THESE

présentée par

Gaëlle CALVARY

pour obtenir le titre de
DOCTEUR de L'UNIVERSITE JOSEPH-FOURIER-GRENOBLE I
(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)
Spécialité : **Informatique**

PROACTIVITE ET REACTIVITE : DE L'ASSIGNATION A LA COMPLEMENTARITE EN CONCEPTION ET EVALUATION D'INTERFACES HOMME-MACHINE

Date de soutenance : 2 Octobre 1998

Composition du jury :

Président :	Mr Jean Caelen
Directeur de thèse :	Mme Joëlle Coutaz
Rapporteurs :	Mme Jocelyne Nanard
	Mr Philippe Palanque
Examinateurs :	Mr Dominique Scapin
	Mr Jean-Luc Voirin

Thèse préparée au sein du laboratoire de Communication Langagière et Interaction Personne-Système
Fédération IMAG

Université Joseph Fourier - Grenoble I

RESUME : Cette thèse s'inscrit dans le cadre de l'Ingénierie des Interfaces Homme-Machine. Si elle traite du développement complet des systèmes interactifs, elle cible, de façon privilégiée, l'évaluation de l'interaction homme-machine. En matière de systèmes critiques, l'évaluation requiert en effet désormais des outils permettant de compenser les pressions industrielles croissantes en termes de coûts et de délais. Aujourd'hui, aucun outil commercialisé ne répond, de façon satisfaisante, à ces nouvelles exigences. Si le monde académique se mobilise, les propositions restent hélas de nature exclusivement proactive ou réactive : tandis qu'un outil proactif agit en phase de construction d'interfaces à des fins de qualité et d'efficacité, un outil réactif relève résolument de l'évaluation de l'utilisabilité. Cette thèse propose CatchIt, un environnement de développement, brisant ce paysage dichotomique par une polyvalence proactive et réactive.

CatchIt fonde sa complémentarité sur la capitalisation de descriptions métier. Ces descriptions consignent, pour un domaine applicatif donné, les connaissances métier mûries par l'expérience. CatchIt offre une structure d'accueil permettant de modéliser et pérenniser ces descriptions. Il leur reconnaît un statut normatif sur lequel il fonde ses prestations proactive et réactive. Sa proactivité relève du caractère prescriptif et réutilisable de ces descriptions. Il l'assortit d'un calcul de taux de code réutilisé. Sa réactivité consiste en une détection automatique des déviations entre l'application testée et sa norme métier. Cette confrontation requiert une connexion manuelle entre l'application et les concepts, tâches et stratégies opérateur de cette référence métier. CatchIt propose alors deux évaluations : une version prédictive mesure, dans l'application, les couverture et correction de la modélidation métier, l'observabilité, la multiplicité de la représentation et la curabilité. Une version expérimentale évalue l'utilisabilité de l'interface proposée, ceci par comparaison des comportements opérateur prévu et effectif. CatchIt procède, pour ce faire, à une instrumentation automatique et transparente du code applicatif : sur la base des liens sémantiques tissés entre l'application et sa norme métier, CatchIt identifie dans l'application les méthodes déterminant le contexte d'interaction. Il y insère des instructions espion propageant, vers le modèle normatif, les événements applicatifs ainsi capturés. L'instanciation de ce modèle définit le contexte d'interaction. CatchIt contrôle, dans ce contexte, les correction, complétude et concision de comportements d'opérateurs experts du domaine considéré.

MOTS-CLEFS : Utilisabilité, évaluation, critique, explication, modèle métier, capitalisation, réutilisation, proactivité, réactivité, ingénierie de l'Interaction Homme-Machine.

ABSTRACT : This dissertation contributes to the software engineering domain of Human Computer Interaction. Although the manuscript concerns all aspects of the development process of interactive systems, contributions are primarily focused on usability testing. The evaluation of complex systems requires tools that fit increasingly difficult temporal and financial constraints. Unfortunately, current commercial products are too limited in scope to produce useful results. The research communities have developed tools which are either proactive or reactive. Proactive tools support the construction of high quality user interfaces, whereas reactive tools provide feedback about the usability of a system. None of the tools developed to date cover both the proactive and reactive approaches to the development of interactive systems. This dissertation proposes CatchIt, a computer-based environment that supports both proactivity and reactivity.

CatchIt is grounded on domain descriptions that model accumulated knowledge for a particular work domain. In CatchIt, domain knowledge serves two complementary purposes. As reusable code, it acts proactively for the quality of the new derived application. As a reference model, it behaves reactively by automatically detecting deviations of the new application from the norm. The application quality is measured proactively in terms of the coverage of reused code. Reactive evaluation requires the developer to explicitly specify links between the concepts, tasks, and strategies implemented in the new application with those of the reference model. From the link specifications, code instrumentation, i.e., the insertion of spy instructions, is performed automatically. Reactive evaluation is performed predictively or experimentally. Predictive evaluation is expressed in terms of metrics that measure link coverage, observability, recoverability, and representation multiplicity. The experimental evaluation consists in detecting discrepancies between the normative expected states and the effective interaction trajectory captured through the spy instructions, of a human expert using the new developed system. In addition to detecting deviations, CatchIt provides the designer with possible explanations.

KEYWORDS : Usability testing, critics, explanation, model-based description, reusability, proactivity, reactivity, software engineering, Human Computer Interaction.

A mes parents

Remerciements

Si les remerciements sont de coutume, ils sont vraiment des plus sincères. Constamment tiraillée entre une thèse ancrée à Grenoble et un travail prenant sur Brest, je n'aurais pu achever, ni même amorcer, ce projet sans le soutien, la confiance, l'attention critique et constructive de Joëlle et de Jean-Luc. Qu'ils en soient donc ici, et en tout premier lieu, vivement remerciés. Merci donc à :

- Joëlle Coutaz, Professeur à l'Université Joseph-Fourier de Grenoble, pour avoir immédiatement accepté de diriger cette thèse, malgré des conditions matérielles difficiles. Merci pour toute cette culture, cette expérience, cette sensibilité, communiquées avec tant de chaleur, d'intelligence et de pédagogie, dans un climat de confiance absolue. Merci enfin de m'avoir fait révêr à CHI ;

- Jean-Luc Voirin, Responsable des Etudes Logicielles à THOMSON-CSF Radars et Contre-Mesures Brest, pour avoir instantanément adhéré au projet. Merci de l'avoir investi de tant d'idées bouillonnantes, nées de cette curiosité frénétique. Merci enfin d'avoir accepté et défendu cette solution hybride de mi-temps, pourtant, en pratique, si difficilement compatible des contraintes d'affaire.

Tous mes remerciements s'adressent aussi à :

- Jocelyne Nanard, Philippe Palanque, Jean Caelen et Dominique Scapin émanant respectivement des Universités de Montpellier, Toulouse, Grenoble et Paris pour avoir honoré ce jury de leur présence ;

- Patrick Kijewski, Responsable du Service Logiciel à THOMSON-CSF Radars et Contre-Mesures Brest, pour avoir assuré toute la logistique inhérente au projet ;

- Patrick Hénin, Directeur de la Communication à THOMSON-CSF Radars et Contre-Mesures, pour avoir délivré l'autorisation de publication de ce mémoire ;

- Jacques Tisseau et Yvon Autret, respectivement Maîtres de Conférence à l'ENIB et l'UBO, pour la confiance qu'ils m'ont témoignée, en m'accordant des heures d'enseignement.

Sans oublier toute ma famille, mes amis et plus particulièrement :

- Hermance, Didier et mes parents pour leur patience sans limite et leurs encouragements quotidiens ;

- Ronan, mon frère, pour sa contribution bibliographique inhumaine sur la condition paradoxalement humaine ;

- Laurence, ma sœur, Franck, mon beau-frère, Marion, ma filleule et leur adorable Mimi ;

- ma grand-mère ;

- Patrick et Nina pour leurs soutien et dévouement permanents ;

- Laurence, Francis, François, Frédéric et Daniel de l'équipe ou ex-équipe IIHM du laboratoire CLIPS-IMAG ;

- Jean-Pierre de THOMSON-CSF Radars et Contre-Mesures Brest ;

- et enfin Michel, mon sauveur, le jour où le PC rendit l'âme ...

TABLE DES MATIERES

INTRODUCTION

Verba volant, scripta manent ...
Les paroles s’envolent, les écrits restent ...

Née d'une prise de recul par rapport à notre pratique industrielle en matière d'Ingénierie des Interfaces Homme-Machine (IHM), cette thèse vise à améliorer nos processus de conception et d'évaluation d'interfaces. Menée en partenariat entre THOMSON-CSF Radars et Contre-Mesures Brest (RCM) et l'équipe Ingénierie de l'Interaction Homme-Machine (IIHM) du laboratoire CLIPS-IMAG de Grenoble, elle repose sur une confrontation et un enrichissement mutuel entre expériences complémentaires. Elle cible l'utilisabilité, c'est-à-dire ce facteur, paradoxalement bien ancré dans les mentalités, mais systématiquement compressé.

Victime d'une conjoncture économique défavorable, l'utilisabilité souffre de son occurrence tardive dans les processus de développement. Non seulement minimisé, voire occulté, dans les devis, pour des raisons de concurrence, son budget sert bien souvent à compenser d'éventuels surcoûts survenus dans les étapes précédentes. Ainsi, l'utilisabilité reste-t-elle aujourd'hui bien plus un argument commercial qu'une réalité quantifiée. Et tel sera son tribut tant qu'aucune méthode, aucun outil ne rationalisera, de manière convaincante pour l'industrie, le savoir-faire aujourd'hui artisanal. C'est à ce destin, incompatible des convictions actuelles et de toute criticité applicative, que répond cette thèse.

1. Le sujet

Motivée par les applications critiques développées à THOMSON-CSF Radars et Contre-Mesures, cette thèse s'intéresse aux moyens permettant de garantir, à coûts et délais maîtrisés, une utilisabilité mesurée. Pour ces applications, susceptibles de mettre en péril des vies humaines, l'adéquation homme-machine, loin d'être un luxe, y est une stricte nécessité [Gould 88]. Et, de façon plus générale, 48% du code étant aujourd'hui, en moyenne, consacré aux interfaces [Myers 92], il est légitime d'octroyer à l'utilisabilité la considération et le budget idoines. Aussi rappelons-nous, dans un premier temps, la définition de ce concept et la difficulté que soulève son évaluation. Nous formulons alors nos objectifs.

1.1. L'utilisabilité

En un sens, l'utilisabilité n'est qu'une perspective restreinte de considérations plus générales relatives à l'acceptation d'un système[1] [Nielsen 93] : si l'acceptation d'un système se réfère à son adéquation vis-à-vis des besoins et exigences des utilisateurs et autres interlocuteurs, l'utilisabilité se limite à la facette pragmatique liée à l'utilisation concrète de ce système. Elle complète l'utilité pour caractériser, de façon plus générale, la serviabilité ou « usefulness » du système (Figure 1). La serviabilité se réfère à la faculté du système à permettre à l'utilisateur d'atteindre ses buts[2]. L'utilisabilité en qualifie les facilités d'apprentissage et d'appropriation, l'efficacité à l'usage, la robustesse aux erreurs et enfin, de façon plus subjective, la convivialité ou le plaisir concrètement ressenti à l'utilisation de ce dernier. Aussi, l'utilisabilité ne se limite pas à des critères de performance dans l'accomplissement de tâches. Elle se réfère, de façon plus générale, à la satisfaction de buts personnels et collectifs[3] [Gilmore 95].

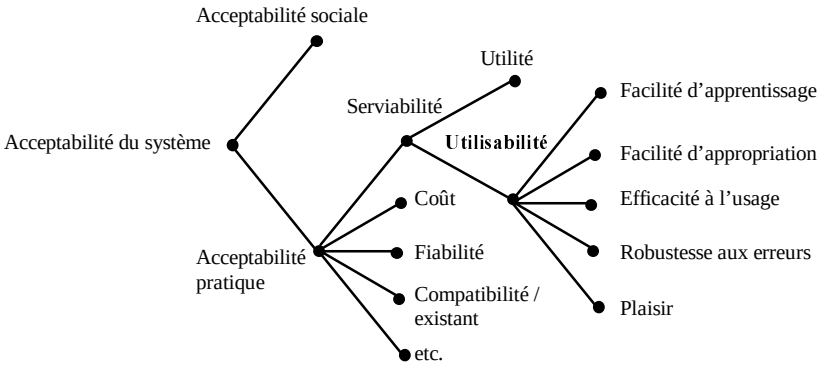

Figure 1 : Un modèle des attributs de l'acceptabilité d'un système. Traduit de [Nielsen 93] p 25.

[1] « To some extent, usability is a narrow concern compared to the larger issue of system acceptability, which basically is the question of whether the system is good enough to satisfy all the needs and requirements of the users and other potential stakeholders, such as the users's clients and managers » [Nielsen 93] p 24.

[2] « Usefulness is the issue of whether the system can be used to achieve some desired goals » [Nielsen 93] p 24.

[3] « Redefine usability as successful fulfillment of user and organisational goals, rather than rapid, error-free performance of unit-tasks » [Gilmore 95] p 177.

Si depuis 1977, l'utilisabilité est reconnue comme facteur à part entière en matière de qualité logicielle [McCall 77], force est de constater aujourd'hui que son évaluation reste artisanale dans le milieu informatique.

1.2. L'évaluation de l'utilisabilité

La subjectivité de certaines dimensions de l'utilisabilité, dont notamment le plaisir, contribue à renforcer un sentiment général de variabilité, de fugitivité. Elle confère finalement à l'utilisabilité ce caractère impalpable, si souvent invoqué, pour justifier l'absence d'évaluation. Des slogans apparaissent alors pour nous rappeler, au quotidien, l'enjeu, la nécessité, mais aussi la complexité de la tâche :

- C'est si facile d'être difficile mais tellement difficile d'être facile[4] !
- Si une fonction est introuvable, c'est comme si elle n'existait pas[5] !
- L'erreur est humaine, mais toujours reprochée[6] !
- L'intuition n'est pas de bon conseil[7].
- L'utilisateur a toujours raison[8].
- L'utilisateur n'a pas toujours raison[9].
- Les utilisateurs ne sont pas des concepteurs[10].
- Les concepteurs ne sont pas des utilisateurs[11].
- Les vice-présidents ne sont pas des utilisateurs[12].
- Moins, c'est plus[13] !
- Les détails importent[14] !
- L'aide n'en n'est pas une[15] !
- L'ingénierie de l'utilisabilité est un processus à part entière[16].

L'abondance et, parfois même, l'antinomie de ces slogans reflètent la difficulté que constitue l'évaluation de l'utilisabilité. Pourtant, les recherches s'activent dans le domaine et de cette ébullition jaillissent aujourd'hui méthodes [Ravden 89] et taxonomies [Coutaz 95b]. Mais, si ces propositions prouvent le caractère civilisable de la discipline, elles manquent encore de maturité. Elles restent notamment trop peu directives et surtout trop peu outillées pour conquérir l'industrie. Seules les techniques expérimentales à base de maquettage se sont réellement imposées. Mais leur contribution se limite à la construction des interfaces et toute évaluation leur échappe par ailleurs. Si le recours à des professionnels en la matière peut être une alternative au problème, le coût de prestation de ces experts s'avère hélas souvent incompatible du budget alloué. Aussi, ce palliatif reste-t-il marginal, certes approprié pour des projets de grande envergure, mais inadapté à des affaires de faible rayonnement.

