULAIN T., VILAIN B., KOLSKI C., MILLOT P., 1991, 1992 et 1993, *Atelier de création de synoptiques industriels.* Rapports de contrat dans le cadre d'une collaboration entre CSEE, 3IP et le L.A.I.H. faisant suite à un appel d'offres de l'ANVAR, décembre 1991, mai 1992, juin 1993.

PRADENC H., 1992, Superviseurs : des fenêtres ouvertes sur le process, *Revue AXES Robotique-Automatique,* 61, Janvier.

RASMUSSEN J., 1986, Information processing and Human-Machine Interaction, an approach to cognitive engineering, North Holland series in System Science and Engineering.

SCAPIN D.L., REYNARD P., POLLIER A., 1988, La conception ergonomique d'interfaces : problèmes de méthode, *Rapport de recherche n° 957*, INRIA, Décembre.

SENACH B., 1990, Evaluation de l'ergonomie des interfaces homme-machine, *Actes du Congrès ERGO-IA'90*, ergonomie et informatique avancée, Biarritz, 19-21 septembre.

SHERIDAN T.B., 1988, Task allocation and supervisory control, In *Handbook of Human-Computer Interaction*, M. Helander (ed.), Elsevier Science Publishers B.V., North-Holland.

SOMMERVILLE I., 1988, Le génie logiciel et ses applications, InterEdition, Paris.

TABORIN V., 1989, Coopération entre opérateur et système d'aide à la décision pour la conduite de procédés continus: application à l'interface opérateur système expert du projet ALLIANCE, *Thèse de Doctorat*, Université de Valenciennes.

VAN DAELE.A., 1993, La réduction de la complexité par les opérateurs dans le contrôle de processus continus. Contribution à l'étude du contrôle par anticipation et de ses conditions de mises en oeuvre. Dissertation présentée pour obtenir le titre de Docteur en Psychologie, Liège, 1993.

VILAIN B., RIERA B., MILLOT P., 1994, Towards a new way of presenting information to a supervision system, XIII European Annual Conference on Human Decision Making and Manual Control.

WILSON J.R., CORLETT E.N. (Eds.), 1990, *Evaluation of human works : a practical ergonomics methodology*. Taylor & Francis, pp. 890.