Un atelier de génie logiciel pour la spécification, la réalisation et et l'évaluation de synoptiques industriels embarqués

Thierry Poulain, Jean-Pierre Germe, Christophe Kolski

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

Un atelier de génie logiciel pour la spécification, la réalisation et l'évaluation de synoptiques industriels embarqués

Thierry POULAIN (1), Jean-Pierre GERME (2), Christophe KOLSKI (3)

(1) CSEE, ZA Courtaboeuf, BP80, 91943 Les Ulis Cedex
Boursier Cifre du Laboratoire d'Automatique Industrielle et Humaine (L.A.I.H.)

(2) Société pour l'Innovation l'Informatique Industrielle et la Productique (3IP)
104, Rue Castagnary, 75015 Paris, France

(3) Laboratoire d'Automatique Industrielle et Humaine (L.A.I.H.)
URA CNRS 1118, B.P. 311, Le Mont Houy, 59304 Valenciennes Cedex, France

Résumé

Nous présentons dans cet article un atelier logiciel destiné à la spécification, la réalisation et la validation de synoptiques industriels. Ces synoptiques sont susceptibles d'être embarqués. Cet atelier considère plusieurs types d'utilisateurs collaborant au processus de conception : (i) le Spécificateur du procédé à superviser, qui prend en compte les besoins des opérateurs humains, (ii) le bibliothécaire chargé de la gestion des éléments de base, servant à l'animation des synoptiques, (iii) le développeur du synoptique, et enfin (iv) l'administrateur chargé du bon fonctionnement de l'atelier.

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

I. Introduction

L'automatisation des procédés industriels complexes a permis de rendre des installations quasiment autonomes pendant leur fonctionnement normal. Cependant, la gestion des dysfonctionnements rend la présence de l'homme le plus souvent nécessaire. En effet, l'ensemble des dysfonctionnements n'est pas modélisable de façon exhaustive. De plus, la gestion des dysfonctionnements se base souvent sur des techniques issues d'un savoir faire typiquement humain [7, 14, 19, 23, 27].

Le "superviseur" humain se voit alors confier le rôle de garant ultime de la sécurité du système lorsque celui-ci est déficient. Cependant, pour réaliser ces tâches dans sa salle de contrôle, cet opérateur est la plupart du temps éloigné de la réalité du terrain et des installations. Outre les moyens de communication verbale mis à sa disposition [30], il 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, de type synoptiques, courbes, barre-graphe ou graphe de fluence, et relatifs à différents niveaux d'abstraction. 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 des outils graphiques est d'une importance fondamentale [21]. Dans ces conditions, qu'ils soient intégrés dans les salles de contrôle des procédés continus ou manufacturiers, ou encore embarqués sur des installations off-shore ou sur des engins de transport de plus en plus informatisés et automatisés, les logiciels de supervision deviennent indispensables à l'opérateur pour garder le contrôle des installations.

On recense actuellement une cinquantaine de logiciels standard de supervision disponibles en France, fonctionnant sur micro-ordinateurs, mini-ordinateurs ou stations de travail [22]. Cependant, dans les applications complexes, de tels logiciels s'avèrent rapidement inefficaces et sources de problèmes ergonomiques ou techniques, s'ils ne s'accompagnent pas d'outils informatiques et d'une méthode facilitant, à l'équipe de développement des synoptiques industriels, un *travail coopératif* intégrant une *démarche de prototypage* [3, 4]. Cette démarche exige l'utilisation d'outils de construction d'interfaces de plus en plus ergonomiques d'utilisation, permettant particulièrement la spécification à un haut niveau d'abstraction, le prototypage rapide, la réutilisation des interfaces et la réduction des coûts de conception.

En supervision de procédés comme dans la plupart des domaines d'applications où les interactions homme-machine jouent un rôle prépondérant (conception assistée par ordinateur, enseignement -“intelligemment” ou non- assisté par ordinateur, téléopération,

interactions homme / système expert, etc), de tels objectifs peuvent être maintenant visés (moyennant cependant la prise en compte de contraintes de développement inhérentes au domaine en question, voir plus loin). En effet, les outils et les principes de construction d'interfaces ont connu des progrès considérables ces dernières années. La première génération d'outils graphiques a été constituée principalement (1) de boîtes à outils, c.a.d. d'ensembles de primitives prêtes à l'utilisation pour la construction d'interfaces utilisateurs, (2) de systèmes génériques, prenant la forme de squelettes d'application, réutilisables et extensibles, voire (3) de machines à images abstraites, c.a.d. d'architectures logicielles de systèmes interactifs permettant par une série de transformations de passer d'une structure interne à afficher à une représentation graphique concrète. Cette première génération a fait l'objet de nombreuses études et critiques techniques et ergonomiques, voir à ce sujet celles de J. Coutaz [5]. La tendance actuelle consiste à proposer aux développeurs des systèmes de gestion d'interfaces utilisateurs (SGIU), définis par Hill [12] comme des "ensembles d'outils et de techniques pour la construction d'interfaces utilisateurs", définition à laquelle nous préférons celle de El Mrabet [9] : "un SGIU est un ensemble d'outils logiciels pour la conception, l'implémentation, l'exécution et l'évaluation des interfaces utilisateurs". Parmi l'ensemble des SGIU opérationnels ou en cours de validation que l'on recense dans la littérature technique, certains d'entre eux tels Aida/Masai (ILOG/INRIA), Ergolab (CNET/TNI) ou encore Smalltalk-80 s'insèrent naturellement dans une *méthode itérative par objets*, inspirée de la méthode de conception connue sous le nom de spirale de Boehm. Cette méthode, basée sur la conception par objets, leur réutilisation et la participation de tous les acteurs du projet, ouvre des perspectives prometteuses pour le développement d'interfaces [6]. D'après T. Jurain [15], tout en offrant une approche intuitive de construction de l'interface, cette méthode constitue un passage obligé pour les SGIU pour satisfaire les contraintes (1) de modularité des objets et des interfaces, (2) d'encapsulation des interfaces et (3) de présence de comportements prédéfinis et de comportement à définir pour les objets de l'interface. Cependant, par rapport au développement de synoptiques industriels, il faut souligner que les SGIU actuels sont pour la plupart trop limités :

- Ils permettent souvent exclusivement le développement d'interfaces de dialogue comprenant des menus, des fenêtres, etc, alors qu'une interface de supervision exige l'animation de variables inter-connectées ;

- ils n'offrent pas systématiquement la traduction du prototype (en Lisp, Smalltalk, ou autre langage interprété) en un langage plus approprié à des contraintes liées au temps réel que l'on retrouve dans la plupart des procédés industriels ;

- ils ne facilitent pas la création des interfaces en rapport direct avec une modélisation, sous la forme de variables en relation, de l'application (dans notre cas, un procédé industriel pouvant être constitué de différents réseaux comprenant plusieurs milliers de variable) ; dans ces conditions, une évaluation du prototype, déconnectée d'un modèle du procédé, peut s'avérer insuffisante vis-à-vis des contraintes de sécurité, d'économie ou de production inhérentes au procédé.

Pour ces raisons, une approche de méthode itérative par objets est considérée dans cet article qui présente un atelier de génie logiciel destiné à la spécification, à la réalisation et à la validation de synoptiques industriels. Ces synoptiques sont susceptibles d'être embarqués. Cet atelier considère plusieurs types d'utilisateurs collaborant au processus de développement : (1) le spécificateur du procédé à superviser, qui prend en compte les besoins des opérateurs humains, (2) le bibliothécaire chargé de la gestion des éléments de base, servant à l'animation des synoptiques, (3) le développeur du synoptique, et enfin (4) l'administrateur chargé du bon fonctionnement de l'atelier. Ainsi, après la description de la méthode utilisée, nous expliquons son application sous la forme d'un atelier de génie logiciel développé conjointement par la CSEE, 3IP et le LAIH dans le cadre d'un projet ANVAR.

II. Méthode de spécification, de réalisation et de validation de synoptiques industriels

La conception et l'évaluation de synoptiques de contrôle et de supervision de procédés industriels 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 [7] et Millot [20] décrivent des méthodes globales de conception de système homme-machine. Hancock et Chignell [11] et Kolski et coll. [17] proposent des méthodes de conception d'interfaces "intelligentes". Taborin [29] et Mandiau et coll. [18] recensent les différents types d'outils d'assistance aux opérateurs dans les salles de contrôle de procédés industriels. Senach [25, 26] et Wilson [31] recensent les techniques d'évaluation d'interfaces, etc.