Ainsi, en l'absence d'outils, l'évaluation reste-t-elle, dans la pratique industrielle, appréhendée de façon empirique et très artisanale : c'est concrètement le nombre d'itérations nécessaires à la finalisation du produit qui éponge cet amateurisme en la matière. Des versions successives sont soumises aux opérateurs : ils apprécient l'interaction proposée et rédigent d'éventuels faits techniques. L'intervention est alors banalisée en un traitement de bogue logicielle : elle fait l'objet d'une nouvelle version, d'une nouvelle livraison, d'un nouvel archivage. Si ces approches empiriques, à base de tâtonnements successifs, étaient salutaires en un temps, elles sont aujourd'hui insuffisantes. Les contraintes industrielles, en termes de coûts et de délais, s'avèrent, en effet, de plus

[4] « It's easy to be hard and hard to be easy » [INT'L 95]
[5] « If the user can't find it, the function's not there » [INT'L 95]
[6] « To Err is Human But they'll blame you » [INT'L 95]
[7] « Your Best Guess Is Not Good Enough » [Nielsen 93] p 10.
[8] « The User Is Always Right » [Nielsen 93] p 11.
[9] « The User Is Not Always Right » [Nielsen 93] p 11.
[10] « Users Are Not Designers » [Nielsen 93] p 12.
[11] « Designers Are Not Users » [Nielsen 93] p 12.
[12] « Vice Presidents Are Not Users » [Nielsen 93] p 12.
[13] « Less Is More » [Nielsen 93] p 15.
[14] « Details Matter » [Nielsen 93] p 15.
[15] « Help Doesn't » [Nielsen 93] p 16.
[16] « Usability Engineering is Process » [Nielsen 93] p 16.

en plus pressantes et abrègent inexorablement ces cycles d'évaluation. Il s'agit désormais de maîtriser ces itérations, non de les amputer, mais de les optimiser pour une convergence plus directe vers un produit de qualité.

L'ère de l'artisanat étant désormais révolue, seule l'adjonction d'un environnement de travail efficace permettra de réhabiliter ces approches itératives pour un développement performant d'interfaces de qualité. C'est à ce paradoxe que nous nous intéressons ici, à savoir, d'une part, la conscience collective de l'importance de l'utilisabilité, et, d'autre part, l'absence d'outils commercialisés permettant de l'évaluer, et encore moins de l'améliorer.

1.3. Nos objectifs

Cette thèse s'inscrit résolument dans une optique de rationalisation de l'évaluation. Elle cible l'utilisabilité par l'adjonction d'outils pour, à terme, garantir, et non usurper, à coûts et délais maîtrisés, un label d'utilisabilité. Des pressions économiques figurant parmi nos motivations, nous élargissons notre étude à une autocritique complète du processus de développement concrètement déployé. Il s'agit d'optimiser ce processus pour améliorer notre compétitivité et assurer, en pratique, à l'utilisabilité la place et le budget idoines. Ceci signifie identifier, au sein de ce processus, toute imperfection susceptible d'engendrer une quelconque inefficacité : loin de se limiter aux phases d'évaluation, ces puits financiers concernent aussi les étapes de conception et, de façon plus générale, les transitions bien trop franches entre phases.

A ces fins de rationalisation, nous proposons CatchIt, un environnement de développement, intégrant une évaluation automatique et transparente de l'interaction homme-machine. CatchIt puise ses principes dans la tendance industrielle à la spécialisation par domaine : résolument ancré dans les démarches qualité, il prône la capitalisation de connaissances métier pour une meilleure traçabilité des exigences et un support à l'évaluation de l'utilisabilité. Ainsi, motivée par des exigences applicatives largement aiguisées par un contexte économique défavorable, cette thèse s'enracine dans le contexte industriel. Elle établit une passerelle avec le monde académique pour un transfert d'idées, de méthodes, voire d'outils.

Le plan adopté pour ce mémoire retranscrit notre logique de raisonnement. Il débute sur cette prise de recul par rapport à notre pratique industrielle, examine ensuite les propositions émanant de la recherche, puis concrétise le transfert industriel par notre contribution logicielle CatchIt. Nous présentons cette organisation dans la section suivante.

2. Structure du mémoire

Cette thèse s'articule en trois parties. Si les deux premières s'apparentent à des états de l'art, respectivement industriel et académique, la dernière est consacrée à la présentation de notre contribution logicielle, CatchIt. De façon plus détaillée :

- la partie I présente puis critique la pratique industrielle actuelle. Elle s'articule en trois chapitres : le premier installe le décor en présentant le contexte applicatif ; le second décrit le savoir-faire aujourd'hui acquis ; le dernier identifie des axes de progrès ;
- à la lumière de ces exigences et en l'absence de solution commercialisée, la partie II examine les propositions qui émanent de la recherche. Tandis que le premier chapitre propose ESPRIT, un espace de classification des approches envisagées, le chapitre suivant procède à un positionnement de ces démarches au sein de cet espace. Le troisième chapitre consiste alors en une analyse critique des voies envisagées. Il dresse les fils directeurs de notre contribution logicielle CatchIt ;
- la partie III est dédiée à la présentation de cet environnement de développement. Le premier chapitre en détaille les principes ; le second la mise en œuvre ; le troisième l'illustre enfin sur une application simpliste mais représentative de nos exigences quotidiennes.

Nous concluons alors par une synthèse de notre contribution et une analyse critique de CatchIt : originalités, points forts et faiblesses y sont envisagés. Nous élargissons ensuite le cadre d'étude, par l'esquisse de perspectives tant techniques qu'applicatives.

PARTIE I : INGENIERIE DES IHM EN INDUSTRIE : ETAT DE L'ART ET AXES DE PROGRES

« C'est encore en méditant l'objet que le sujet a le plus de chance de s'approfondir » Bachelar

Cette partie est dédiée à une présentation de l'ingénierie des IHM en contexte industriel. Elle porte plus précisément sur les applications critiques développées à THOMSON-CSF RCM Brest. Elle s'articule en trois chapitres :

- le premier décrit le contexte applicatif, en termes de spécificités ;
- le second expose le savoir-faire aujourd'hui acquis ;
- le troisième présente enfin les axes de progrès, à l'origine de cette thèse.

Nous commençons par l'étude du contexte.

CHAPITRE I : CONTEXTE APPLICATIF

La tendance industrielle à la spécialisation par domaine permet de dresser une carte très précise des domaines d'activité de THOMSON-CSF Radars et Contre-Mesures (RCM) : il s'agit de concevoir et réaliser des équipements et systèmes dans les domaines radar, contre-mesure électronique (CME) et renseignement.

- Un équipement est une entité embarquée sur porteur marine ou aéronef. THOMSON-CSF RCM est spécialisé dans le domaine électromagnétique. Il fournit des capteurs et effecteurs visant respectivement à écouter ou modifier l'environnement électromagnétique du porteur. Parmi eux, les détecteurs de radar, les radars de pointe avant d'avions de combat ou les radars de patrouille maritime.
- Un système coordonne les différents équipements embarqués. Outre les équipements électromagnétiques, il inclut des équipements traitant d'autres domaines. Par exemple : des caméras de conduite de tir ou des détecteurs laser.

La réalisation de ces systèmes et équipements sollicite plusieurs domaines technologiques transversaux : le traitement du signal, la furtivité ou les matériaux avancés en sont des exemples. C'est bien entendu ici le logiciel, et plus précisément les IHM, qui retiennent notre attention. Après avoir cerné le cadre de l'étude dans la première section, nous soulignons les spécificités applicatives des IHM ainsi désignées.

1. Les IHM : cadre de l'étude

Avec les avancées de la technologie de l'information, les ordinateurs envahissent aujourd'hui le secteur de la défense [Goodman 96]. S'ils se limitaient jadis à de gros calculateurs autonomes, ils intègrent désormais l'opérateur dans les traitements. Les IHM deviennent ainsi le seul point d'entrée de l'homme dans des systèmes de plus en plus complexes [Boy 91] [Bares 96]. Aussi, leur étude constitue aujourd'hui une part d'activité croissante à THOMSON-CSF RCM.

Après en avoir décrit le spectre applicatif dans la première section, nous nous focalisons sur les IHM tactiques embarquées. Nous nous plaçons, à ces fins, dans le domaine de la Guerre Electronique Marine, secteur d'activité phare du centre de Brest.

1.1. Les IHM : spectre applicatif

La typologie matricielle la plus rudimentaire consiste à distinguer les interfaces de notre domaine selon deux critères : leurs caractères opérationnel et embarqué[17] (Figure 2). Tandis que le premier dénote les interfaces directement impliquées dans la réalisation d'une mission opérateur, le second regroupe celles installées à bord des bâtiments porteurs[18].

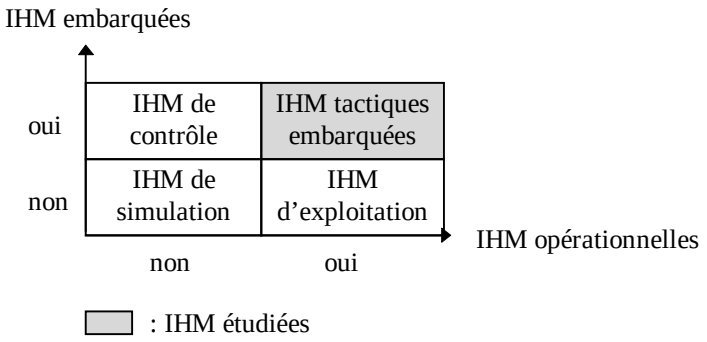

Figure 2 : Taxonomie matricielle des IHM développées à THOMSON-CSF RCM.

- Les interfaces de simulation sont des exemples d'IHM non opérationnelles, non embarquées. Souvent développées à des fins internes, elles permettent notamment de pallier l'indisponibilité de matériels ou logiciels. Notons qu'elles peuvent aussi revêtir le caractère opérationnel lorsque leur finalité consiste, par

[17] L'appellation « système embarqué » désigne l'ensemble des équipements électroniques et informatiques installés à bord d'un porteur (avion, hélicoptère, bateau, etc.). La navigation aérienne, la guerre électronique sont des exemples de domaines, civil et militaire, recourant à ce type de systèmes [Calvary 97b].
[18] Bateaux, avions, hélicoptères, etc.

exemple, en l'entraînement des opérateurs : elles constituent alors des IHM opérationnelles, non embarquées.
- Les outils d'exploitation constituent un autre exemple d'IHM opérationnelles, non embarquées. Ils ciblent essentiellement les préparation et dépouillement de mission. Ils facilitent, par exemple, l'exploitation d'enregistrements par la mise à disposition de fonctions graphiques, statistiques, etc.
- En aide au diagnostic de panne, des interfaces de contrôle peuvent être provisoirement embarquées : ce sont alors des IHM non opérationnelles, embarquées.
- Mentionnons enfin, pour illustrer les IHM opérationnelles, embarquées, les interfaces tactiques embarquées. Fleuron de notre technologie, elles conjuguent les contraintes et constituent ainsi un terrain d'étude privilégié. Nous leur consacrons la troisième section après une présentation préalable du domaine de la Guerre Electronique.

1.2. La Guerre Electronique

C'est la guerre du Golfe qui a couronné les principes de la Guerre Electronique (GE) : préconiser l'économie des armes dures[19] au profit d'un « jeu » tactique. D'une ampleur sans précédent, la GE a permis de dominer le conflit et a ainsi définitivement révolutionné les armées. Pour preuve, les F16 belges non encore équipés de moyens de contre-mesures, sont restés cloués au sol sur ce seul motif [Lidouren 90]. Une page de l'histoire est donc incontestablement tournée et un nouveau mouvement amorcé.

La Guerre Electronique est en fait desservie par son appellation. L'agressivité de ce label influence l'appréhension de ce domaine. Loin de se restreindre à des applications de défense ou d'attaque, la GE désigne l'ensemble des actions ayant pour but de [Orga 94] :

- tirer parti de l'utilisation adverse d'ondes électromagnétiques ;
- perturber l'utilisation adverse de ces mêmes ondes, voire de l'en empêcher ;
- se prémunir contre les actions précitées mises en œuvre par l'adversaire.

Se distinguent ainsi trois sous-domaines d'activité :

- la Mesure de Recherche Electronique (MRE) vise à détecter, intercepter, analyser, identifier les émissions adverses ;
- les Contre-Mesures Electroniques (CME) désignent les actions mises en œuvre pour empêcher ou réduire l'utilisation du spectre électromagnétique par l'adversaire ;
- les Mesures de Protection Electronique (MPE) visent à réduire la vulnérabilité des moyens engagés vis-à-vis des actions électromagnétiques adverses.

Les experts séparent la GE en deux domaines d'actions :

- l'offensive concerne les actions d'antidétection, d'antilocalisation et d'intrusion contre, d'une part, les moyens de transmission et de liaisons de données, et, d'autre part, les radars d'observation et de surveillance ;
- la défensive[20] concerne, quant à elle, les actions ayant pour objectif de se protéger contre les attaques. Dans cette quête de neutralisation, après une détection, localisation, reconnaissance, identification des menaces, un classement est établi selon la criticité de chacune d'elles. On peut alors jouer sur l'environnement, notamment par substitution ou dissimulation. La substitution consiste à placer de fausses cibles, de signature adaptée. La dissimulation vise, quant à elle, à masquer la cible. Pour des raisons de confidentialité, nous ne détaillerons pas plus ces réactions. Précisons, par contre, qu'elles peuvent être manuelles ou semi-automatiques, c'est-à-dire toujours placées sous le contrôle de l'opérateur. D'où l'enjeu des IHM tactiques embarquées.

1.3. Les IHM tactiques embarquées

Nous nous plaçons ici dans le domaine de la Guerre Electronique Marine (GEM). Nous décrivons, dans ce cadre, le rôle des différents opérateurs puis étudions, de façon privilégiée, le poste de l'opérateur dit tactique.

---

[19] Canons, missiles, etc.
[20] Défensive ou autoprotection.

1.3.1. Rôles opérateur

Dans un local dit centre opérations (Figure 3), cohabitent plusieurs opérateurs chargés de la mission GE :

- l'opérateur tactique gère une vue globale de la situation environnante. Il pilote les équipements dépendants et peut solliciter, si nécessaire, les opérateurs affectés à ces équipements ;
- un opérateur est, en principe, réquisitionné par équipement embarqué. Conformément à la tactique décidée, il configure cet équipement et déploie l'action commanditée. Notons que, dans certains systèmes, ces interventions sont automatisées dès lors que la tactique est pilotée de l'interface opérateur tactique ;
- enfin, un opérateur dit GE assume une mission à caractère d'alerte et de renseignement. Affecté aux détecteurs, il enrichit des bibliothèques d'informations par corrélation de données capturées.

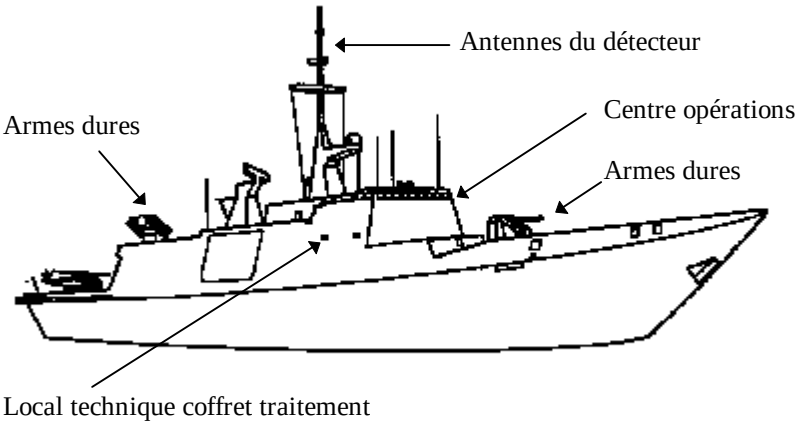

Figure 3: Frégate équipée d'un système d'autoprotection.

Si plusieurs opérateurs concourent ainsi au déroulement d'une mission, le commandant de bord reste seul maître à bord. Certaines actions lui sont strictement réservées : c'est le cas de tactiques bien précises, mais confidentielles, dont le déploiement reste soumis à son accord. Souvent mentionnées sur l'interface présentée à l'opérateur tactique, ces actions sont, en pratique, protégées par un code d'accès. Si nécessaire, le commandant rejoint ce poste opérateur et déploie la tactique décidée.

Si la tendance actuelle est à médiatiser les collaborations entre opérateurs, ces coopérations restent aujourd'hui à dominante sociale : la communication s'effectue, en général, par interphone, avec, tout au plus, un intermarquage à l'appui[21].

C'est l'opérateur tactique qui retient ici notre attention. S'il en réfère certes au commandant de bord, il reste un décideur clé et assume de fortes responsabilités. Sa perception de la situation est essentielle pour des réactions justes et optimisées.

1.3.2. Opérateur tactique

L'opérateur tactique travaille sur une représentation de la situation environnante. Cette situation reflète la carte des acteurs présents sur le « terrain » : bâtiments voisins, missiles, ou tout autre élément matériel détecté par les équipements connectés. Parmi ces derniers, les détecteurs qui analysent la nature, les paramètres physiques et l'évolution temporelle des signaux émis par les radars ainsi détectés. Les détecteurs tentent alors une identification par consultation de bibliothèques prédéfinies. Ils transmettent, au calculateur distant, les plots après habillage d'informations de détection et d'identification (Figure 4).