Nos travaux de recherche n'ont pas pour objectif de fournir des outils ou des éléments de réponse concernant toutes les étapes d'une démarche de conception ergonomique des interfaces homme-machine (modélisation des tâches humaines a priori ou a posteriori, analyse de l'activité...), mais de proposer une méthode et un atelier de génie logiciel permettant aux concepteurs de formaliser et d'évaluer l'impact de leurs choix de conception.

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

En effet, Scapin [24] montre que certaines décisions de conception sont le fruit de compromis entre plusieurs critères. Par conséquent, et comme le montrent de nombreux auteurs issus de domaines aussi différents que l'informatique, l'automatique et l'ergonomie, l'utilisation d'outils reposant sur des techniques de prototypage rapide peuvent contribuer à l'amélioration ergonomique des interfaces homme-machine. La méthode proposée (figure 1) comprend deux phases : une première phase, descendante, consistant à élaborer et valider un prototype du synoptique à réaliser et une seconde phase, ascendante, destinée à concevoir et à valider le synoptique dans sa version définitive.

Adapté du modèle en “V”, on peut distinguer dans cette approche plusieurs étapes séquentielles qui sont :

- **Description du procédé :** Cette étape conduit à décrire et à définir le procédé à interfacer que ce soit sous un angle fonctionnel (par exemple la mission du système) ou sous un aspect structurel (les entrées, les sorties...), et ceci selon différents niveaux d'abstraction. Cette étude permet au concepteur 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. A ce sujet, Fadier (90) distingue deux types de méthodes : (1) les méthodes d'analyses en fonctionnement normal (diagramme blocs, graphes de fluence...), (2) les méthodes d'analyse du système en fonctionnement dégradé (AMDEC, arbres de défaillances...).

- **Définition rapide d'un prototype** : Cette étape consiste à spécifier les différentes images composant le synoptique. Celle-ci requiert, d'une part, la connaissance du procédé à interfacer, à partir des données issues de l'étape précédente, et d'autre part, une étude conduisant à recenser les besoins informationnels de l'opérateur. Ce recensement nécessite une analyse de la tâche et de l'activité réelle ou future probable des opérateurs (voir à ce sujet les travaux de Daniellou [7] et Abed [1]). Dans le cas de la supervision des procédés, il est important de spécifier l'ensemble des fonctionnalités des synoptiques; en conséquence, il est indispensable de définir un prototype et non une maquette des images de supervision, ces deux aspects différant de part les fonctionnalités développées. Cette étape de spécification est primordiale. En effet *prototyper* n'est pas *spécifier* [13], le prototypage, n'étant qu'un moyen de tester et de valider les spécifications, ne peut remplacer l'étape de spécification.

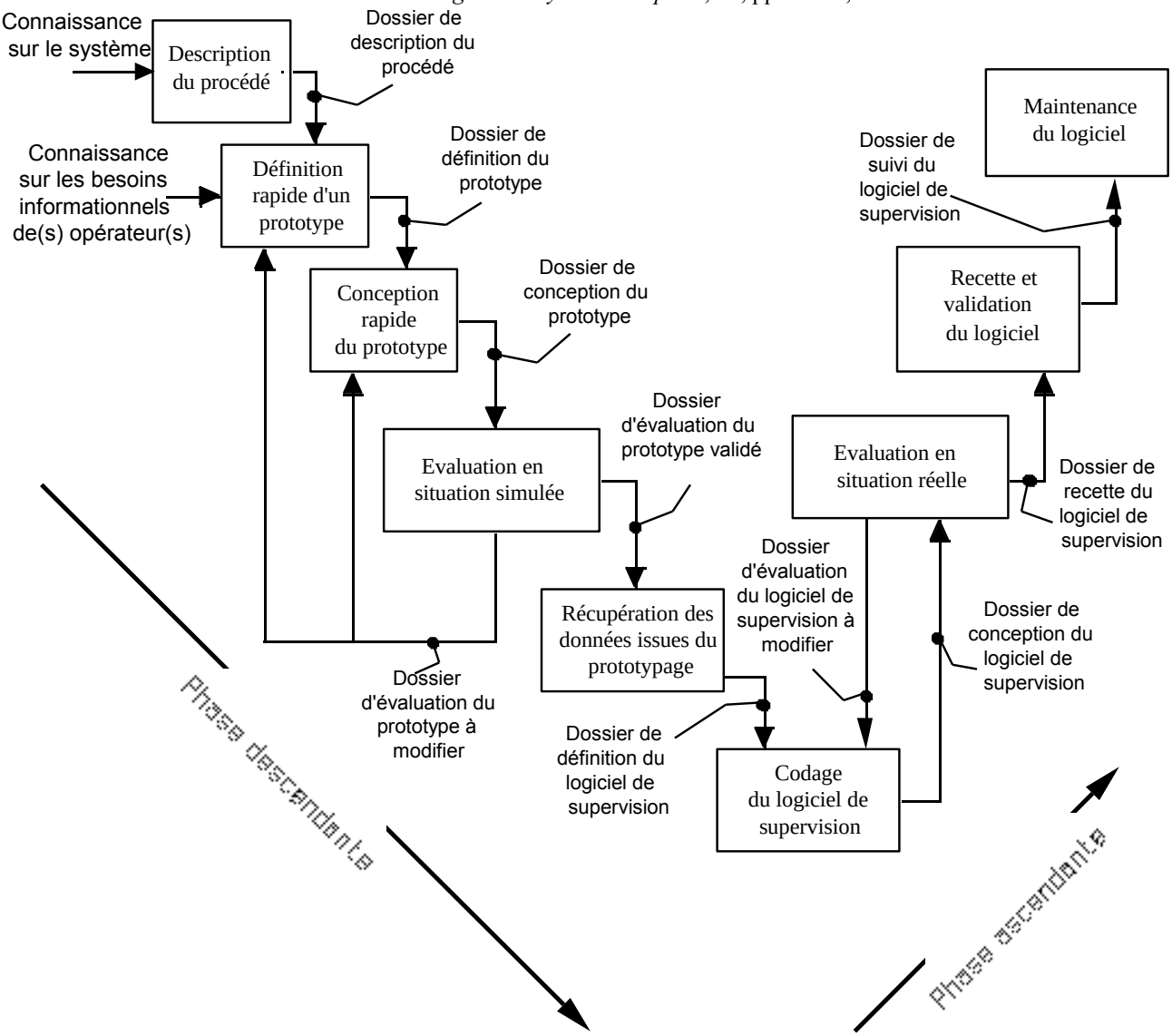

Figure 1 : Méthode de développement de synoptiques de supervision

- **Conception rapide du prototype** : Lors de cette étape, le concepteur réalise le prototype à évaluer. Le type d'outil mis en œuvre lors de cette étape conditionnera 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 du prototype et être facile d'utilisation, et cela après une période acceptable par les concepteurs [16]. De plus, la simulation du prototype doit être la plus réaliste possible, ceci dans le but de respecter la sémantique de l'application.

- **Evaluation en situation simulée :** Cette étape a pour objectif de tester les images réalisées lors d'un ensemble de situations simulées. Un ensemble de méthodes d'évaluation a été décrit dans la littérature [25, 26]. Celles-ci peuvent être regroupées en deux approches : (1) les méthodes analytiques qui visent à contrôler

la qualité de l'interface selon un modèle défini a priori, (2) les méthodes empiriques qui permettent d'évaluer à partir d'un ensemble de données comportementales l'ergonomie de l'interface. Il est à noter que dans notre cas, nous adoptons une approche empirique de type prototypage. Cette approche met l'accent sur les techniques de recueil mises en œuvre pour évaluer les choix de conception et, par conséquent, analyser les défauts rencontrés et en mesurer l'impact. Des travaux menés par Bewley et coll. [2] (cités dans [25]) montrent qu'il est impératif de bien préciser les objectifs et de définir les tâches de contrôle adéquates pour effectuer une évaluation de qualité.

- **Récupération des données issues du prototypage :** Une motivation supplémentaire du concepteur, à utiliser une telle démarche, réside dans la possibilité de récupérer au maximum le travail issu de la phase de prototypage. C'est pourquoi il faut veiller à engendrer, d'une part, une documentation du prototype (images, description de la sémantique de l'application...) et, d'autre part, à interfacer d'un point de vue logiciel l'outil de prototypage avec ceux utilisés lors des étapes suivantes.

- **Codage du logiciel** : Cette étape consiste à développer dans sa version finale le synoptique de supervision et à l'implanter sur le site. Une génération automatique maximale de code est préconisée lors de cette étape.

- **Evaluation en situation réelle** : Il est indispensable, une fois le logiciel de supervision opérationnel et implanté sur le site, d'effectuer une nouvelle évaluation. En effet, lors de la phase de prototypage, l'évaluation a été effectuée, d'une part, en situation simulée, où le stress et la motivation des opérateurs ne sont pas conformes à la réalité du terrain, et, d'autre, part dans un environnement “réduit” ce qui ne permet pas l'évaluation de l'interaction entre les autres éléments de la salle de contrôle (informations provenant de rondiers, autres moyens d'affichage, imprimantes...).

- **Recette et validation du logiciel** : Cette étape a pour but de vérifier l'adéquation du logiciel aux besoins du client et donnera lieu à la mise en œuvre de scénarios de recette.

- **Maintenance :** Cette dernière étape consiste à maintenir le logiciel réalisé. A ce sujet, comme le rapportent de nombreux auteurs [28], les coûts de maintenance peuvent être deux à trois plus élevés que les coûts de développement dans le cas de

systèmes complexes. L'expérience montre alors que le coût de maintenance est principalement dû à des changements dans la définition des besoins et non pas à des erreurs. De ce fait, cette approche reposant sur le prototypage rapide, dans la mesure ou elle permet de mieux définir le besoin, doit conduire à la réduction des coûts du projet.

Nous informatisons actuellement les six premières étapes de la phase descendante de cette méthode sous la forme d'un atelier de génie logiciel appelé ATELIER-UTV.

III. L'atelier ATELIER-UTV

La méthode présentée précédemment fait l'objet actuellement d'une formalisation et d'une intégration dans un atelier de génie logiciel, appelé provisoirement ATELIER-UTV (Unité de Traitement de Visualisation), initialement créé pour le compte de la Marine Nationale. Cet atelier est destiné à la spécification, à la réalisation et à la validation de synoptiques industriels de type “images de surveillance” de réseaux (électriques, hydrauliques) embarqués à bord des navires. L'une des contraintes initiales de l'atelier est de faire en sorte de limiter l'intervention des personnels non informaticiens lors de la réalisation des synoptiques.

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.

III.1. Principe de base de l'atelier

L'atelier utilise l'environnement matériel suivant : (1) une station de travail Sun servant de support à l'atelier pour les aspects de spécification de l'application, réalisation des synoptiques industriels et validation selon une approche de prototypage ; (2) une machine cible, image de la machine embarquée, reliée via ETHERNET à la station de travail et servant aux tests de validation finale de l'application (figure 2).

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

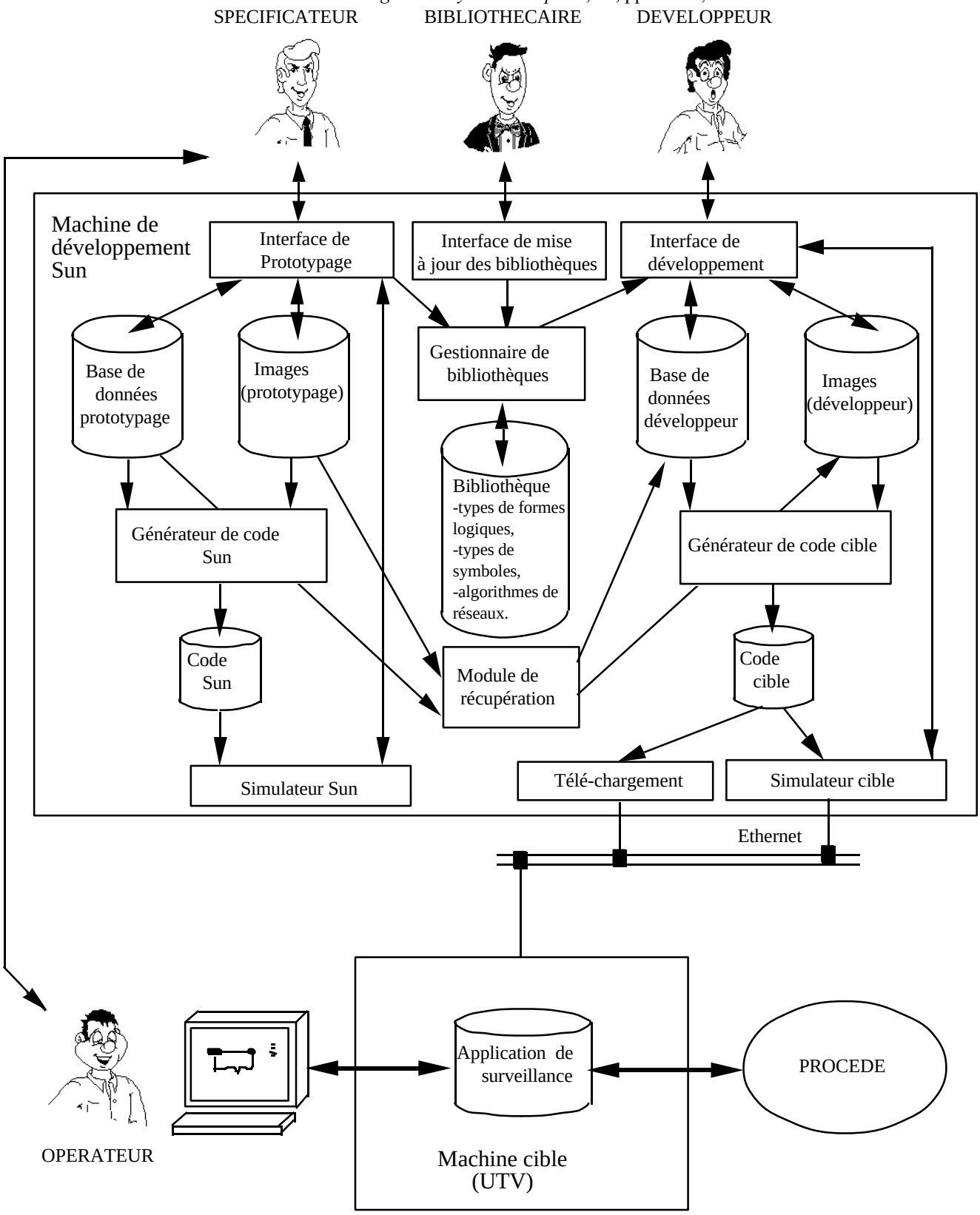

Figure 2 : Architecture de l'ATELIER-UTV

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

Le principe de base de l'atelier repose sur la notion de bibliothèque de composants (symbolisant des pompes, des vannes, des moteurs par exemple) gérée par un bibliothécaire. A partir des éléments de cette bibliothèque, le spécificateur décrit interactivement dans un premier temps le réseau de l'installation à superviser (constitué particulièrement de différents composants connectés entre eux). Puis, dans un second temps, il réalise un ou plusieurs prototypes des images de supervision. Les images peuvent être ensuite évaluées en dynamique à l'aide d'un simulateur sur Sun, avec les opérateurs, et en liaison éventuelle avec des spécialistes des facteurs humains, et ceci afin d'évaluer l'adéquation de l'imagerie aux besoins nécessaires à la réalisation des tâches de supervision. Puis, un module de récupération permet d'extraire une partie de la base de données "prototypage" obtenue précédemment ainsi que les images validées, et ceci afin de constituer les images finales et un sous-ensemble de la base de données nécessaire à l'animation des images sur la machine cible. Le développeur complète ensuite les informations nécessaires à la production du code qui sera ensuite téléchargé puis exécuté sur la machine cible. Un second simulateur, appelé "simulateur cible" (ayant les mêmes fonctionnalités que le simulateur sur Sun) a pour rôle de valider, non pas le contenu des images, mais certains aspects techniques (par exemple des adresses pour l'acquisition des données). Les différents modules informatiques composant l'atelier, ainsi que les différents utilisateurs de celui-ci, seront progressivement décrits dans la suite de l'article. Ecrit principalement en C, l'atelier repose sur les outils logiciels de base suivants :

- **TeleUse** : ce générateur d'interfaces sous **X-MOTIF**, commercialisé par la société TeleLogic Europe, nous a permis essentiellement de décrire l'interface homme-machine d'utilisation de l'atelier sur la machine de développement Sun (Interfaces de prototypage, de mise à jour des bibliothèques et de développement, voir figure 2). Cet outil est composé d'une part d'un éditeur graphique (VIP) permettant la construction des interfaces de dialogue à partir de Widgets Motif et d'autre part d'un langage (appelé D) faisant le lien entre les interfaces créées sous éditeur et l'application (gestion des événements et manipulation des différents widgets).