[21] Désignation croisée d'objectifs.

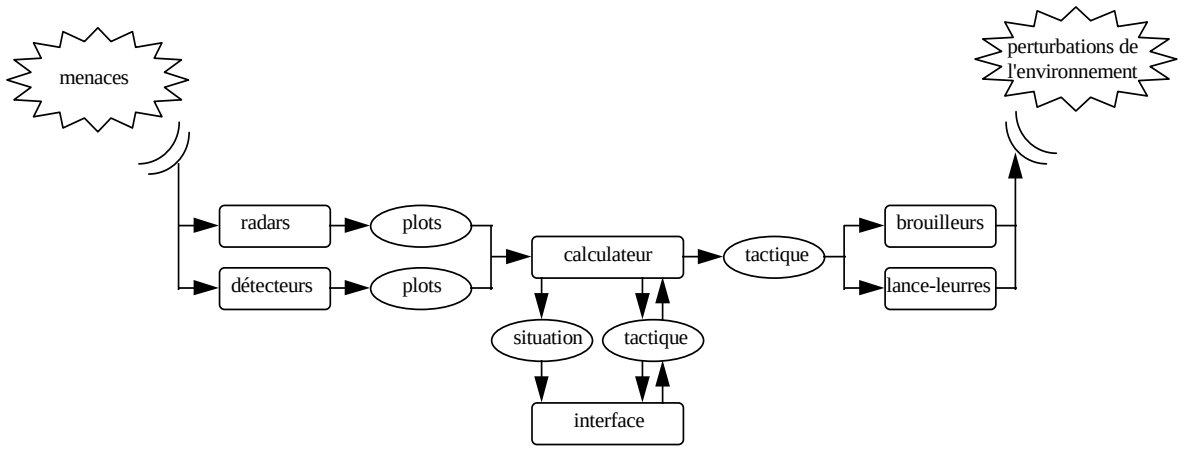

Figure 4 : Traitements en autoprotection.

Le calculateur corrèle, sous forme de pistes, les plots reçus des différents équipements. Il élabore une situation, préconise éventuellement une tactique et transmet ces informations à l'opérateur. Les pistes sont classiquement visualisées sous la forme d'une carte panoramique (Figure 5). Cette carte, souvent centrée porteur, explicite, outre les positions relatives des différentes pistes, des informations prédominantes telles que leur caractère ami/ennemi, leur origine de détection, etc. Des icônes sont souvent utilisées pour un codage condensé de ces informations. Bien entendu, ces descriptions synthétiques restent insuffisantes pour une prise de décision. Pour l'obtention d'informations détaillées, l'opérateur prend les pistes d'intérêt en « contrôle direct ». Une zone dédiée, ou « aire de renseignement opérateur » (ARO), visualise alors, sous une forme détaillée, les paramètres de ces pistes (Figure 5). Les valeurs issues des différents capteurs y sont notamment affichées. Notons que cette prise en contrôle direct s'effectue aujourd'hui par manipulation directe : l'opérateur sélectionne interactivement la piste d'intérêt en zone panoramique. Cette sélection s'effectue via un index ou la boule roulante.

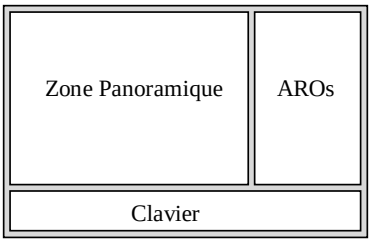

Figure 5 : Représentation typique d'une situation tactique.

Selon son appréciation de la situation, l'opérateur peut envisager diverses stratégies : à lui d'opter pour la plus adaptée, voire les combiner judicieusement, en fonction des équipements disponibles. L'opérateur transmet alors au calculateur la tactique retenue. Ce dernier en assure ensuite la mise en œuvre : par exemple, perturber l'environnement par les brouilleurs ou lance-leurres connectés (Figure 4).

Ainsi, l'expressivité de la situation est-elle fondamentale. Elle doit refléter le danger potentiel et susciter, chez l'opérateur, des réactions justes et optimisées. Aucun retour d'efficacité n'existant aujourd'hui, seule l'interprétation de cette situation permet, en retour, d'estimer le succès ou l'échec de la tactique déployée. Si impasse il y a, les armes dures restent bien entendu envisageables, mais le nerf de la GE consiste, rappelons-le, à les économiser au profit d'un jeu tactique.

Notons que si ce schéma s'inspire d'un temps de guerre, la philosophie est drastiquement différente en climat pacifique : les missions revêtent alors un caractère d'entraînement qui atténue quelques spécificités de ces applications. Mais ce sont bien les conditions extrêmes qui nous intéressent ici.

2. Les IHM : spécificités

Nous adoptons ici le point de vue de l'opérateur et recensons, sous cet angle, les particularités des IHM tactiques embarquées. Pour de plus amples détails sur les caractéristiques anthropocentrées mentionnées, nous invitons le lecteur à se référer à l'annexe A.

2.1. Criticité applicative

La spécificité fondamentale de nos applications réside dans leur intrinsèque criticité : c'est non seulement la perception de la menace et des réactions envisageables que la mise en œuvre rapide et efficace de la tactique retenue qui conditionne la survie de l'équipage. Si des systèmes de surveillance sont mobilisés pour anticiper les menaces, par détection et trajectographie par exemple [Lidouren 90], le préavis reste bref et c'est bien souvent dans l'urgence que les opérateurs réagissent. Dans le cas d'un engagement par exemple, les temps de réaction accordés restent inférieurs à la dizaine de secondes. Il est donc vital que les opérateurs disposent, à tout instant, de leur plus fort potentiel en termes de concentration, de réflexion et de décision.

En la matière, la théorie des ressources postule l'existence d'un capital attentionnel limité [Reason 93]. Cette théorie rejoint le principe dit du « canal unique » qui stipule que, dès lors que l'attention est sollicitée, les informations transitent par une unité centrale à capacité limitée [Paty 95]. A partager entre toutes les opérations mentales, cette réserve s'accomode d'activités monotâche. Elle souffre, en revanche, de sources concurrentes, de l'habitude et du changement [Reason 93]. Or, nos applications, comme tous systèmes dynamiques ouverts, sont, par essence, sujettes à des perturbations de cet ordre. Des transitions brutales peuvent notamment s'opérer entre situations de veille et de crise. Elles nécessitent alors des traitements d'urgence et une mobilisation instantanée des ressources opérateur.

Si ces irrégularités de rythme affectent le capital attentionnel de l'opérateur, l'environnement le dégrade par ailleurs [Wisner 90] : l'espace est restreint, les postes de travail nombreux, une ambiance sonore, lumineuse et thermique en résulte [Calvary 97b]. Or, dans une activité à fortes contraintes, un environnement agressif s'avère néfaste [Wisner 90].

Si divers facteurs se conjuguent ainsi pour accroître l'intrinsèque criticité de nos applications, la littérature rapporte très peu d'études sur la combinaison et l'intrication de ces facteurs [Giry 95]. La prise de conscience collective du terrible coût de l'erreur humaine est heureusement aujourd'hui à la source d'un nouvel élan et d'un intérêt renouvelé [Reason 93]. L'Europe s'y attaque aujourd'hui. Pour preuve, le projet Romi (Robust Human-Machine Interaction), lancé il y a trois ans sur financement du ministère européen de la Recherche. Piloté par Eurisco, Romi s'achève cette année après avoir abordé, dans un contexte aéronautique, différents aspects du problème de l'interaction homme-machine : méthodologies de conception, évaluation, aide à la décision, ergonomie, modèles et mesures de fiabilité, etc. [Tardif 96]. La mission Cassiopée en est un autre exemple [Charles 96] : elle s'est déroulée en août 1996 avec la spationaute Claudie André-Deshays. A bord de la station russe Mir, elle a participé à un programme scientifique de cinq expériences, ayant nécessité trois années de travail préparatif. Impliqués dès le départ, les facteurs humains ont dirigé l'étude : des analyses fonctionnelle et opérationnelle menées en parallèle ont permis d'identifier très tôt leurs interactions. Cette démarche a prouvé que la prise en compte de l'homme lors de l'utilisation opérationnelle peut avoir un impact sur les fonctions du système.

2.2. Poids et subjectivité des facteurs humains

« Dire que l'homme est un composé de force et de faiblesse, de lumière et d'aveuglement, de petitesse et de grandeur, ce n'est pas lui faire son procès, c'est le définir » disait Diderot. Comment mieux exprimer cette complexité et cette diversité humaines ? La Rochefoucauld ajoutait « il est plus aisé de connaître l'homme en général, que de connaître l'homme en particulier ». Aussi, est-ce bien de l'homme en général dont nous allons traiter maintenant. « Ni bon ni méchant. Il naît avec des instincts et des aptitudes » disait Balzac. Et ce sont ses caractéristiques que nous souhaitons ici mieux cerner. L'étude est évidemment biaisée : « Sujet observant, le sujet est organe de la connaissance et objet de son étude » [Paty 95].

Un consensus émerge aujourd'hui pour admettre que « ce qui est important pour l'homme, ce n'est pas tant ce qui lui arrive mais ce qu'il pense qu'il lui arrive ». Si Schopenhauer l'affirmait déjà à l'époque, la définition de la charge de travail l'intègre désormais de façon officielle : la charge de travail quantifie l'écart qui peut exister entre « les exigences du travail et la capacité disponible de l'opérateur pour y faire face » [Spérandio 92]. C'est

bien de la différence entre l'estimation de cette complexité et des ressources mentales disponibles en termes d'intelligence, de capacité perceptive, mnésique, etc. dont il s'agit [Doppler 94].

Le problème s'exprime donc en termes de sentiments, et plus précisément, du sentiment d'une parfaite maîtrise de la situation. Cette confiance est un élément déterminant dans le comportement opérateur : il lui faut l'acquérir pour toujours garantir un contrôle efficace de l'événement. Elle le préserve du stress, cette notion subjective dont les réactions sont complexes. [Smelick 79] (cité dans [Mallion 82]) en compare la diversité à l'image d'un sablier : "un spectre large de stimuli converge vers un goulot commun et s'élargit ensuite sous la forme de multiples réactions adaptatives qui peuvent être modifiées par des agents particuliers, dits "conditionnants" : principalement, le stimulus, mais aussi l'organisation mentale de l'individu." Parfois paralysé, il est en tout cas obnubilé par l'urgence soudaine et enclin aux réflexes. Les réflexes sont des réactions très rapides qui anticipent toute réflexion [Larousse 94] et économisent le potentiel attentionnel humain limité. Ils s'inscrivent dans le cadre de la structuration cérébrale par niveaux d'âge : tandis qu'un stade archaïque offrirait des fonctions intégrées, les degrés les plus récents seraient de plus en plus analytiques et différenciés. L'évolution n'entraînerait aucune régression des structures plus anciennes mais seulement leur masquage ou leur inhibition. Ainsi coexisteraient plusieurs modes de fonctionnement d'âges différents. Les modes plus anciens seraient susceptibles de se manifester, non seulement dans la pathologie, mais également dans certaines situations d'urgence ou de saturation des niveaux plus élevés (réaction émotionnelle lorsqu'un geste fin est indisponible, gesticulation désordonnée, sursaut en situation de danger) [Paty 95]. C'est dans cette lignée que s'inscrit le modèle de Rasmussen [Rasmussen 86] qui propose une structuration tripartite en automatismes, règles et connaissances.

Ainsi, la solution semble-t-elle s'exprimer aujourd'hui en termes d'expertise opérateur : il s'agit de « compiler », au niveau réflexe, le comportement opérateur pour, d'une part, supprimer l'effet potentiellement désastreux de ces réminiscences et, d'autre part, réserver le capital attentionnel humain aux seules situations d'urgence.

2.3. Savoir-faire opérateur

L'expertise opérateur est aujourd'hui cultivée par programmes de simulation et d'entraînement. Il s'agit de ramener, au strict niveau opératoire, le plus de stratégies contrôlées et sérielles, afin de satisfaire les rationalités limitée, paresseuse et réticente [Reason 93]. Ces rationalités incitent les sujets humains à « agir par reconnaissance de configurations spécifiques au contexte, plutôt que de calculer ou d'optimiser » [Reason 93] :

- elles les conduisent à une confiance excessive dans des indices apparemment familiers pour l'application aisée de solutions bien connues ;
- elles les dirigent inlassablement vers les sentiers battus plutôt que de favoriser l'exploration de nouvelles voies potentiellement profitables ;
- elles les amènent à accorder une confiance excessive à la justesse de leurs connaissances et les complaisent dans l'observation des données validant leur choix, négligeant tout signe discordant.

Mais attention : « à celui qui n'a qu'un marteau, tout problème apparaît comme un clou ! » [Reason 93]. Aussi, l'entraînement doit-il être intensif et exhaustif. C'est le cas dans nos applications de défense dont la longévité est compatible de cet apprentissage. Les systèmes sont typiquement développés pour des durées allant de 10 à 20 ans [Boudigou 95]. Si le coût de ces systèmes s'oppose à leur renouvellement trop fréquent, la nécessaire appropriation de l'interface par les opérateurs milite aussi en faveur de ce conservatisme : les opérateurs doivent en être suffisamment familiers pour une prestation efficace, même en contexte défavorable. Ainsi, au fil du temps, un certain savoir-faire se cultive, se renforce. Les opérateurs s'en imprègnent et l'expérience montre alors le danger d'une transgression intempestive. D'où un certain paradoxe, à savoir, d'une part, respecter le savoir-faire opérateur, d'autre part, mettre à profit les avancées technologiques.

2.4. Complexité croissante

Jadis de simples consoles de contrôle de fonctionnement, les IHM sont désormais le seul point d'entrée de l'homme dans des systèmes de plus en plus complexes [Boy 91] [Bares 96]. Les progrès en électronique, informatique, miniaturisation se concrétisent par des équipements de plus en plus perfectionnés, sophistiqués et intelligents. Il nous faut désormais disposer d'interfaces capables de traiter et de « véhiculer des messages de plus en plus riches, de plus en plus denses, sans augmenter pour autant la charge de travail globale » [Tardif 96].

Demain, les avancées technologiques nous permettront de médiatiser les collaborations humaines, restées jusqu'ici sociales. Comment alors exploiter cette ascension technologique et préserver le savoir-faire opérateur ?

Il ne s'agit, en effet, pas de camper sur ses positions et de défendre des technologies dépassées sous prétexte d'une parfaite maîtrise. Les IHM étant, par nature, la vitrine technologique d'une entreprise, l'innovation en la matière constitue un argument de vente. Elles exportent le dynamisme, la vivacité, le savoir-faire, voire la culture, d'une entreprise et sont vitales pour s'imposer face à une concurrence acharnée. Comment alors concilier cette ascencion technologique et le savoir-faire opérateur, entre temps, devenu quasiment immuable ?

Des compromis sont, en pratique, adoptés pour atteindre un juste équilibre entre conservation et innovation : il s'agit de proposer sans imposer, d'innover sans révolutionner, mais d'innover impérativement. C'est ainsi que des grandeurs, aujourd'hui obsolètes, sont exclusivement maintenues pour le confort et la sérénité des opérateurs. Parmi elles, la rémanence des échos radar liée aux anciens tubes ou l'axe de balayage de l'antenne, désormais électronique [Boudigou 95]. Les opérateurs s'étant développé des heuristiques, fondées sur la représentation graphique de ces informations, il s'est avéré délicat de les supprimer brutalement. Elles ont ainsi été recréées artificiellement dans l'unique objectif de préserver le savoir-faire opérateur.

Si ces compromis anthropocentrés sont salutaires, toute solution technocentrée est, en revanche, bannie en matière de systèmes critiques. Pourtant, en pratique, la tentation est forte d'imposer aux opérateurs les modèles des systèmes connectés : la non-concordance et les contraintes de chacun d'eux compliquent, en effet, l'intégration système et induisent de fortes exigences en termes de simulation. S'il faut lutter contre ces compromis simplificateurs, il faut parallèlement maîtriser la complexité technologique : il convient notamment de masquer aux opérateurs cette complexité sous-jacente pour préserver leur capital attentionnel.

Les caractéristiques de la tâche interfèrent, en effet, de façon conséquente avec son coût physiologique et psychique. Tandis que les tâches élémentaires très simples, détection d'un point lumineux ou d'un bip sonore par exemple, semblent relativement résistantes à la fatigue, les tâches complexes nécessitant une sélection ou une interprétation d'informations s'avèrent nettement plus sensibles. La présentation de l'information joue alors un rôle essentiel, tant en intensité, durée, fréquence que répartition spatiale : pour chacune de ces caractéristiques, il s'agit de trouver le juste compromis pour optimiser l'effort de détection et de traitement ainsi que la surcharge cognitive.

Ainsi, « la conception d'un avion, d'un hélicoptère, d'une navette spatiale et même d'un centre de contrôle du trafic aérien se heurte désormais à des problèmes d'un nouveau type : ceux liés aux facteurs humains » [Tardif 96]. La consigne est alors claire : toujours laisser la technique au service de l'homme, quitte à ternir quelques prouesses technologiques.

3. Synthèse

Cette thèse s'inscrit dans une démarche de progrès. Elle vise à améliorer nos technicité et compétitivité, dans un respect du savoir-faire opérateur. L'autocritique ne pouvant se faire sans une appréhension fine du contexte applicatif, ce premier chapitre en propose une revue. Il est structuré en deux volets :

- le premier délimite le contexte de l'étude. Il choisit les IHM tactiques embarquées comme support à la réflexion ;
- le second justifie ce choix en soulignant les spécificités intrinsèques de ces applications. Outre leur criticité, il mentionne le poids et la subjectivité des facteurs humains, l'existence d'un savoir-faire opérateur quasiment immuable et l'inexorable tendance à la complexification technologique.

L'antinomie de ces deux derniers points laisse présager de la difficulté de la tâche. Elle est, en pratique, considérablement aiguisée par une conjoncture économique défavorable. Des pressions industrielles, en termes de coûts et de délais, allourdissent, en effet, inexorablement le tribut des informaticiens (Figure 6). Une concurrence acharnée, des marchés de moins en moins protégés délimitent désormais un terrain d'actions de plus en plus étroit. Or, s'y conformer est vital pour des raisons de crédibilité et compétitivité.

L'informaticien ne déroge pas à la règle : il doit respecter ses engagements techniques, financiers et temporels. Et ceci d'autant plus que l'envergure des affaires justifie souvent la collaboration de plusieurs équipes dont le

moindre retard peut enrayer la gestion du projet. Comment alors, en pratique, s'accomoder de contraintes aussi discordantes ? C'est l'objet du chapitre suivant : il traite de la pratique industrielle.

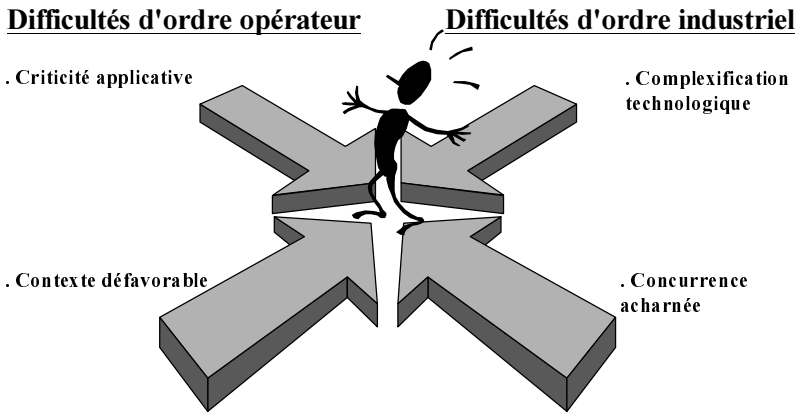

Figure 6 : Pressions subies par les équipes de développement : les exigences applicatives et industrielles se combinent pour allourdir inexorablement le tribut des informaticiens.

CHAPITRE II : PRATIQUE INDUSTRIELLE

Tandis que le spectre technologique s'étend de façon exponentielle, les degrés de liberté accordés aux équipes techniques s'amenuisent progressivement. Il s'agit désormais d'innover, certes, mais toujours dans un respect rigoureux des coûts et des délais. A ces exigences industrielles, s'ajoutent des spécificités applicatives des plus contraignantes. Parmi elles, le respect d'un certain savoir-faire opérateur, quasiment immuable et essentiel pour ces applications critiques. Les facteurs humains y sont ainsi omni-présents et conditionnent la pratique.

Nous consacrons ce chapitre à la description de cette pratique industrielle. Après en avoir présenté les constats, nous évoquons les méthodes et outils concrètement déployés pour la réalisation de ces applications.

1. Constats

Bien que l'opérateur soit au cœur de nos préoccupations, les spécifications restent insuffisantes en la matière. Evidentes pour certains, inconnues pour d'autres, les habitudes, exigences et limites opérateur sont très rarement spécifiées dans les documents transmis aux équipes logicielles. La dimension de fonctionnement y prédomine systématiquement et occulte les informations de l'ordre de l'utilisation. Seule une double-compétence, domaine opérationnel d'une part, logiciel d'autre part, permet de compenser ces lacunes. Mais le renouvellement du personnel s'interpose bien souvent à cette capitalisation de connaissances. Aussi, la qualité insuffisante des spécifications constitue-t-elle aujourd'hui un obstacle majeur au déroulement linéaire et serein du processus de développement.

1.1. Imperfection des spécifications

Si les principales lacunes en matière de spécification concernent les exigences d'utilisabilité, il est intéressant de noter qu'elles sont pourtant connues et parfois invoquées en phases amont de spécification système. La difficulté essentielle provient en fait de l'ampleur des projets. Bien souvent multi-industriels, presque toujours multisites, ils nécessitent le concours de nombreuses équipes dont la collaboration reste souvent tacite et peu formalisée. Etrangères les unes aux autres, géographiquement dispersées, de compétences complémentaires, ces équipes mobilisent leurs connaissances, leur expérience au sein d'échanges fructueux certes, mais hélas fugitifs. En résulte une perte d'informations notable qui se répercute par des spécifications de qualité insuffisante. C'est en termes de non-complétude, voire de non-correction, que s'expriment ces griefs. La métaphore de la cafetière en illustre le principe (Figure 7) :

- des connaissances métier, tel le café moulu, de granularité variable selon la qualité de la torréfaction, sont invoquées en phases de spécification système ;
- infusées au cours de discussions techniques, elles sont ensuite, tel le marc de café, définitivement filtrées ;
- seules les spécifications, c'est-à-dire le café obtenu, sont ensuite transmises aux équipes de développement.

A travers cette métaphore, apparaissent quelques facteurs influents dans la qualité de la boisson obtenue. Selon les conditions de réalisation, l'essence sera en effet plus ou moins consommée : la qualité du café, c'est-à-dire l'expertise des intervenants, la chaleur de l'eau et la finesse du filtre, c'est-à-dire les conditions matérielles de réunion, etc., sont autant de paramètres qui vont influencer la qualité du café consommable, c'est-à-dire des descriptions produites.

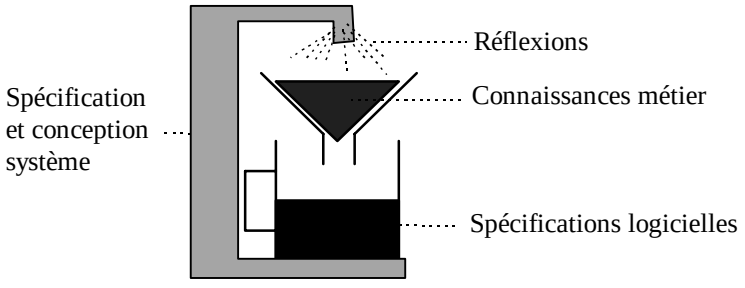

Figure 7 : Perte d'informations en phases de spécification et conception système : tels la poudre de café puis le marc de café, les connaissances originelles, invoquées en phases de spécification et conception système, permettent de générer des spécifications avant d'être définitivement filtrées. Seule la boisson ainsi obtenue est servie aux équipes logicielles, mais sa qualité est largement influencée par l'expertise des intervenants, leur sensibilité et les conditions de réunion.

Pour contrer ces insuffisances et forcer les non-dits, les équipes logicielles déploient, en pratique, des démarches itératives.

1.2. Démarches itératives

Les approches itératives à base de maquettage et de prototypage trouvent toute leur quintessence dans l'industrie de ces systèmes. Elles permettent, par tâtonnements successifs, de converger vers des spécifications externes, compatibles avec les exigences opérateur. Elles militent pour une évaluation non plus sommative mais formative [Hix 93] et protègent ainsi les développeurs d'un verdict terminal, à allure de sanction. Elles sont donc particulièrement adaptées à l'ingénierie de nos applications dont les spécifications incomplètes compromettent un développement logiciel déterministe et linéaire. Si la littérature rapporte l'efficacité des conceptions participatives en termes de détection précoce de problèmes d'utilisabilité, elle souligne aussi une plus forte adhésion du client par cette implication centrale [Hix 93] [Poltrock 95].

Si ces pratiques itératives sont aujourd'hui communes et reconnues en génie logiciel (GL) [Chang 97], force est de constater que peu de modèles étayent ces théories. On en distingue essentiellement deux (Figure 8) :

- le modèle en étoile assure à l'évaluation une position centrale. Il identifie des tâches et impose une validation du fruit de chacune d'elles avant passage à la suivante ;
- le modèle en spirale s'inscrit dans cette même lignée. Il s'avère néanmoins bien plus directif que son homologue. Il impose notamment un séquencement précis entre tâches.

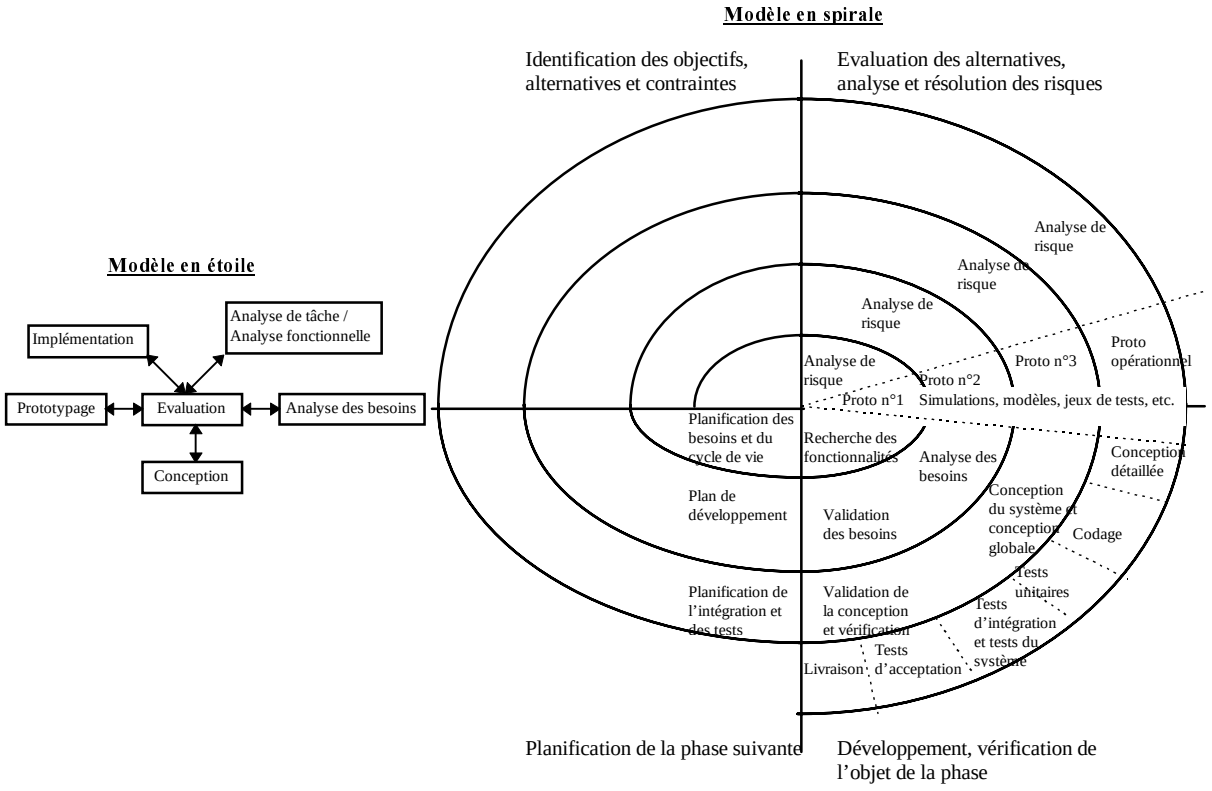

Figure 8 : Modèles en étoile et spirale, respectivement extraits de [Preece 94] et [Balbo 94].

Si ces modèles définissent un cadre de travail efficace et utilisé dans des affaires de grande envergure, ils restent, au quotidien, bien trop génériques ou trop stricts pour être appliqués à la clé. Souvent assimilés à des méta-modèles, ils sont instanciés pour définir un cadre méthodologique plus adapté [Rettig 93]. Typiquement :

- l'analyse des tâches, mentionnée dans l'étoile, n'est qu'indirectement validée via un maquettage : le prototype incarne les spécifications externes contractuelles de l'application (Figure 9) ;

- une fois la présentation entérinée, le processus de développement logiciel est alors déroulé pour les aspects fonctionnels de l'application (Figure 9). Il peut consister en un traditionnel modèle en V du génie logiciel ou une procédure « maison » conforme à la culture d'entreprise. Nous évoquerons ces méthodes dites « dédiées » dans la section 2.

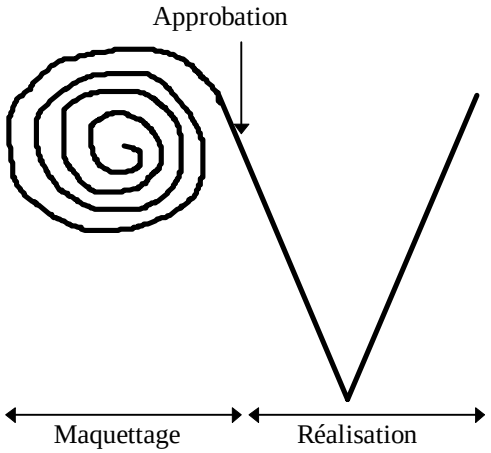

Figure 9 : Une instanciation pragmatique des méta-modèles.

Quelles que soient la formulation et la granularité du processus adopté, l'itération est aujourd'hui la réponse apportée en matière d'IHM. Mais, cette orientation soulève un autre problème : celui de la modifiabilité du logiciel. L'objet est le remède aujourd'hui proposé.

1.3. Approches orientées objet

Si les approches itératives s'imposent aujourd'hui, elles soulèvent le problème de la modifiabilité du logiciel. La réponse jusqu'ici apportée s'exprime en termes d'approches orientées objet. Leur principe consiste à identifier et isoler des unités de compétence applicative. Par la modularité ainsi acquise, ces démarches facilitent conjointement la satisfaction de plusieurs facteurs qualité de McCall [McCall 77].

Pourtant, force est de constater les avis très contrastés émis entre les années 90 et 95 à propos de l'objet. Si le concept était réellement attendu, des réactions plus que frileuses de la part des industriels ont ralenti son émergence. C'est en fait l'absence de normalisation et l'incertitude résultante qui ont longtemps terni ce nouveau concept. Or, la dimension psychologique y est prépondérante. Aussi, seule une adhésion unanime pouvait mettre un terme à cette période transitoire : si l'objet fournit en effet « les bons ingrédients » [Jeanne 92], le volontarisme reste essentiel. « Toute seule, la technologie objet n'améliorera pas la productivité. Elle offre un environnement qui valorise l'ouverture et le partage du travail » [Cash 95].