- **DATAVIEWS** : cet outil est utilisé pour la description des synoptiques sur la machine de développement Sun, et pour leur animation graphique sur la Sun 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 images de prototypage (éditeur intégré à l'interface de prototypage) et d'autre part pour produire le code destiné à la machine cible. Contrairement à TeleUse, DATAVIEWS n'est pas conçu initialement pour créer des interfaces de dialogues.

- **ORACLE** : ce SGBD sert au stockage des données décrivant l'application,

- **VxWorks**, dont le fournisseur est WindRiverSystems, sert de système d'exploitation temps réel de base à la machine cible. VxWorks intègre, en couche logiciel de base, un noyau temps réel (pouvant être VRTX, PSOS ou WRSK). VxWorks apporte des fonctionnalités de haut niveau par rapport au noyau temps réel : librairies de fonctions standardisées, interfaces réseaux, interpréteur de commandes, dévermineur et désassembleur symbolique.

Avant de décrire chacun des modules constituant l'atelier, il est utile de présenter les fonctions nécessaires pour la mise en œuvre d'une application de surveillance.

III.2. Une application de surveillance

La figure 3 résume l'architecture d'une application type de surveillance, telle qu'elle est engendrée par la machine de développement et transférée ensuite dans la machine cible. On y voit apparaître quatre grandes fonctions :

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

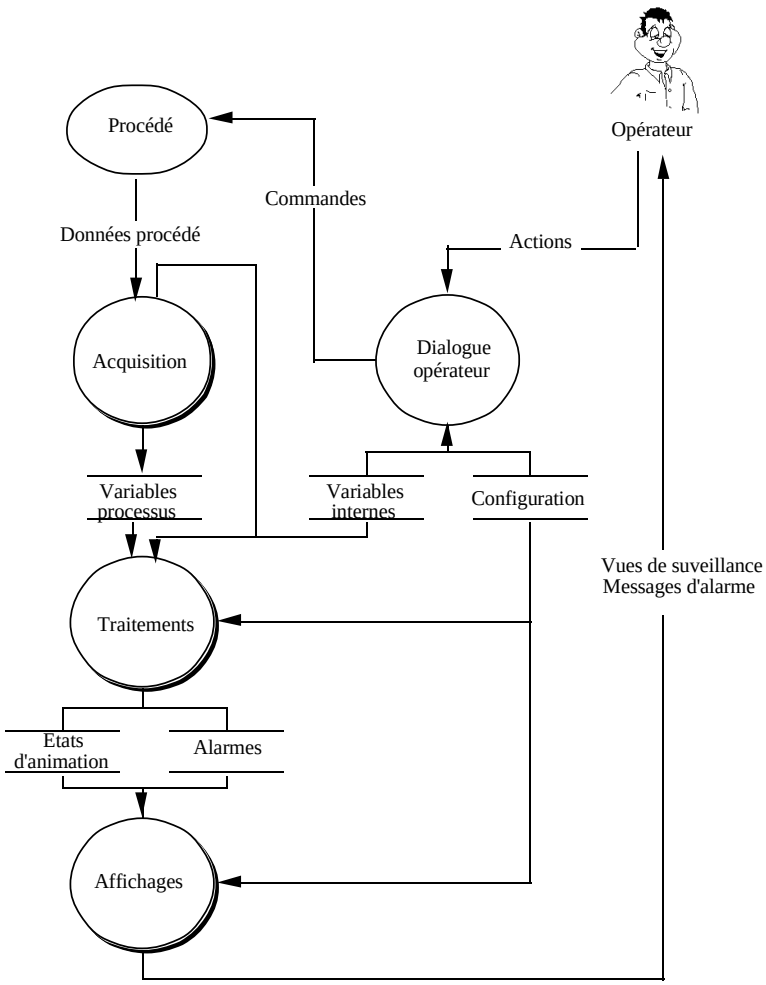

Figure 3 : Application de surveillance.

- **Traitements** : c'est dans cette fonction que réside la logique permettant, d'une part, de passer d'un ensemble de données provenant du processus 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 intrumentés 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 :

  • lors de la première étape, on détermine les états évalués suivants :

  - La vanne est ouverte;
  - La pompe est en marche.
- La propagation permet de fixer un élément supplémentaire
  - La vanne est en débit;
- La troisième étape détectera par exemple une incohérence si le capteur de débit ne signale pas de fluide alors que la vanne est en débit.

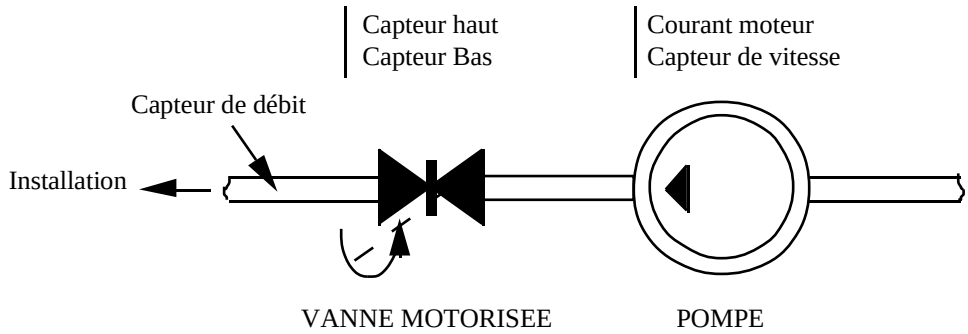

Figure 4 : Sous-ensemble d'un réseau

- **Affichages** : cette fonction traduit graphiquement le résultat des fonctions précédentes et cela en modifiant l'aspect graphique des symboles présent à l'écran.

- **Dialogue "opérateur"** : cette fonction gère les relations avec l'opérateur de surveillance, comme par exemple des changements de vues.

III.3. Description fonctionnelle de la machine de développement

Reposant sur les six premières étapes de la méthodologie (figure 1), et afin d'aboutir à des applications de surveillance telles qu'on vient de les décrire, l'ATELIER-UTV est organisé en quatre grands groupes de fonctions destinés chacun à un utilisateur particulier. Les différents modules visibles sur la figure 2 et pilotables par les différentes interfaces homme-machine, sont les suivants :

- **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 (ex: vannes, pompes, disjoncteur...) pour former un ensemble de réseaux. Les synoptiques sont construits à partir des différents éléments composant le procédé et

d'éléments complémentaires de visualisation d'informations (cadrans, afficheurs numériques...).

- **Générateur de code Sun** : Le code nécessaire à l'exécution d'une application de surveillance est constitué du cycle d'enchaînement suivant : l'acquisition, les traitements et l'affichage. Le code engendré se présente sous la forme de fichiers source C, répartis par type : (1) code d'enchaînement des structures de traitement logique et traitement d'animation, (2) code d'animation, (3) structure de données Sun. 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 Sun.

- **Simulateur Sun:** Celui-ci a pour fonction de simuler les différents constituants des images (ex : mise en marche d'un moteur), et, par conséquent, d'évaluer les images de supervision sur la Sun. Des séquences de stimuli peuvent être enregistrées pour créer des scénarios de simulation.

- **Simulateur cible:** Le simulateur cible remplit les mêmes fonctionnalités que celui dédié à la Sun; néanmoins dans ce cas l'évaluation s'effectue directement sur la machine cible.