« L'objet concerne toutes les étapes du cycle de vie des logiciels, depuis la méthode et la conception, jusqu'à la maintenance. Et l'usage de l'objet n'a pas déçu les attentes des entreprises qui, en pionnières, ont tenté l'aventure. Tous les utilisateurs actuels de l'approche objet s'accordent à vanter la facilité de modélisation qu'elle apporte et le temps gagné sur les charges de développement et de maintenance applicative » [Leroy 92] (p 31). Bien entendu, la mise en œuvre de ces approches nécessite une conception rigoureuse dont le surcoût peut aller de 30 à 40% selon J.M. Letourneux (propos recueillis dans [Cahier 92]). Mais, les retombées sont aujourd'hui chiffrées et peuvent compenser largement l'investissement précurseur : tandis qu'il s'évaluerait, en développement, dans un rapport de 2 à 10, il se stabiliserait, en maintenance, entre 25 et 30% (propos de V. Verbeque et J.M. Letourneux respectivement recueillis dans [Scavénius 92] et [Cahier 92]). Mentionnons aussi, à titre indicatif, le projet Artemis qui annonce, en phase de maquettage, un rapport de 40 % dans la gestion désormais orientée objet du trafic ferroviaire belge [Rémy 96].

Si l'objet [Détienne 95b] s'est vu réserver un accueil très mitigé, il s'est à l'inverse très vite imposé en matière d'interfaces homme-machine. Ainsi, [Jeanne 92] p 32 par exemple, qui reste prudent et « rend à l'objet ce qui lui appartient », souligne le bienfait de ces approches en « programmation des interfaces homme-machine, où l'objet apporte à l'heure actuelle des solutions satisfaisantes, là où la programmation traditionnelle avait échoué. L'objet

optimise le travail sur les structures complexes et dynamiques. Parce qu'il raccourcit la distance sémantique entre la réalité, telle qu'elle nous apparaît lors de la modélisation, et sa représentation dans un système informatique ». Convaincu de cette adéquation et, de façon plus générale, des bienfaits de l'approche, THOMSON-CSF RCM Brest décidait à l'époque de contourner le scepticisme en développant en Smalltalk au prix d'une traduction C [Scavénius 92]. Aujourd'hui cette suspicion est levée et la prédilection pour l'orienté objet se généralise. L'approche se voit désormais assortie de méthodes et d'outils et « l'offre l'intègre dans un large éventail de produits, du langage au système d'exploitation, des SGBD aux outils de développement » [Leroy 92] p 31.

2. Méthodes

Par définition, une méthode s'apparente à une technique ou un procédé [Larousse 94]. Une méthode consigne un savoir-faire, le prescrit sous la forme de guides, mais n'en assume pas le respect. C'est la différence essentielle entre méthode et outil : un outil s'inscrit, à l'inverse, résolument dans une démarche de productivité. Il véhicule une méthode, en assiste l'application, voire l'automatise.

Nous nous consacrons ici à une revue des méthodes déployées dans l'industrie. Nous les distinguons selon leur caractère générique ou, au contraire, dédié : tandis qu'une méthode générique émane typiquement du GL, une méthode « dédiée » consigne une procédure « maison », propre à la culture de l'entreprise.

2.1. Méthodes génériques

OMT [Rumbaugh 95], OOA [Coad 93] et aujourd'hui UML [Rumbaugh 97] incarnent les méthodes analytiques de grande diffusion. Elles induisent un cadre méthodologique pour une analyse orientée objet de l'application. Si elles s'ancrent dans l'analyse des besoins, elles couvrent aussi les phases de conception. Elles peuvent même se prolonger jusqu'à l'implémentation, moyennant l'adoption de quelques heuristiques. L'outillage y est alors fondamental :

- il décharge les concepteurs d'une mise en forme fastidieuse ;
- il leur permet de se focaliser sur la sémantique et non les aspects lexicaux et syntaxiques ;
- il favorise la cohérence entre modélisation et implémentation ;
- il restitue finalement au processus de développement sa continuité originelle, si souvent théorique.

StP[22] et Rose 98[23] sont des exemples de tels outils. Ils ciblent OMT et UML. Si la pertinence et l'efficacité de ces méthodes et outils sont aujourd'hui indéniables, nous regrettons leur trop forte imprégnation GL. Ils privilégient la dimension de fonctionnement au détriment de l'utilisation : ils ne ciblent pas les IHM. L'analyse des tâches y est notamment ventilée et les spécifications externes absentes (Figure 10). Si, à l'inverse, des propositions académiques telles MAD [Sebillotte 94] et UAN [Hartson 90] par exemple, couvrent respectivement ces deux aspects, force est de constater que ces formalismes pénètrent timidement l'industrie. Pourtant, leur intégration dans les méthodes génériques en étendrait, non seulement l'applicabilité, mais aussi la couverture.

[22] StP (Software through Pictures) de Aonix.
[23] Rose 98 de Rational Software.

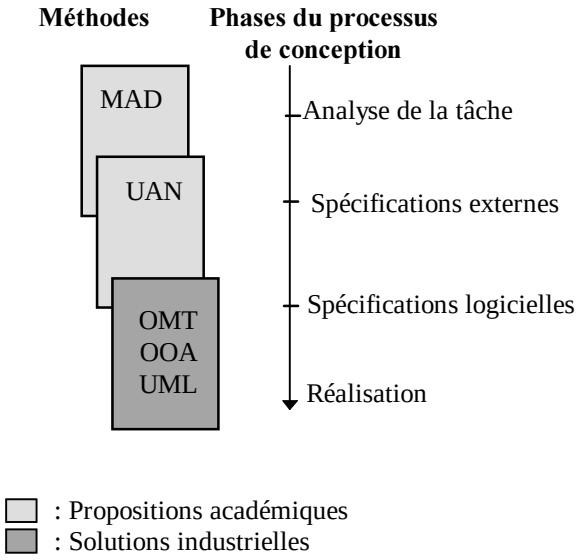

Figure 10 : Comparaison des espaces de couverture des formalismes industriels et académiques. Adapté de [Jambon 96].

OMT et ses homologues souffrent, en effet, d'une couverture de modélisation des plus restreintes. Considérant la décomposition fonctionnelle des systèmes interactifs telle que la propose le modèle Arch [UIMS 92], ils couvrent seulement l'adaptateur du noyau fonctionnel et/ou le noyau fonctionnel. Tout autre composant leur échappe par ailleurs : c'est le cas du contrôleur de dialogue et des présentations logiques et physiques (Figure 11). A l'inverse, UAN cible, de façon privilégiée, ces deux derniers composants. La conviction d'une complémentarité se conforte.

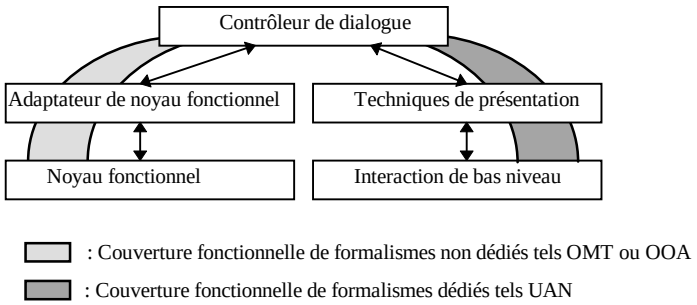

Figure 11 : Comparaison des couvertures fonctionnelles des formalismes industriels et académiques.

A l'opposé de ces méthodes génériques, nous traitons, dans la section suivante, des propositions « maison ». Elles intègrent et formalisent un savoir-faire éprouvé dont le respect est, a priori, gage de qualité et d'efficacité.

2.2. Méthodes dédiées

Les méthodes « maison » sont une manifestation tangible de la fameuse culture d'entreprise. Elles matérialisent et perpétuent un savoir-faire consolidé par l'expérience. Leur objectif est d'éviter la reproduction d'erreurs ou maladresses coûteuses, de tirer parti des expériences passées.

Les entreprises ennemies de l'artisanat et motivées par cette capitalisation méthodologique couchent ainsi leur savoir-faire dans des méthodes spécifiques [Szczur 94]. Ces méthodes s'apparentent, en pratique, à des instanciations de méta-modèles de processus de développement :

- elles contextualisent les méthodes génériques trop éloignées des difficultés quotidiennes ;
- elles intègrent des préoccupations métier propres à l'entreprise ;

- elles formulent ces prescriptions dans une terminologie adaptée à l'entreprise ;
- elles bénéficient d'un soutien et de moyens pour en imposer, assister et vérifier l'application concrète. C'est le cas notamment des services d'assurance qualité logicielle.

Parmi ces méthodes, mentionnons DIADEM [Diadem 94] et ProSPEcT [Brisson 95] :

- DIADEM (Dialog Architecture and Design Method) est la méthode préconisée à THOMSON-CSF Réseaux de Communication et Systèmes de Commandement (RCC). Elle couvre l'ensemble du développement d'une IHM, y compris les phases de spécifications système, dont la qualité est décisive pour celle de l'IHM produite. DIADEM introduit, par ailleurs, la notion de Livres des Normes. Ce livre consigne l'ensemble des règles ergonomiques et des choix adoptés dans le projet. Il s'inscrit dans une recherche de cohérence et d'homogénéité entre les interfaces utilisateurs d'une même application ;
- ProSPEcT (Procédé pour SPEcifier un système interactif Centré sur la Tâche) est une méthode à l'étude aujourd'hui à EDF. Son objectif est de rationaliser, par formalisation, un recours intensif, mais faiblement structuré, à la sous-traitance. Si EDF spécifie, en effet, ses logiciels, il en confie systématiquement la réalisation à des sociétés de service. Il s'agit désormais de canaliser ces prestations par l'identification d'étapes, de fournitures et de plans-types.

Bien entendu, pour ces méthodes prescriptives, l'adhésion de tous est essentielle. Il est donc nécessaire d'en soigner l'utilisabilité : l'outillage en est un moyen. Pour preuve, les outils DIANE++ [Tarby 93], Siroco-Guide [Normand 92] et ERGOVAL [Farenc 97] qui véhiculent respectivement les méthodes DIANE [Barthet 88], SIROCO [Normand 92] et ERGOVAL [Farenc 97]. Le monde académique est donc sensible à cette tendance. Les outils constituent ainsi indiscutablement un créneau industriel de pointe. Nous en dressons une revue dans la section suivante.

3. Outils

Si aujourd'hui le concept du sans-couture[24] s'impose d'un point de vue utilisateur [Ishii 94] [Tani 94], l'informaticien ne dispose toujours pas de méthodes et d'outils couvrant le processus de développement complet des IHM. Nous organisons notre revue sur cette scission selon la phase de construction ou d'évaluation principalement ciblée par l'outil.

3.1. Outils de construction

Si, en termes de noyau fonctionnel, la programmation est essentiellement soutenue par des langages informatiques, un large spectre d'outils cible le dialogue et la présentation. Tous reposent sur un langage informatique, aussi commençons-nous par recenser les langages disponibles, dans le cadre des approches orientées-objet.

Langages et environnements de développement

Après l'apogée du langage C [Kernighan 90], les recherches se concentrent aujourd'hui sur les langages orientés objet. Si JAVA [Flanagan 97] [Badouel 96] pénètre depuis peu les entreprises, son application sur les systèmes critiques n'est pas envisagée pour l'instant : son instabilité est encore incompatible de solution pérenne. Aussi, C++ [Stroustrup 92] n'est-il toujours pas détrôné : il a conquis le domaine et n'en octroie qu'une faible part à Smalltalk [Goldberg 84] et Ada [Booch 94], qui « conservent des niches dans le prototypage rapide et le temps réel » [Berger 92].

THOMSON-CSF RCM Brest illustre parfaitement ce profil : si C++ s'impose aujourd'hui dans le développement de logiciels non interactifs, Smalltalk est omniprésent en matière d'interfaces homme-machine. Tout au moins en phases de maquettage et prototypage où son caractère semi-interprété facilite les itérations. Mais, si par cette propriété, il offre des facilités de mise au point considérables, les relatives lenteurs d'exécution et surtout leur non déterminisme sont souvent invoqués à l'encontre de ce langage. Aussi se limite-t-il essentiellement à des activités de maquettage. Notons pourtant, en toute objectivité, que nous ne nous sommes encore jamais heurtés à une incompatibilité de ce type : bien souvent, les limites cognitives des opérateurs dépassant celles du langage. L'introduction de Smalltalk dans le domaine bancaire en atteste : il n'est plus

[24] Il est intéressant de noter que la littérature propose le terme de cicatrice comme synonyme de couture [Larousse 94].

seulement impliqué dans les traitements de fond, mais aussi dans le développement des postes de travail [André 96]. En matière d'interfaces, c'est en fait bien plus la notoriété du langage C++ qui entretient et renforce ce quasi-monopole : il figure parfois même parmi les exigences client. Nous satisfaisons alors cette contrainte au prix d'une traduction Smalltalk / C++.

Si le caractère semi-interprété de Smalltalk et son faible typage en sont les principaux atouts, c'est aussi grâce à son environnement de développement qu'il résiste aujourd'hui face au favori C++. Il intègre l'édition, la compilation, l'exécution ; et pour tous ces services, une seule ressource est nécessaire : la machine virtuelle. A l'inverse, en C ou C++, même si des produits tels OpenText[25] permettent une navigation au sein du code source, une mosaïque d'outils reste nécessaire : éditeurs, compilateurs, metteurs au point, etc. ObjectCenter[26] et Visual C++[27] sont néanmoins des exemples d'environnements de programmation, mais la convivialité du navigateur Smalltalk reste inégalée.

Si, en termes de réalisation, le langage est le minimum requis, des outils d'aide à la construction d'IHM envahissent désormais le marché : ils affranchissent le développeur de tâches basses et revendiquent, en ces termes, une forte productivité. Nous en étudions le spectre ci-dessous.

Outils

En matière d'outils d'aide à la construction d'interfaces, une taxonomie émerge [Coutaz 90] : elle distingue les boîtes à outils des squelettes d'application et générateurs d'interface (Figure 12).

- Les boîtes à outils s'apparentent à des bibliothèques de procédures. OSF/Motif[28] en est un illustre exemple.
- Les squelettes d'application sont des programmes réutilisables et extensibles. Le programmeur y greffe ses traitements applicatifs.
- Les générateurs d'interface sont des systèmes de spécification. Ils portent sur la présentation, le dialogue ou la sémantique de l'application.

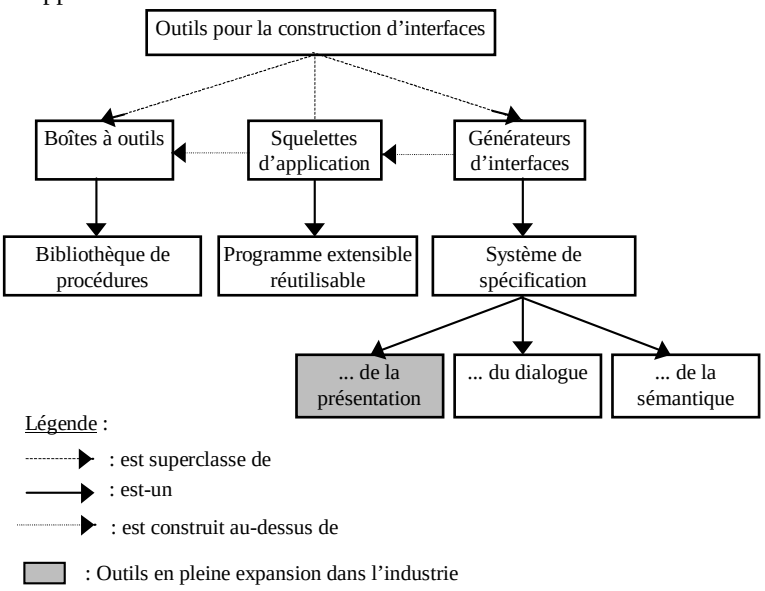

Figure 12 : Taxonomie des outils d'aide à la construction d'interfaces. Adapté de [Coutaz 90].

Si les boîtes à outils s'imposaient dans les années 90, elles sont aujourd'hui supplantées par les générateurs d'interface. Ce sont plus précisément les éditeurs de présentation qui ont réellement conquis l'industrie. Ils doivent cette rapide expansion à leur extraordinaire commodité de mise en œuvre. Des produits, tels

[25] Browser de TNI (Technique Nouvelle Informatique).
[26] De CenterLine Software Inc.
[27] De Microsoft.
[28] De l'Open Software Foundation.

VisualWorks[29] et Views[30], intègrent ces services : ils agrémentent leurs bibliothèques, respectivement construites au-dessus de Smalltalk et C++, d'éditeurs de présentation et plus timidement de dialogue.

Si ces éditeurs de présentation répondent parfaitement aux exigences de maquettage, ils ne constituent aucunement la panacée :

- ils n'assurent pas la conformité de l'application au modèle de tâches ;
- ils ne sont garants d'aucune architecture logicielle et laissent même planer un faux sentiment de sécurité ;
- ils sont inadaptés à la présentation de concepts originaux : ils affranchissent certes le développeur de l'apprentissage fastidieux de langages spécialisés, comme Motif-UIL par exemple, mais en limitent le pouvoir d'expression ;
- ils détournent l'attention vers des éléments surfaciques au détriment de problèmes plus profonds, tels que l'adéquation réelle à l'environnement de travail. Or, la mise en contexte opérationnel s'avère souvent trop onéreuse pour être réellement appliquée. Les retours d'information objectifs sont donc rares [Atwood 95].

Ainsi, les outils de construction disponibles en milieu industriel sont largement insuffisants. Ils le sont d'autant plus qu'ils ne couvrent nullement l'évaluation.

3.2. Outils d'évaluation

En matière d'IHM, l'évaluation est double : tandis que l'évaluation dite fonctionnelle porte sur les traitements, l'évaluation ergonomique traite de l'utilisabilité de l'IHM proposée. Aujourd'hui, aucun outil ne couvre ces deux aspects. Aussi, les recensons-nous selon ce critère discriminant.

Evaluation fonctionnelle

L'enjeu des tests logiciels étant aujourd'hui connu, les propositions affluent en la matière :

- Logiscope[31], par exemple, est un analyseur statique et dynamique de code C. Sur la base de critères et métriques configurables, il évalue la qualité logicielle escomptable. Les métriques classiques répondent à des besoins de programmation structurée. C'est, par exemple, en version statique, le taux de commentaires, le nombre d'instructions par ligne ou les niveaux d'imbrication dans les fonctions. La combinaison de ces critères fournit une estimation de la maintenabilité de la fonction testée. En version dynamique, Logiscope cible les couvertures fonctionnelle et structurelle de l'application. Il rapporte les taux de couverture effectif et graphe d'appels. Si Logiscope s'est longtemps imposé dans les applications de défense et en aéronautique, il partage désormais le marché avec des concurrents tels TCOV[32] et PureCoverage[33];
- Purify[34] et Quantify[35] sont d'autres exemples d'analyseurs. Ils reposent, cette fois, sur des critères orientés ressources : respectivement l'allocation mémoire et le temps d'exécution. Notons que ce premier aspect est fondamental pour nos applications longue durée. Les missions pouvant, en effet, durer plusieurs mois, la détection des fuites est de première priorité. Or, une rigueur dans la libération des octets ne suffit pas toujours à garantir leur pérennité : les fuites peuvent, en effet, provenir des bibliothèques sollicitées, comme Motif, par exemple. Quantify, quant à lui, cible les problèmes de performance : il identifie les portions consommatrices en temps et annote, chaque ligne de code, du temps d'exécution correspondant ;
- à la différence de ces analyseurs, ATOLL[36] est un générateur automatique de tests. Il cible les phases de tests unitaires et tests d'intégration. Sur la base d'un langage propriétaire, il permet la spécification de jeux d'entrées/sorties. Il génère alors un programme de test dont l'exécution se consigne en un fichier : son contenu reflète la conformité des sorties vis-à-vis des valeurs attendues. ATOLL permet aussi de rejouer des scénarios en regard d'exécutions précédentes. Ce service est adapté aux tests de non-régression ; les démarches itératives en accentuent encore l'à propos. Cependant, ATOLL souffre aujourd'hui d'une utilisabilité limitée : la spécification des sorties est, en pratique, délicate et fastidieuse.

[29] De ParcPlace Digitalk.
[30] De Ilog.
[31] De VERILOG (France).
[32] De SUN.
[33] De RATIONAL Software.
[34] De PureSoftware.
[35] De PureSoftware.
[36] De Marben.

Bien que cette revue soit non exhaustive, elle cerne l'offre en matière d'outils d'aide à l'évaluation fonctionnelle. Elle en souligne les caractères décousu et fortement concurrentiel. Ces symptomes se prolongent dans les outils d'aide à l'évaluation de l'utilisabilité.

Evaluation de l'utilisabilité

Dans le principe, les outils d'aide à l'évaluation ergonomique s'inscrivent dans la lignée d'ATOLL : sur la base de descriptions de jeux d'entrées/sorties, ils rejouent des scénarios et comparent les sorties aux valeurs attendues. Les entrées sont ici une interaction de référence enregistrée ; les sorties sont, quant à elles, établies à l'issue de l'exécution. Elles s'expriment en termes de bitmaps ou d'état des éléments de présentation.

On distingue, dans les faits, deux générations d'outils. Tandis que les premiers travaillent en mode point, les plus récents relèvent désormais du niveau logique :

- en mode point, l'interface est appréhendée au niveau physique : les événements d'interaction sont exprimés en termes de point. Ils ne sont pas rattachés aux widgets activés. En conséquence :
  - les descriptions en entrée ne sont pas robustes aux réorganisations spatiales de l'interface : positions et redimensionnements, notamment ;
  - elles excluent le portage sur toute autre plate-forme ;
  - elles exigent un minimum de stabilité de l'interface et sont donc inadaptées aux démarches itératives.

  Xrunner/WinRunner[37] font partie de cette première vague d'outils ;
- les outils les plus récents augmentent d'un niveau d'abstraction : ils s'affranchissent du niveau physique et raisonnent, au contraire, au niveau logique. Ils intègrent la notion de widget et y rattachent les événements opérateur. Les scripts de tests s'en trouvent ainsi, non seulement, plus lisibles, mais aussi plus résistants aux classiques remaniements de l'interface. En cela, ces outils s'avèrent plus adaptés que les précédents aux démarches itératives. QA Partner[38] et QC-Replay[39] en sont des exemples. A la différence du premier, QC-Replay repose sur un langage de commandes non propriétéraire : il adopte Tcl [Welch 97] et affranchit ainsi les développeurs d'un apprentissage de langage supplémentaire.

Malgré une nette tendance de ces outils à l'élévation du niveau d'abstraction, ils restent résolument centrés technique. ErgoLight[40] innove en recentrant le débat sur l'utilisateur : il traque les problèmes d'utilisabilité. Ces problèmes, souvent trahis par des temps de réponse excessifs, des annulations ou des recours aux aides, sont capturés par l'outil et signalés à l'utilisateur. ErgoLight s'enquiert alors, auprès de ce dernier, de son intention originelle. Un modèle de tâche lui sert de structure de contrôle.

Les problèmes d'utilisabilité détectés relèvent :

- de difficultés dans la navigation au sein de l'interface : problèmes de terminologie, de compréhension des procédures, etc. Ces problèmes concernent essentiellement les utilisateurs novices ;
- de la sensibilité de l'interface aux erreurs d'ordre psychomoteur : ces erreurs sont plus prononcées chez les utilisateurs expérimentés ;
- de sa sensibilité aux erreurs d'exploitation : paramètre erroné, etc. Ces erreurs sont typiques des utilisateurs occasionnels.

ErgoLight fournit en sortie :

- des enregistrements de couples actions/intentions pour tout problème détecté ;
- une évaluation du temps perdu dans les erreurs récurrentes ;
- des recommandations pour l'amélioration des performances utilisateur.

Si ErgoLight intègre enfin l'opérateur dans son évaluation, celle-ci reste néanmoins d'ordre opératoire : la description des tâches s'effectue en termes d'actions physiques et le contexte d'interaction semble échapper aux descriptions.

---

[37] De Mercury.
[38] De Productivity Through Software.
[39] De CenterLine.
[40] HTTP ://www.ergolight.co.il

A l'opposé du spectre, Kronos [Sagory 95] est résolument orienté utilisateur. C'est un logiciel d'aide à l'analyse ergonomique. Il cible l'exploitation de données d'observations chronologiques en analyse ergonomique du travail. Il suppose l'adoption préalable de protocoles de description : catégories d'observables - postures, déplacements, communications, prises d'information, etc.-, codes-saisie, etc.

Ainsi s'achève la description du spectre des méthodes et outils disponibles en contexte industriel. Situons THOMSON-CSF RCM Brest dans cette offre : nous recourons, de façon intensive, aux outils d'évaluation fonctionnelle et déployons une veille technologique active en matière d'utilisabilité. Aucun outil commercialisé ne répond, en effet, aujourd'hui à nos exigences applicatives : tous mettent l'accent sur les aspects procéduraux et modèles opérateur. Or, pour nos applications, où les opérateurs sont experts du domaine et l'entrainement intensif, ces notions ne sont pas pertinentes : le savoir-faire et l'appréciation du contexte prédominent, en revanche. Ils échappent hélas à l'offre industrielle.

4. Analyse critique et axes de progrès

Cette analyse de l'ingénierie industrielle en matière d'IHM appelle trois remarques :

- l'absence de structure d'accueil générique embrassant le processus de développement complet des IHM ;
- la profusion de méthodes et d'outils de couverture fonctionnelle très ciblée et partielle vis-à-vis de ce processus de développement ;
- et une offre insuffisante en matière d'outils d'aide à l'évaluation de l'utilisabilité.

En conséquence, le savoir-faire de l'ingénieur prédomine. Si ses technicité et sensibilité peuvent compenser ce manque d'assistance, il est désormais nécessaire de limiter cette délégation excessive pour satisfaire des contraintes de plus en plus pressantes. Des méthodes « maison » apparaîssent pour consigner et pérenniser l'acquis. DIADEM, par exemple, met l'accent sur les spécifications système et l'homogénéité des interfaces. Nous prolongeons ces consignes méthodologiques en termes d'outils. Il s'agit de :

- disposer d'un environnement de travail intégré pour contrer la discontinuité trop franche du processus de développement et garantir ainsi une meilleure traçabilité ;
- favoriser la capitalisation logicielle de connaissances métier pour pallier les spécifications trop souvent incomplètes voire contradictoires ;
- et couvrir l'évaluation de l'utilisabilité.

Le chapitre suivant dresse un cahier des charges en réponse à ces exigences. Nous proposons, au préalable, une synthèse de ce chapitre.

5. Synthèse

Avec l'importance croissante des facteurs humains dans nos applications, les approches itératives par maquettage et prototypage connaissent un vif succès. Clés de voûte à une meilleure appréhension des opérateurs, elles offrent, à moindre coût, un support concret à la discussion. Elles révèlent rapidement les incomplétudes et incorrections des spécifications et permettent une correction anticipée de ces problèmes.

Si les approches itératives aident à converger vers des spécifications externes compatibles des attentes et exigences opérateur, elles soulèvent le problème de la modifiabilité du logiciel. Tandis que les modèles d'architecture traitent de cette exigence, les outils commercialisés l'ignorent. De couverture fonctionnelle très ciblée, ces outils ne couvrent en effet que partiellement le processus de développement. Si les problèmes d'architecture sont ainsi occultés, l'évaluation de l'utilisabilité l'est encore plus. Des méthodes existent certes comme cadre de pensée, mais leur mise en œuvre reste du bon vouloir et des compétences de l'informaticien concerné.

Si cette délégation, en conjoncture économique favorable est tolérable, elle ne l'est plus aujourd'hui. Les exigences en termes de compétitivité nécessitent l'élimination de toute inertie, de toute inefficacité. Aussi, nous orientons-nous vers un environnement de développement pour fédérer ces outils épars, compléter la fresque actuelle, et non alimenter une mosaïque décousue. Nous présentons maintenant le cahier des charges de cet environnement.

CHAPITRE III : CAHIER DES CHARGES

Le postulat de Marie-France Barthet exprime clairement l'absence d'alternative en matière d'adéquation homme-machine : « En l'absence d'amélioration de l'homme, l'amélioration de la communication homme-machine ne peut porter que sur l'amélioration de la conception » [Barthet 88] p 10. Mais, de la même façon, à un niveau méta, en reconnaissant la condition humaine des concepteurs : en l'absence d'amélioration de l'homme, l'amélioration de la conception ne peut porter que sur l'amélioration des méthodes et outils. Sensible au stress et à la fatigue, d'efficacité et de performance variables, le concepteur, comme tout homme, est en effet, par nature, capable du meilleur comme du pire. Aussi, est-il nécessaire de l'assister dans sa tâche par la mise à disposition d'outils adaptés et notamment d'un environnement de travail complet, homogène et intégré.

Si l'analyse de l'existant révèle pléthore d'outils épars et disparates, elle souligne l'absence d'un tel environnement : la critique expérimentale automatique, à fort niveau d'abstraction, fait notamment défaut. En l'absence d'un tel environnement sur le marché, deux solutions se présentent : soit des laboratoires de recherche en esquissent les fils directeurs et il reste alors à les prolonger en une version industrielle finalisée, soit le terrain est vierge en la matière et tout reste à faire. Cette prospection ne peut se faire, de façon efficace, sans un cahier des charges bien ficelé : l'exploration du vaste monde de la recherche, avec toute sa richesse et sa créativité, est en effet propice aux vagabondages. C'est pourquoi, nous nous focalisons ici sur cette expression des besoins. Nous l'organisons en deux volets : après en avoir cerné les aspects fonctionnels, nous nous focalisons sur la mise en œuvre de l'outil. Nous formulons alors des exigences relatives à son utilisabilité, ceci afin d'en assurer une utilisation la plus systématique.

1. Exigences fonctionnelles

A la lumière de l'analyse critique précédente (chapitre II), les exigences fonctionnelles, de première priorité, s'expriment en termes de capitalisation et aide à l'évaluation de l'utilisabilité. Nous les détaillons dans cet ordre.

1.1. Capitalisation métier

L'imperfection des spécifications et la tendance industrielle à la spécialisation par domaine militent pour une capitalisation des connaissances métier. Si cette pérennisation est traditionnellement évoquée en termes d'objets, il s'agit ici de l'étendre au savoir-faire opérateur. Pour les applications de défense, l'entraînement est, en effet, intensif et permet de converger vers un savoir-faire uniforme, représentatif du métier. L'explicitation des procédés est donc, a priori, possible.

Divers formats d'expression sont envisageables. Les versions exécutables sont préférables :

- elles favorisent une plus forte continuité, et donc traçabilité, entre les phases de développement ;
- elles autorisent une génération automatique du code. Elles garantissent alors une conformité entre spécifications et code applicatif.

Dans l'hypothèse de cette capitalisation, les descriptions sont alors :

- consultables pour la conception de nouvelles applications ;
- et disponibles à la réutilisation.

Ces descriptions représentent le domaine et capitalisent l'expérience. C'est, par opposition à l'effet cafetière déjà évoqué, une prédilection pour les infusions : les connaissances immergées dans l'application imprègnent cette dernière de sa sémantique opérationnelle (Figure 13).

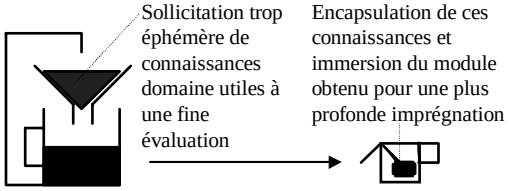

Figure 13 : De la cafetière à l'infusion pour une exploitation durable de connaissances métier.

Dans cette perspective, la mise en œuvre de la capitalisation métier exige :

- le choix d'un langage d'expression pour les spécifications ;
- et une structure d'accueil pour les gestion et exploitation de ces descriptions : organisation, évolution et réutilisation des spécifications, navigation au sein de la base, etc.

Si cette capitalisation est, a priori, une solution aux spécifications incomplètes et pressions industrielles, nous y pressentons aussi un apport en matière d'évaluation de l'utilisabilité.

1.2. Evaluation de l'utilisabilité

Le postulat à la base de notre approche suppose que l'observation d'experts du domaine, placés en conditions opérationnelles, suffit à appréhender, de façon significative, la qualité de l'interaction proposée. Nous postulons que, pour nos applications, où un savoir-faire prédomine, l'adéquation homme-machine se mesure en termes de conformité du comportement opérateur en regard d'un schéma nominal prescrit. Nous déclinons cette efficacité en complétude, correction et concision opératoires. Ces trois dimensions définissent notre espace 3Co (Figure 14) :

- la complétude opératoire s'attache à vérifier que toute procédure attendue est, en pratique, mise en œuvre. Sa transgression peut provenir du non respect de propriétés d'utilisabilité, comme l'observabilité du contexte d'interaction, par exemple, ou d'un écart plus profond entre modèles mental et conceptuel [Kieras 85] ;
- la correction opératoire agit en complémentarité. Elle vérifie que toute procédure mise en œuvre est, en pratique, attendue ;
- enfin, la concision opératoire encourage aux mises en œuvre optimales. Elle condamne les tâches inutiles, comme les sessions d'exploration de l'interface, par exemple.