- **Module de récupération :** Ce module a pour but de recopier une partie des informations de la base de données de prototypage dans la base de données de développement. Il est à noter que les images de supervision seront entièrement récupérées dans la base des images de développement.

- **Générateur de code cible :** Ce générateur de code cible présente les mêmes fonctionnalités que le générateur de code Sun, la différence étant qu'il utilise la chaîne de fabrication croisée du domaine public GNU, entre une machine Sparc et un processeur 68030.

III.4. Les utilisateurs de l'atelier.

En plus de l'ADMINISTRATEUR, chargé du bon fonctionnement d'ensemble et qui a accès à des fonctions de gestion de comptes et d'applications, trois types d'utilisateur de l'atelier sont considérés (figure 5): le BIBLIOTHECAIRE, le SPECIFICATEUR et le DEVELOPPEUR, dont les tâches sont décrites ci-après.

In : Génie Logiciel et Systèmes Experts, 31, pp. 58-70, 1993.

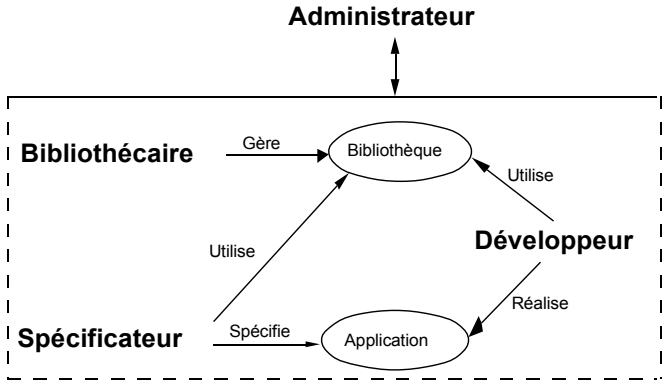

Figure 5 : Les utilisateurs de l'atelier

III.4.1. Le Bibliothécaire

Le BIBLIOTHECAIRE est l'utilisateur qui a en charge la bibliothèque. Il est le garant du bon fonctionnement de celle-ci. Il la délivre au SPECIFICATEUR et au DEVELOPPEUR. La bibliothèque est composée de deux types de données : les données instanciables (les types) et les données référençables (les tables). Les tables correspondent à un ensemble d'informations élémentaires nécessaires à la construction de la bibliothèque et donc d'une application. Les types utilisés pour engendrer les données de l'application sont accessibles par le menu visible en figure 6 :

- **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 venant du procédé. Citons l'exemple suivant : une vanne motorisée est caractérisée par les informations “procédé” suivantes : capteurs de position haut et bas, capteur d'intensité, les règles logiques associées peuvent être “si le capteur haut et le capteur bas ne sont pas actionnés alors la vanne est en train de s'ouvrir”, les différents états logiques déduits peuvent être “vanne ouverte”, “vanne fermée” ou “en défaut”.

- **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). Prenons l'exemple suivant : *“si la vanne est en défaut alors sa couleur est rouge”*.

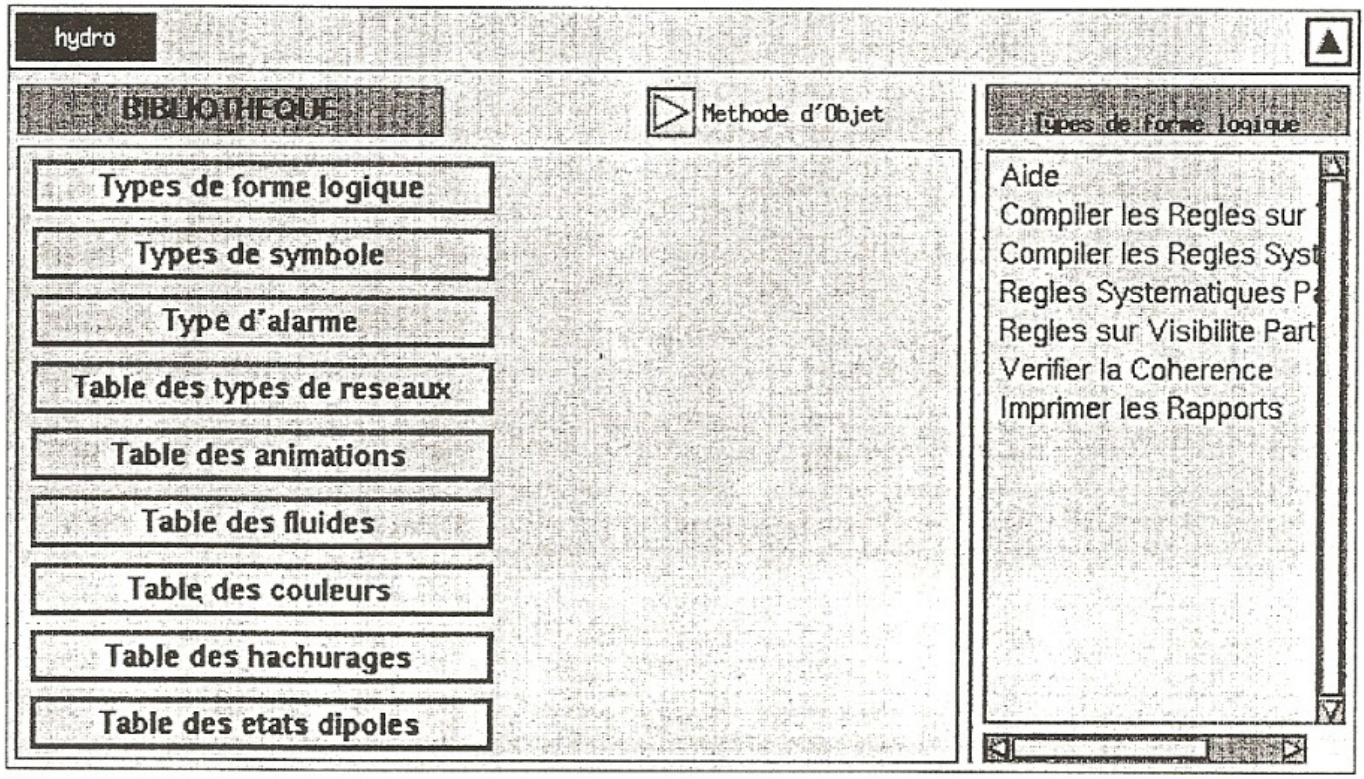

Figure 6 : Menu d'accès à la bibliothèque

- **Les types d'alarmes :** pour chaque constituant de l'installation, 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 *le courant absorbé par un moteur est supérieur à la normale*.

- **Les algorithmes 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 (conduite, barres de connexions électriques...). On ne peut pas, par conséquent, déduire leurs états à partir des informations “procédé”. Ces états sont évalués par un algorithme de propagation d'état. De plus, l'algorithme de réseaux permettra de déterminer d'autres états logiques associés aux composants de l'installation (par exemple une vanne est ouverte, et une pompe placée en amont est en marche ; de ce fait l'algorithme en déduit que la vanne est

sous pression). Il est à noter qu'un algorithme de réseau spécifique au type de réseau à superviser est développé.

- **La table des animations :** cette table regroupe l'ensemble des animations utilisables dans une application par un type de symbole par exemple un symbole pourra changer de couleur entrer en rotation, etc...

- **La table des fluides :** cet objet contient les types de fluide qui peuvent circuler dans les symboles. Un fluide est défini par une animation particulière (couleur, hachurage...).

- **La table des couleurs et hachurages :** cette table intègre la définition des couleurs et hachurages pour spécifier une application.

- **La table des états dipôles :** cette table regroupe l'ensemble des synonymes associés aux différents états que peuvent prendre un dipôle type. 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.

III.4.2. Le spécificateur

Le SPECIFICATEUR est l'utilisateur de l'atelier qui connaît le procédé. Il est en rapport avec les concepteurs du procédé et avec les futurs utilisateurs de l'application de surveillance. Sa fonction est de spécifier l'application (ainsi que les compléments de bibliothèque) et en particulier l'ergonomie des synoptiques, en liaison avec un spécialiste de la communication homme-machine.