Propriétés d'utilisabilité	**Définitions**
Complétude opératoire	Toute procédure prévue est effective
Correction opératoire	Toute procédure effective est prévue
Concision opératoire	Toute procédure effective est minimale

Figure 14 : Les complétude, correction et concision opératoires mesurent la conformité du comportement opérateur en regard d'un schéma normatif prescrit. Ces trois critères définissent notre espace 3Co.

Dans ces définitions, nous adoptons les terminologies de [Balbo 94] p43 et [Barthet 88] p217, à savoir :

- une procédure est « le composant exécutif d'une tâche » [Balbo 94] ;
- une procédure prévue (ou prescrite), « est une description d'une tâche telle qu'elle est prévue dans son déroulement standard ». A l'inverse, une procédure effective « est une description permise d'une tâche. C'est un modèle de l'activité » [Barthet 88]. Les complétude et correction opératoires vérifient respectivement, en ces termes, que toute procédure prévue est effective et inversement ;
- une procédure minimale « est l'ensemble des opérations et des enchaînements minimaux pour que le but de la tâche puisse être considéré comme atteint par l'ordinateur » [Barthet 88]. La concision vérifie que toute procédure effective est minimale.

Il s'agit ici de mesurer le respect de ces critères. Nous proposons pour ce faire :

- d'observer un ensemble d'experts du domaine placés en situation opérationnelle ;
- et de détecter, dans leur comportement, toute déviation par rapport au schéma de référence.

Comme l'exigence de capitalisation stipule, par ailleurs, la formalisation du savoir-faire opérateur, ce schéma prescriptif est, dans cette hypothèse, disponible sous forme exécutable : l'automatisation de la détection est donc possible. Plusieurs niveaux sont envisageables :

- l'observation se limite à une surveillance. Elle se consigne en un ensemble de remarques [Larousse 94]. L'expérience montre que, même munie de points de mesure, elle reste insuffisante : le volume des enregistrements la sanctionne sans appel. Le dépouillement des enregistrements s'avère long et fastidieux et ne permet pas aisément de rendre compte des incohérences ou inadaptations de l'interaction proposée

[Calvar 92] [Balbo 94] [Aublet-Cuvelier 96]. Ce dépouillement permet encore moins d'inférer les actions correctives à mener et se heurte à l'impérative nécessité d'une fine connaissance du domaine ;

- l'analyse vise à affiner la connaissance d'un tout [Larousse 94]. Elle recherche des motifs récurrents, établit des statistiques ;
- la critique, selon sa polarité positive ou négative, tend à discerner défauts ou qualités. Elle repose sur une analyse préliminaire (Figure 15) ;
- enfin, le conseil préconise des actions correctives.

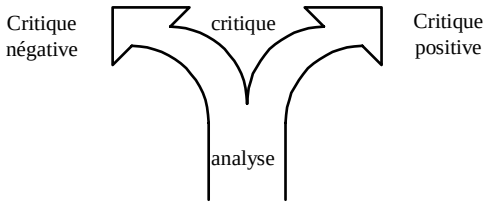

Figure 15 : L'analyse au service de la critique.

Ces niveaux quantifient le « degré d'inférence réactionnelle » de l'outil. Si nous ciblons, sur cette échelle, en priorité, les critiques négatives, les conseils seront appréciés. Ce degré d'inférence dimensionne, en partie, l'espace d'automatisation de l'outil (Figure 16). Les « degré d'autonomie réactionnelle » et « informations additionnelles » le complètent :

- l'autonomie se réfère aux latitudes accordées à l'outil en termes de correction d'anomalie : elle oppose suggestion et implémentation. L'implémentation automatique s'assimile à l'adaptativité automatique de l'IHM. Nous ciblons la suggestion, le bien-fondé des outils correctifs étant loin d'être prouvé ;
- les informations additionnelles ont trait aux justifications et explications fournies par l'outil : tandis que les premières « justifient » les inférences de l'outil, les dernières « expliquent » le comportement de l'opérateur. Seules les premières sont ici strictement nécessaires : elles sont indispensables à l'instauration d'un climat de confiance vis-à-vis de l'outil.

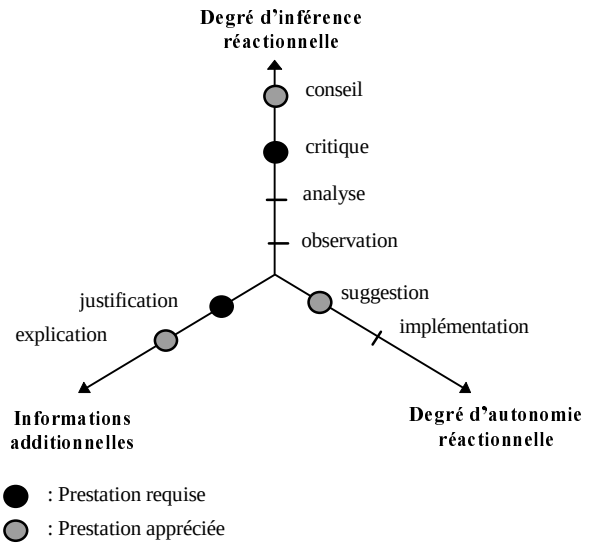

Figure 16 : Espace d'automatisation de l'évaluation de l'utilisabilité : le degré d'inférence réactionnelle de l'outil se réfère à la nature des informations restituées ; le degré d'autonomie cerne les latitudes réactionnelles accordées à l'outil ; enfin les informations additionnelles relèvent de la capacité de l'outil à justifier sa propre contribution ou le comportement de l'utilisateur. Position des profils requis et appréciés au sein de cet espace.

Si les exigences en termes de capitalisation figent la dimension « connaissances » de la grille de choix des méthodes d'évaluation de [Coutaz 94], les exigences en termes d'utilisabilité relèvent de la dimension

« résultats ». Les trois autres facteurs, à savoir, moyens humains, facteurs situationnels, et ressources matérielles ont trait aux exigences de mise en œuvre.

2. Exigences de mise en œuvre

Les difficultés de mise en œuvre liées à l'indisponibilité et au coût de prestation des experts militent en faveur d'une prévalidation prédictive. Il s'agit, par l'application d'heuristiques multidisciplinaires ou de modèles théoriques, de résoudre, hors contexte opérationnel, des problèmes d'utilisabilité potentiels.

2.1. Evaluation prédictive

L'évaluation prédictive répond, en partie, à l'exigence d'économie des moyens humains, comme le suggère l'approche « discount usability testing » de [Nielsen 93]. Si des critères [Bastien 93] [Smith 86], heuristiques ou méthodes sous-tendent ces pratiques dans les faits, [Desurvire 92] rappelle que leur application reste une affaire de spécialistes. Les « Cognitive Walkthrough » en sont un exemple.

Aussi, nous limitons-nous ici aux propriétés de souplesse et robustesse proposées par [IFIP 96] pour affiner l'utilisabilité : tandis que la souplesse se réfère à l'éventail des choix laissés à l'utilisateur, la robustesse s'intéresse aux erreurs (Figure 17). Leur associer des métriques et en automatiser l'évaluation constituerait un axe de progrès majeur.

Propriétés	**Définitions**
Souplesse	
Atteignabilité	Accessibilité d'un but opérateur.
Non-préemption	Accessibilité directe du prochain but opérateur.
Interaction à plusieurs fils	Possibilité d'effectuer des tâches de manière entrelacée.
Multiplicité de la représentation	Disponibilité de plusieurs représentations d'un même concept.
Réutilisabilité des données d'entrée et de sortie	Faculté de réinjecter en entrée du système des données issues de ce même système ou de l'opérateur.
Adaptabilité	Personnalisation du système sur intervention explicite de l'utilisateur.
Adaptativité	Capacité du système à s'adapter à l'utilisateur sans intervention explicite de ce dernier.
Migration de tâche	Capacité de délégation dynamique de tâches entre le système et l'utilisateur.
Robustesse	
Observabilité	Capacité pour l'utilisateur à évaluer l'état interne du système.
Insistance	Capacité du système à forcer la perception de son état.
Honnêteté	Capacité du système à rendre observable son état, de façon conforme à la réalité, et à en assurer une interprétation correcte de la part de l'utilisateur.
Curabilité	Capacité pour l'utilisateur de corriger une situation non désirée.
Prévisibilité	Capacité pour l'utilisateur de prévoir, dans un état donné, l'effet d'une action.
Tolérance du rythme	Faculté pour l'utilisateur de maîtriser le rythme de l'interaction.

Figure 17 : Affinement de l'utilisabilité en propriétés mesurables [IFIP 96].

L'évaluation expérimentale succèdera à cette prévalidation prédictive. Nous formulons, à ce sujet, une deuxième exigence : l'automatisation et la transparence de l'éventuelle instrumentation du code.

2.2. Instrumentation automatique et transparente

Si une instrumentation du code est nécessaire, l'expérience montre qu'elle doit impérativement être automatisée : la déléguer à la machine, c'est en effet s'affranchir de problèmes de correction, complétude et maintenance d'instructions espion. Ces difficultés sanctionnent immanquablement les démarches manuelles [Calvar 92].

Nous complétons cette exigence, par la notion de transparence. Elle se réfère à la discrétion de l'instrumentation. Autrement dit, si l'outil immisce des instructions espion dans l'application, jamais celles-ci ne devront être perçues par le développeur. Non seulement cette discrétion est garante d'une commutation plus aisée entre les modes d'édition et d'évaluation, mais elle élimine, en outre, tout risque de sentiment de dépossession de la part des développeurs : ceux-ci restent maître de leur code et les méthodes qu'ils parcourent leur apparaissent bien telles qu'ils les ont implémentées.

3. Synthèse

L'outil recherché s'adresse à des applications émanant de domaines révélant l'existence de connaissances expertes : conduite automobile, diagnostic médical ou guerre électronique en sont des exemples. Il s'agit de capitaliser ces descriptions pour, conjointement, couvrir les phases de conception et d'évaluation (Figure 18). Il s'agit notamment de pallier les spécifications système trop souvent incomplètes, voire contradictoires.

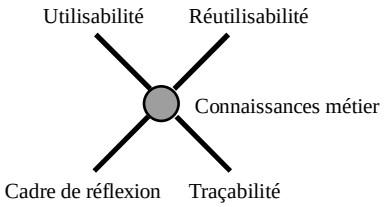

Figure 18 : Vers une modélisation des connaissances métier, pivot aux utilisabilité, réutilisabilité, traçabilité et cadre de réflexion à la conception de nouvelles applications.

Aux arguments de difficulté et de coût qui peuvent être invoqués, rétorquons, outre la criticité de nos applications, leur forte durée de vie et l'inexorable complexification technologique. Aiguisée par une concurrence de plus en plus pressante, la complexité aujourd'hui atteinte n'est désormais plus compatible d'approches artisanales : la rigueur est de mise et il est urgent de l'assortir de méthodes et d'outils.

Si, en matière de conception, le service visé s'exprime en termes de réutilisation, de traçabilité et de guide à la production de nouvelles applications, l'aide attendue pour l'évaluation de l'interaction est une assistance automatisée. Il s'agit de détecter, dans le comportement opérateur, toute déviation par rapport à un schéma nominal prescrit. Pour en optimiser l'efficacité, et notamment réduire les dépendances en termes de facteurs humains, une première passe prédictive détectera toute anomalie théorique, indépendante du contexte opérationnel.

Ainsi, basée sur une capitalisation des connaissances métier, largement encouragée par la tendance industrielle à la spécialisation par domaine, cette thèse traite du processus de développement complet des IHM. Elle met l'accent sur l'évaluation de l'utilisabilité et postule la faisabilité de son automatisation. Il s'agira donc :

- de vérifier la faisabilité technique d'une formalisation, structuration et réutilisation des connaissances métier ;
- de déployer des mécanismes automatiques d'instrumentation du code et d'exploitation des connaissances pour une critique justifiée de l'interaction ;
- d'encapsuler ces services au sein d'une structure d'accueil pour une plus forte continuité du processus de développement.

En l'absence d'outil commercialisé satisfaisant ces exigences, seule une revue de l'état de l'art académique peut délimiter l'ampleur de l'effort à fournir. C'est l'objet de la partie II.

PARTIE II : PROPOSITIONS ACADEMIQUES : ETAT DE L'ART ET AXES DE PROGRES

« Il n'y a qu'une méthode pour inventer, qui est d'imiter. Il n'y a qu'une méthode pour bien penser, qui est de continuer quelque pensée ancienne et éprouvée » Alain.

L’activité des recherches en matière de méthodes et d’outils logiciels se consigne en une abondance de propositions. L’automatisation étant un critère essentiel de notre cahier des charges, nous évinçons, de notre étude, toutes les méthodes manuelles, dénuées de tout espoir d’automatisation. Parmi elles, les modèles cognitifs tels le Cognitive Walkthrough [Lewis 90] [Dix 93] ou les techniques d’évaluation ergonomique à base de questionnaires [Valentin 93] [Barthe 95] [Dix 93].

Malgré ce filtrage, de nombreuses approches persistent. Aussi, définissons-nous, en tout premier lieu, un espace de classification. Cet espace identifie des critères discriminants pour une comparaison objective et rigoureuse des approches envisagées : il fait l’objet d’une description dans le chapitre IV. A la lumière de cet espace problème, nous procédons, dans le chapitre V, à une revue et caractérisation des approches envisagées. Le chapitre VI clot cet état de l’art académique par une analyse critique des voies jusqu’ici considérées.

CHAPITRE IV : ESPACE DE CLASSIFICATION ESPRIT

ESPRIT (Examination Space for Proactive and Reactive Innovative Tools) est un espace de classification visant à rationaliser l'évaluation comparative des approches envisagées, dans le monde de la recherche, pour la construction et l'évaluation des interfaces. S'il cible résolument les outils de construction, il s'inspire indéniablement des acquis en matière d'assistants à l'utilisation. Aussi, dans la première section, proposons-nous, en tout premier lieu, une prise de recul quant aux outils logiciels disponibles en général : nous en esquissons le paysage, y localisons notre zone d'intérêt et mentionnons, sur cette base, des enseignements ciblés.

A la lumière de cette analyse et en vue de nos objectifs, nous introduisons alors deux concepts innovants : la proactivité et la réactivité. Nous en précisons la terminologie, étudions leur synergie puis en proposons une caractérisation.

Nous nous consacrons alors, en section 3, à la description de l'espace ESPRIT. Comme en témoigne son acronyme, cet espace s'articule autour des notions de proactivité et de réactivité. Une fois cet espace décrit, nous spécifions, en ces termes et au regard de notre cahier des charges, le profil de l'outil requis.

1. Spectre des outils, zone d'intérêt et enseignements

Dans un effort de simplification, la vie d'un logiciel peut se décomposer en deux phases :

- une phase de construction, couvrant les itérations nécessaires à la finalisation du produit ;
- et une phase d'utilisation, post-livraison dans laquelle la construction est entérinée.

En regard de cette décomposition macroscopique, les outils se distinguent selon un critère de focus : le focus identifie, dans la vie d'un logiciel, la phase de construction ou d'utilisation ciblée par l'outil.

- Les premiers, dits outils de construction, s'adressent aux informaticiens. Ce sont les boîtes à outils, squelettes d'applications et générateurs d'interface, évoqués au chapitre II.
- Les seconds sont les assistants à l'utilisation. Ils s'adressent, en revanche, à l'utilisateur du logiciel. Ils l'assistent dans ses interactions et le guident notamment en cas d'anomalie ou d'hésitation. Ce sont, en pratique, les outils dits « à base de critique » qui reposent sur un espionnage de l'opérateur. OPADE [De Rosis 94], FRAMER [Lemke 90], CRACK [Fischer 88], JANUS-CRACK [Fischer 90] et HYDRA [Fischer 93] en sont des exemples.