Ce type d'utilisateur est chargé des premières étapes de la méthode présentée précédemment (description du procédé, définition et conception d'un prototype et son évaluation). Dans ce but, il s'agit de lui proposer des fonctionnalités faciles à utiliser, le déchargeant au maximum de tâches répétitives de description. Celui-ci utilise l'interface “de prototypage” (figure 2) qui permet les fonctionnalités suivantes :

- **Description du procédé :** à l'aide d'un éditeur graphique, le SPECIFICATEUR décrit son installation. Il dispose pour cela de la bibliothèque réalisée par le BIBLIOTHECAIRE. La description du procédé repose sur une méthode d'analyse

systémique, désormais classique, décrite par de nombreux auteurs tel E. Fadier [10] : la méthode “diagramme blocs”. Celle-ci consiste à décrire graphiquement le système et ses divers composants, en partant du système global et en arrivant aux composants élémentaires par affinements successifs. Cette méthode présente l'avantage d'aboutir à une bonne description du procédé et facilite également le dialogue entre les différents intervenants. Tout au long de sa description, le réseau des constituants est mémorisé dans la base de données de prototypage. Comme nous l'avons vu précédemment, un composant est caractérisé par son type de comportement logique et son type de symbole. Le type de comportement logique a pour but de déterminer les états logiques du composant en fonction des données provenant de son instrumentation (capteurs). Or, pour la phase de prototypage, il n'est pas utile de disposer du type de comportement logique réel. Dans ce but, le SPECIFICATEUR dispose d'un ensemble de types de formes logiques “simplifiées”. La figure 7 montre un exemple de procédé en cours de description par le SPECIFICATEUR. Sur cette page-écran, on distingue la zone de travail (zone centrale sur la figure) sur laquelle un ensemble de composants ainsi qu'un sous-système sont inter-connectés. Sur la partie gauche de l'écran, trois ensembles de fonctions sont accessibles :

- *Edition* : des fonctions permettent, d'une part, de créer et de manipuler des composants élémentaires (pompes, vannes, disjoncteur ...), et, d'autre part, de créer différents sous-systèmes et de naviguer dans ceux-ci

- *Outils* : des fonctions disponibles permettent de couper et coller une partie de la description du procédé, d'accéder au informations stockées dans la base de données de prototypage, d'aligner selon l'axe vertical ou horizontal des symboles graphiques, d'effectuer des zooms.

- *Commandes* : d'autres fonctions permettent de gérer les différents fichiers de description du procédé.

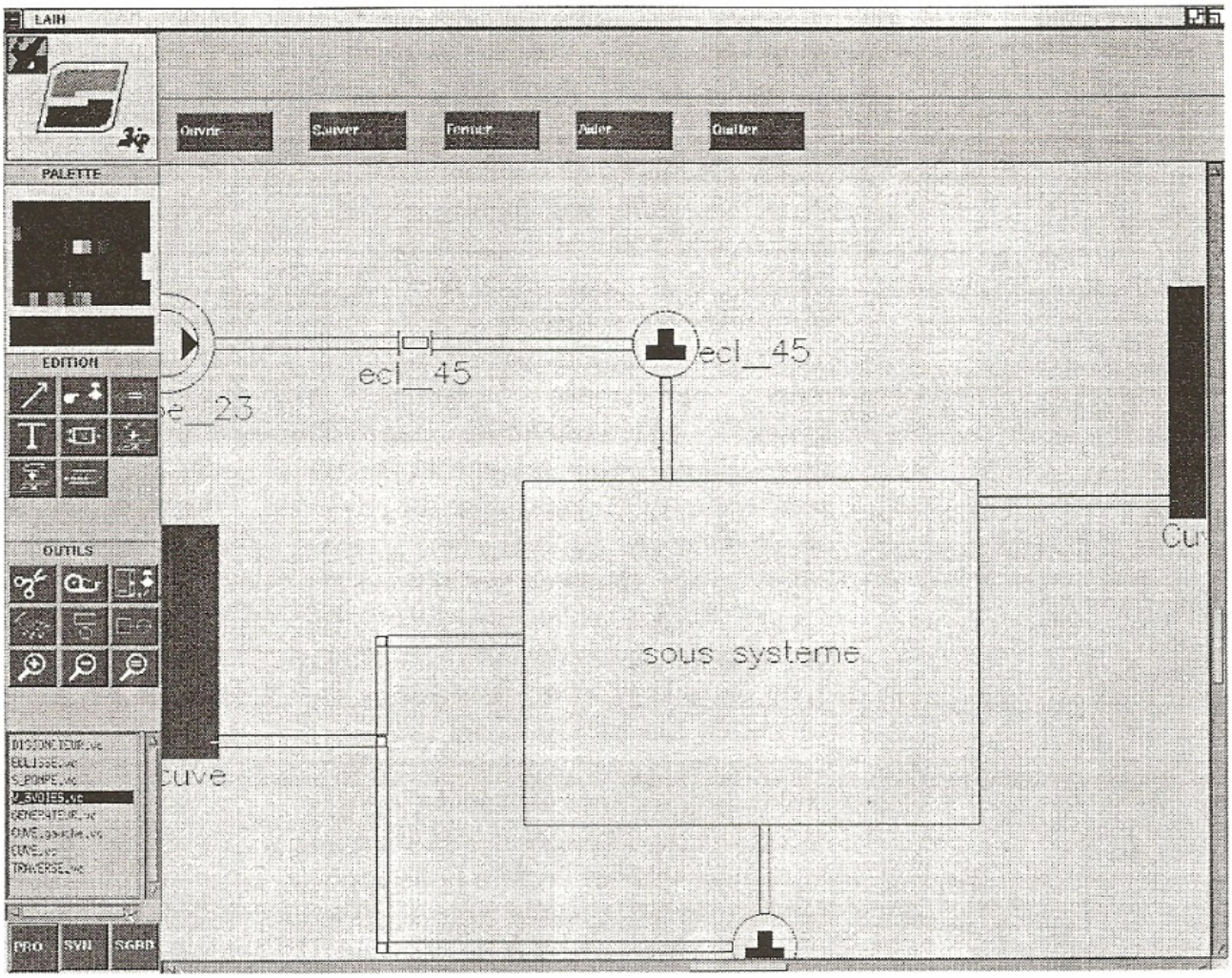

Figure 7 : Exemple de procédé en cours de description

- **Description du synoptique :** suite à la phase de définition rapide du prototype, le SPECIFICATEUR réalise les différentes images de supervision à l'aide d'un éditeur graphique dédié. Pour composer la partie des images appelée “synoptique”, partie correspondant à l'image physique du procédé, le SPECIFICATEUR copie sur les images de description du procédé les parties qui l'intéressent pour les “coller” sur les images de supervision. Puis, il complète ses images par des informations complémentaires (de type barre-graphe, compteur...). La figure 8 donne un exemple d'image en cours de réalisation.

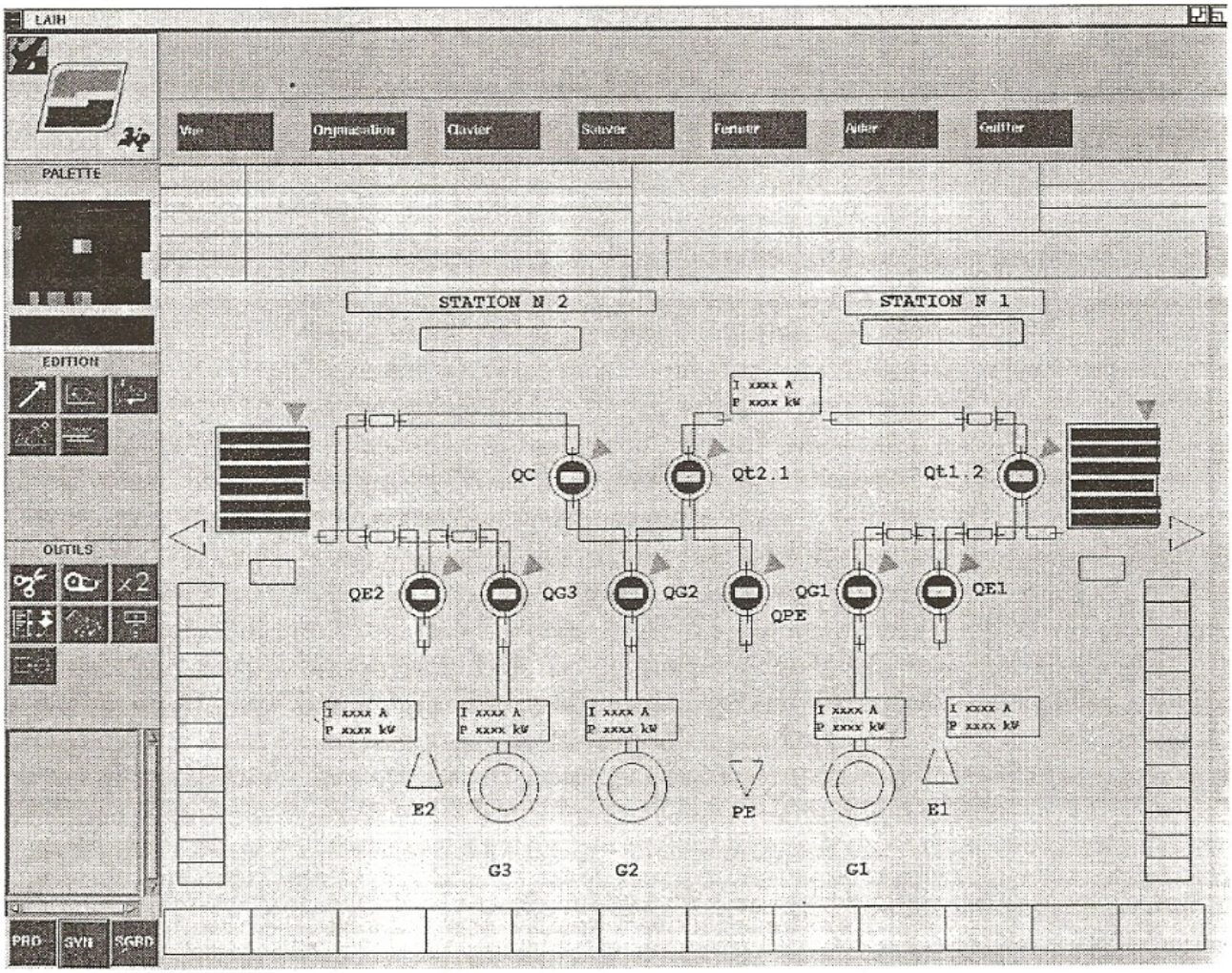

Figure 8 : Exemple d'image de supervision en cours de description

On retrouve sur cette page-écran la même structure que pour la description du procédé (zone de travail, menus d'édition d'outils et de commandes). Cependant, les différents menus ont des fonctionnalités différentes :

- Edition : il est possible de créer et manipuler des symboles appelés “informations complémentaires” (compteurs numériques, barre-graphes...) pour enrichir les images de supervision, copier des parties du réseau issues de la description du procédé, dessiner des fonds de plan.

- *Outils :* des fonction permettent de couper, copier, coller des symboles graphiques, d'accéder aux informations dans la base de données, d'aligner selon l'axe des X et l'axe des Y.

- *Commandes :* il est possible au Spécificateur de créer des vues, définir des organisations d'images (zones d'apparition des vues sur l'écran) et de spécifier les actions possibles pour les opérateurs (changement de vues).

- **Simulation :** une fois les différentes images créées, le spécificateur active dans un premier temps le générateur de code Sun. Puis, dans un second temps, il valide à l'aide du simulateur ses images en situation simulée.

Ainsi, le SPECIFICATEUR, expert du procédé, a à sa disposition un ensemble de moyens lui permettant rapidement et sans intermédiaire humain d'arriver à une première version de l'ensemble des images. Ces images sont alors prêtes pour une première phase d'évaluation avec les opérateurs humains, et en outre directement réutilisables par le DEVELOPPEUR.

III.4.3. Le développeur

Le DEVELOPPEUR est l'utilisateur qui réalise les applications. Cette réalisation s'effectue à partir du prototype réalisé sur la machine Sun par le spécificateur. A partir des données issues du prototypage et transmises par le module de récupération, le DEVELOPPEUR réalise l'application qui sera effectivement implantée sur le site. Il sera amené à spécifier les données suivantes :

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

- **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 (compteur numérique, barre-graphe...), (2) les blocs de configuration, comprenant l'ensemble des blocs de traitement logique, (3) les organisations d'images décrivant la structure physique des écrans de surveillance.

- **Les synoptiques :** il s'agit de la finalisation de la description des synoptiques et les informations procédé. Entre autres, le développeur devra définir l'ensemble des formes logique réelles des composants des images.

- **Les actions opérateurs :** il s'agit de la description des claviers fonctionnels et actions associées par ceux-ci, celles-ci correspondent pour la plupart du temps à

l'organisation des enchaînements des différents synoptiques.

La spécification de ces données se fait grâce à l'enchaînement d'un ensemble de pages-écrans dans lesquelles il s'agit de compléter les données manquantes. La figure 9 montre l'ensemble des menus permettant l'accès à ces différentes fonctionnalités, ces menus étant des points d'entrées à la base de données développeur (sous l'aspect de masques de saisie Oracle).

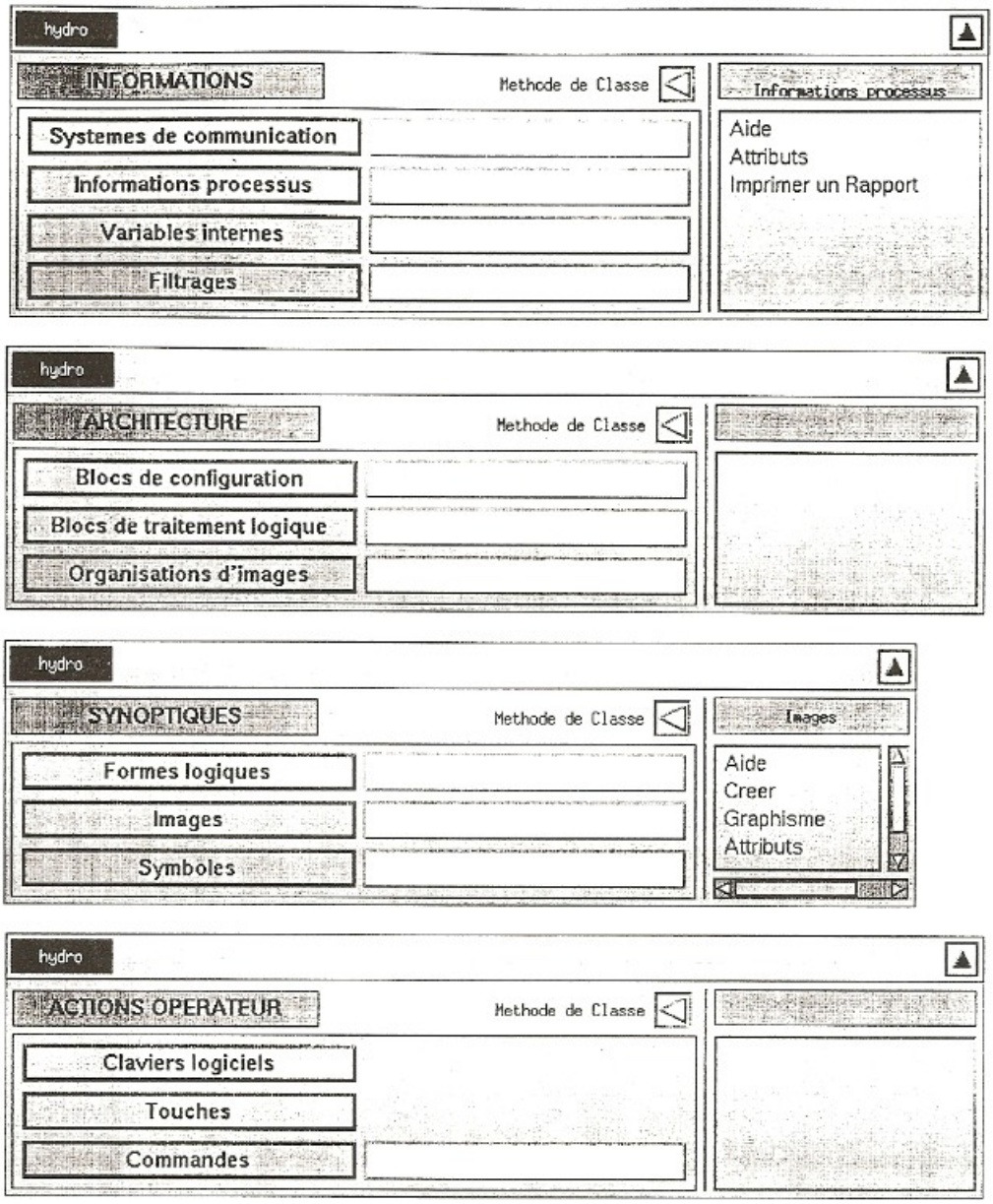

Figure 9 : Menus d'accès destinés au DEVELOPPEUR

L'ensemble de ces données peut être ensuite exploité par le générateur de code cible (voir la figure 2). Ce code est télé-chargé via un réseau ETHERNET et embarqué sur la machine cible. Le synoptique est alors prêt pour une phase d'évaluation sur le site ou en conditions simulées avec les opérateurs. Pour les aspects liés à son évaluation ergonomique, le lecteur pourra se référer à Millot [19], Abed [1], Senach [25, 26] ou Wilson [31].

IV. Conclusion

Dans cet article, nous avons décrit une méthode pour le développement de synoptiques industriels. Celle-ci intègre une approche de prototypage. Les six premières étapes de cette méthode, en l'occurrence la description du procédé, la définition rapide d'un prototype, la conception rapide d'un prototype, l'évaluation en situation simulée, la récupération des données issues du prototypage et le codage du logiciel de supervision sont actuellement informatisées sous la forme d'un atelier de génie logiciel appelé ATELIER-UTV (nom provisoire), visant à faciliter le travail coopératif entre les différents intervenants en leur fournissant un outil informatique commun. Cet atelier est actuellement dans sa phase terminale de développement.

Néanmoins, une première version de l'atelier est actuellement disponible, et utilisée par la Marine Nationale. Cette première version permet le développement complet des synoptiques (de la description textuelle du procédé à la génération de code cible, avec visualisation des images finales sur la machine cible), mais elle n'est pas encore dotée de l'interface de prototypage. Cet état de fait oblige par conséquent le développeur des synoptiques à créer pour l'instant complètement les bases de données à l'aide de masques de saisie Oracle, ainsi que les images grâce à l'éditeur DVdraw de DATAVIEWS. Le travail du développeur sera bien entendu considérablement facilité lorsque l'interface de prototypage sera intégrée, dans la mesure où celle-ci permet de réaliser rapidement et de valider les images de supervision, mais aussi de “remplir” automatiquement au fur et à mesure de la création des images une partie de la base de données (par exemple les connexions entre composants).

Remerciements :

Les auteurs remercient vivement l'ensemble des personnes ayant contribué à la réalisation de l'atelier, ainsi que Mr Jean-Claude Rault pour ses conseils avisés ayant permis d'améliorer cet article.

Références

[1] M. Abed : *Contribution à la modélisation de la tâche par des outils de spécification exploitant les mouvements oculaires, application à la conception et l'évaluation des interfaces homme-machine.* Thèse de Doctorat, Université de Valenciennes, septembre 1990.

[2] W.L. Bewley, J.L. Roberts, D. Schroit & V.L. Verplank : *Human factors testing in the design of XEROX'S 8010 "STAR" office workstation.* In Human Factors in computing systems-I, A. Janda (réd.), ACM, North-Holland, Amsterdam, pp. 72-77, 1983.

[3] S. Bodker & K. Gronbaek : *Cooperative prototyping experiments - users and designers envision a dental case record system.* In J. Bowers and S. Benford (réd.), Proceedings of the first EC-CSCW '89, UK, Computer Sciences Company, 1989.

[4] S. Bodker & K. Gronbaek : *Cooperative prototyping : users and designers in mutual activity.* International Journal of Man-Machine Studies, 34 (3), 453-478, mars 1991.

[5] J. Coutaz : *Interfaces homme-ordinateur, conception et réalisation.* Bordas, Paris, 1990.

[6] K. Crochart & D. Walfard : *Pratiques du maquettage-prototypage.* Génie Logiciel et Systèmes Experts, n°24, pp. 70-81, Editions EC2, septembre 1991.

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

[8] V. De Keyser & coll. : *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 1992.

[9] H. El Mrabet : *Outils de génération d'interfaces, état de l'art et classification.* Rapport technique, n° 126, INRIA-Rocquencourt, 78 pages, février 1991.

[10] E. Fadier : *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 (réd.), Edition Octarés, Marseille, 1990.

[11] P.A. Hancock & M.M. Chignell : *Intelligent interfaces : Theory, Research and Design.* North-Holland, 1989.

[12] R.D. Hill : *Supporting Concurrency, Communication and Synchronization in Human-Computer Interaction - the Sassafras SGIU.* ACM Transactions on Graphics, 5 (3), pp. 179-210, juillet 1986.

[13] P. Jaulent : *Génie logiciel : les méthodes*. Armand Colin, 1990.

[14] G. Johannsen : *Towards a new quality of automation in complex man-machine systems.* Automatica, vol. 28 (2), pp. 355-373, mars 1992.

[15] T. Jurain : *De l'écrit à l'écran, étude et classification des aides logicielles au développement d'interfaces graphiques.* Génie Logiciel et Systèmes Experts, n°24, pp. 28-42, Editions EC2, septembre 1991.

[16] D.K. Keyson & K.C. Parsons : *Designing the user interface using rapid prototyping*. Applied Ergonomics, 21(3), 207 - 211, septembre 1990.

[17] C. Kolski, M. Tendjaoui & P. Millot : *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, 1992.

[18] R. Mandiau, C. Kolski, P. Millot, B. Chaib-Draa : *A new approach for the cooperation between human(s) and assistance system(s) : a system based on intentional states.* Proceedings of World Congress on Expert Systems, Orlando, Florida, 16-19 décembre 1991.

[19] P. Millot : *Supervision des procédés automatisés et ergonomie.* Editions Hermès, Paris, 1988.

[20] P. Millot : *Coopération homme-machine : exemple de la téléopération.* Actes des Journées du GR Automatique, Strasbourg, 17-19 octobre 1990.

[21] T. Poulain & C. Kolski : *Monde réel et monde virtuel, la problématique du contrôle de procédés par un opérateur humain.* Actes de la Conférence Informatique 92 “L'interface des mondes réels et virtuels”, Montpellier, Editions EC2, 23-27 mars 1992.

[22] H. Pradenc : *Superviseurs : des fenêtres ouvertes sur le process.* Revue AXES Robotique-Automatique, 61, janvier 1992.

[23] J. Rasmussen : *Information processing and Human-Machine Interaction, an approach to cognitive engineering.* North Holland series in System Science and Engineering, 1986.

[24] D.L. Scapin, P. Reynard & A. Pollier : *La conception ergonomique d'interfaces : problèmes de méthode*. Rapport de recherche n° 957, INRIA, décembre 1988.

[25] B. Senach : *Evaluation ergonomique des interfaces Homme-machine : une revue de la littérature.* Rapport de recherche n° 1180, INRIA, Sophia Antipolis, 70 pages, mars 1990.

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

[27] T.B. Sheridan : *Task allocation and supervisory control.* In Handbook of Human-Computer Interaction, M. Helander (réd.), Elsevier Science Publishers B.V., North-Holland, 1988.

[28] I. Sommerville : *Le génie logiciel et ses applications.* Addison-Wesley, nouvelle édition et traduction, 1992.

[29] V. Taborin : *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, mars 1992.

[30] A. Van Daele : *L'écran de visualisation ou la communication verbale ? Analyse comparative de leur utilisation par des opérateurs de salle de contrôle en sidérurgie.* Le Travail Humain, tome 51, n°1, 1988.

[31] J.R. Wilson : *A Framework and a context for ergonomics methodology.* In Evaluation of Human works : a practical ergonomics methodology, J.R. Wilson et E.N. Corlett (réd.), Taylor & Francis, 1990.