Motivés par une capitalisation des connaissances métier, nous ciblons ici les outils de construction, et plus précisément, les outils de construction par programmation. Nous restons toutefois sensibles aux enseignements de ces approches parallèles. La littérature les ventile en mises en garde et éléments de structuration que nous synthétisons ci-dessous :

- une critique se caractérise par :
  - une polarité [Fischer 88], négative ou positive ;
  - une granularité [Fischer 88], faible ou forte : elle dimensionne la portée locale ou globale de la critique ;
  - une généricité [Fischer 93] : générique, spécifique, relative. Le niveau générique se réfère à des connaissances du domaine, de validité permanente et notamment indépendante de l'utilisateur considéré. Le niveau spécifique porte, à l'inverse, sur des exigences ciblées, typiquement propres à l'utilisateur courant (par exemple, le caractère droitier ou gaucher). Par opposition, le niveau relatif ne relève pas de la validité absolue des connaissances : il traite d'une évaluation comparative, au regard de critères personnalisés (par exemple la conception d'une cuisine dans un objectif de revente) ;
- le mécanisme de critique adopte :
  - une stratégie d'intervention [Sumner 97] : active, hybride ou passive. En version active, le système procède à un suivi continu (ou monitoring) de l'utilisateur. La configuration passive requiert, à l'inverse, une activation explicite, à l'initiative de l'utilisateur. La version hybride propose un mixte : selon un critère (par exemple le changement de pôle d'intérêt), le système se déclenche pour une évaluation typiquement partielle de la contribution ;
  - une politique de diffusion : à la volée ou en différé. Tandis que la première publie instantanément toute critique inférée, la seconde les capitalise pour une publication massive, a posteriori.

L'expérience montre que, malgré un objectif clair, à savoir « dire la bonne chose au bon moment[41] » [Fischer 93], aucune stratégie ou politique ne s'impose dans l'absolu. La clé s'exprime, en termes d'efficacité et mieux vaut, a priori, laisser l'utilisateur configurer la politique d'intervention de l'outil [Fischer 93]. Il faut, par contre, impérativement éviter :

- les critiques redondantes, polluantes susceptibles de lasser l'utilisateur et de l'inciter, à terme, à inhiber les règles déclenchantes voire le système entier ;
- les critiques intempestives interrompant constamment l'utilisateur dans sa tâche. Notons qu'elles découlent souvent d'une stratégie active et d'une diffusion à la volée ;
- les critiques tardives, de correction coûteuse, souvent liées à une stratégie passive ou une publication en différé ;
- une couture trop franche entre contribution de l'utilisateur et évaluation de cette contribution, couture risquant de nuire à la perception des critiques.

Si nous ciblons résolument les outils de construction par programmation, nous retenons, parmi les enseignements des approches alternatives, l'intrication nécessaire entre construction et évaluation. [Fischer 93] préconise, d'ailleurs, leur union au sein d'un même environnement. Pour leur étude, nous introduisons, dans la section suivante, les notions de proactivité et de réactivité.

2. Proactivité et réactivité

Les concepts de proactivité et de réactivité visent à affiner en deux sous-espaces la région couverte par les outils de construction par programmation. Après avoir précisé la signification de ces deux termes, nous étudions leur synergie. Nous en identifions alors des instances puis en proposons une caractérisation.

2.1. Définitions

Par définition, la proactivité caractérise un phénomène « qui s'exerce d'amont en aval dans le temps » [Larousse 82]. Elle se réfère à une progression continue et directe, prohibant tout aller-retour, source d'inefficacité, et revendiquant, à l'inverse, une saine intention de « bien faire ». Si ce terme n'a, à notre connaissance, encore jamais été appliqué aux outils de construction d'IHM, il en incarne pourtant parfaitement l'esprit et l'enjeu, à savoir : aider le concepteur à progresser, de façon directe et efficace, dans la réalisation d'une IHM et converger, si possible sans détour, vers un produit de qualité.

Mais, cette saine intention de bien faire restant de l'ordre du vœu pieux, la réactivité entre en jeu : elle contrôle, a posteriori, le fruit de cette proactivité, hélas non assortie d'une quelconque garantie. La réactivité relève résolument de l'évaluation : elle se réfère à l'aptitude d'un outil à manifester quelque opposition ou approbation quant à la construction courante [Larousse 94]. Elle est typiquement incarnée par les outils d'évaluation.

Ainsi, à l'instar des prévention et guérison, les proactivité et réactivité agissent en complémentarité (Figure 19) : elles sont animées d'un même objectif de qualité et productivité. Tandis que la proactivité vise à bien faire du premier coup, la réactivité vérifie, a posteriori, l'atteinte de l'objectif commun, ceci sur des données tangibles.

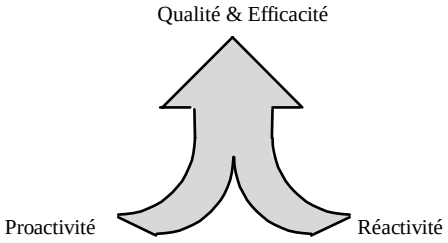

Figure 19 : Mobilisation commune des proactivité et réactivité dans un même objectif de qualité et d'efficacité.

Nous étudions plus finement cette synergie dans la section suivante.

[41] « To say the right thing at the right time » [Fischer 93]

2.2. Synergie

Nous entendons par synergie la mobilisation commune de moyens au service d'un même objectif. Pour l'étudier, nous nous référons aux classifications aujourd'hui admises en matière d'évaluation. Parmi elles, celles qui distinguent les évaluations, d'une part, prédictives et expérimentales ; d'autre part, formatives et sommatives [Coutaz 95b] (Figure 20).

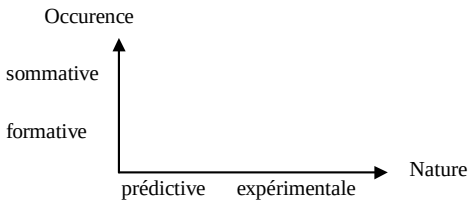

Figure 20 : Taxonomie matricielle des techniques évaluations.

C'est sur cette base que nous étudions les intrications entre proactivité et réactivité.

Prédictif / expérimental

Dans le principe, la proactivité cible la ligne droite entre la finalisation du cahier des charges et la livraison du produit. Elle postule la pertinence d'une intégration précoce et prédictive de consignes multidisciplinaires issues de domaines aussi variés que l'ergonomie, la psychologie ou la sociologie.

Si la proactivité est ainsi intrinsèquement prédictive, la réactivité se décline, à l'inverse, en prédictif et expérimental (Figure 21). Tandis que la première peut être absorbée par une proactivité intense, la dernière reste indispensable à une vérification contextuelle de l'interaction homme-machine. Elle peut être déployée de façon formative ou sommative : nous étudions ci-dessous les intrications de cet ordre.

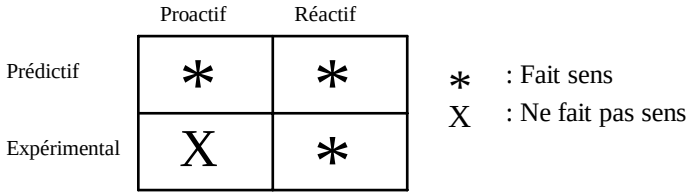

Figure 21 : Si la proactivité est de nature prédictive, la réactivité s'applique, à l'inverse, de façon prédictive et expérimentale.

Formatif / sommatif

Les approches itératives militent pour une combinaison alternée des phases de construction et d'évaluation : l'évaluation ne se limite plus à des usages exclusivement sommatifs, elle est aussi désormais appliquée de façon formative. C'est cette intrication entre, d'une part, le caractère proactif ou réactif d'un outil et, d'autre part, la nature formative ou sommative de l'évaluation, que nous étudions ici.

Considérons, à ces fins, une évaluation pratiquée de façon formative. Elle se consigne en un ensemble de remarques, mises à profit dans l'itération suivante pour en améliorer les sorties. D'où :

- la nature sommative du fruit de l'évaluation : les remarques consignent, c'est-à-dire somment, les singularités détectées lors de l'évaluation ;
- et l'effet proactif de l'évaluation : les remarques sont réinjectées en entrée de l'itération suivante pour en augmenter l'efficacité. Nous baptiserons cette réactivité proactive de « proactivité éducative », par opposition à la proactivité native, liée à un effort initial soutenu en phases amont.

Ainsi, du réactif, appliqué de façon formative, procure-t-il du sommatif, de nature proactive (Figure 22). Cette proactivité est dite éducative par opposition à la proactivité native, liée à un effort initial soutenu en phases amont.

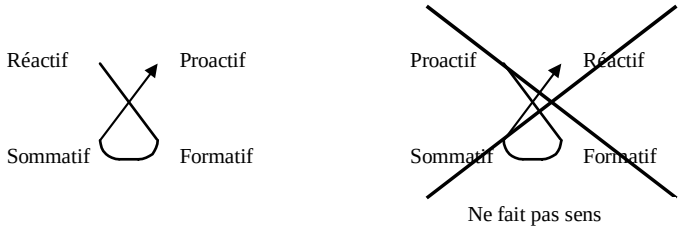

Figure 22 : Du réactif, appliqué de façon formative, procure du sommatif, de nature proactive.

Considérons maintenant une évaluation pratiquée de façon sommative : elle intervient en fin d'affaire et est de nature à valider ou sanctionner une réalisation globale. De par ce caractère terminal, elle profite essentiellement aux réalisations futures : il s'agit bien de tirer parti des expériences passées pour un meilleur avenir. Les bilans d'affaire s'inscrivent dans cette lignée : ils permettent de capitaliser le vécu et d'en faire profiter les développements futurs. L'évaluation sommative participe ainsi à la formation des développeurs, d'où :

Du réactif appliqué de façon sommative fournit des informations formatives, de nature proactive, pour les versions futures ou logiciels à venir (Figure 23).

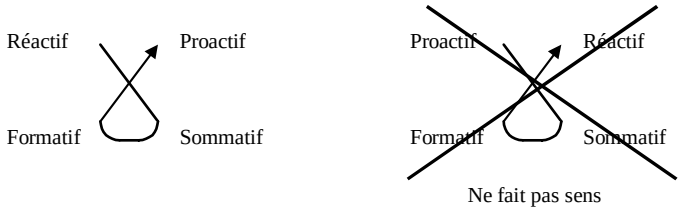

Figure 23 : Du réactif, pratiqué de façon sommative, produit du formatif, de nature proactive.

Nous illustrons ces deux théorèmes par une représentation matricielle : elle reflète les croisements entre les dimensions proactif/réactif et formatif/sommatif (Figure 24).

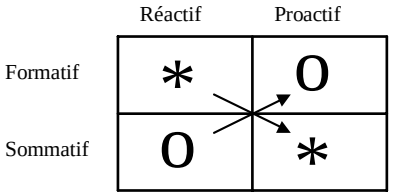

Figure 24 : Compilation des figures 22 et 23 pour une représentation globale des croisements entre les dimensions proactif/réactif et formatif/sommatif.

Si cette analyse a permis de préciser la synergie entre les concepts de proactivité et réactivité, elle a aussi permis de pressentir l'existence de degrés dans la réactivité. En particulier, la réactivité proactive reboucle sur les phases amont, pour l'enrichir d'enseignements pragmatiques. Nous retrouverons cette nuance dans la caractérisation proposée en section 3. Nous concrétisons, au préalable, ces concepts en en mentionnant l'essence.

2.3. Essences proactive et réactive des outils

Si les notions courantes de « feedback proactif » et de « réaction » facilitent l'appréhension des concepts de proactivité et réactivité, nous levons ici toute éventuelle ambiguité résiduelle : nous identifions, dans le contexte précis de la construction d'IHM, des classes d'outils incarnant ces notions.

2.3.1. Essence proactive

La proactivité d'un outil se réfère, par définition, à sa directivité de nature à guider un individu dans son accomplissement de tâche. L'essence la plus connue et la plus représentative, mais non revendiquée en ces termes, en est le paradigme du « Model-Based user interface Design » (MBD). Né voici dix ans, ce paradigme s'inscrit dans la lignée des UIMS (User Interface Management Systems) [Szekely 96a][Szekely 96b]. Il se réfère, dans l'absolu, à une description explicite, largement déclarative, capturant la sémantique de l'application et toute connaissance nécessaire à la spécification tant de l'apparence que du comportement du système interactif[42] [Sukaviriya 94].

Ce sont bien les progrès en langages de spécification qui ont permis d'étendre le pouvoir d'expression des UIMS. Alors que ces derniers se limitent typiquement à la spécification du dialogue, il s'agit désormais de rassembler le plus de connaissances possible au sein de modèles, d'identifier des composants réutilisables, pour réduire la quantité de code procédural nécessaire au développement d'une nouvelle application[43] [Sukaviriya 94]. Le degré d'automatisation de l'exploitation de ces connaissances varie alors selon les instances considérées ; il peut atteindre la génération du code applicatif. Ce degré apparaît dans la caractérisation proposée en section 3.

Retenons que ce paradigme du MBD propose une réelle alternative à la construction des interfaces. Il prône une appréhension descendante et requiert des concepteurs, non plus une programmation informatique, mais la rédaction de spécifications formelles. Décrites dans un langage haut niveau spécialisé, ces connaissances sont ensuite traduites ou interprétées pour une génération totale ou partielle du code applicatif. C'est cette exaltation des connaissances amont, cette réhabilitation de la sémantique qui confèrent à ce mouvement son caractère proactif.

2.3.2. Essence réactive

La réactivité incarne, par nature, l'évaluation de l'interaction proposée. Qu'elle soit prédictive ou expérimentale, la littérature en révèle des instances. Mentionnons par exemple :

- les outils de critique de style, embarquant une version informatisée des recommandations ergonomiques ;
- et les outils d'observation, d'analyse et de critique portant sur l'interaction mise en œuvre par un opérateur en session de travail.

Avant d'étudier ces outils, dotons-nous d'une grille d'analyse. Aussi, caractérisons-nous, en tout premier lieu, les concepts de proactivité et réactivité.

2.4. Caractérisation

A la lumière de l'architecture générale d'un MB-IDE (Model-Based Interface Development Environment) [Szekely 96a][Szekely 96b], nous articulons notre caractérisation autour de deux notions complémentaires : les « degré d'automatisation » et « référentiel de critères ».

2.4.1. Degré d'automatisation

Le degré d'automatisation apprécie le niveau d'inférence atteint par l'outil :

- s'il précise, en matière de proactivité, le niveau de spécification/génération offert, ce degré d'automatisation souligne aussi les opportunités en termes de capitalisation de connaissances. Ce degré se mesure, de façon générale, en termes de couverture :
  - couverture des données d'entrée et de sortie, d'une part ;
  - couverture des données réutilisables, d'autre part.

---

[42] « Model-based user interface design refers to a paradigm which uses an explicit, largely declarative representation capturing application semantics and other knowledge needed to specify the appearence and behaviour of an interactive system » [Sukaviriya 94]

[43] « The goal of the model-based user interface design is to identify reusable components of a user interface and to capture more knowledge in the model, while reducing the amount of new (procedural) code that has to be written for each new application » [Sukaviriya 94]

Tandis que les données d'entrée se réfèrent aux informations spécifiées par le développeur, les données de sortie dénotent celles générées par l'outil. A des fins de quantification et pour fixer une terminologie encore hésitante, nous nous dotons d'une grille d'analyse personnalisée. Elle distingue :

- les objets métier ;
- les tâches utilisateur ;
- le dialogue ;
- et la présentation.

Les objets métier modélisent le domaine en termes d'objets. Ils figurent explicitement dans l'architecture générale des MB-IDE [Szekely 96a][Szekely 96b] (Figure 25). En référence au modèle Arch [UIMS 92], ils se consignent dans l'adaptateur du noyau fonctionnel et/ou le noyau fonctionnel (Figure 26).
Les tâches utilisateur ou tâches métier modélisent le savoir-faire opérateur. Elles conditionnent les spécifications abstraites (Figure 25), c'est-à-dire induisent dialogue et présentation logique (Figure 26).
Le dialogue est géré par le contrôleur de dialogue (Figure 26).
La présentation fédère enfin les aspects logiques et physiques (Figure 26).
Nous adoptons cette grille pour refléter, en capitalisation et spécification/génération, l'apport de l'outil.

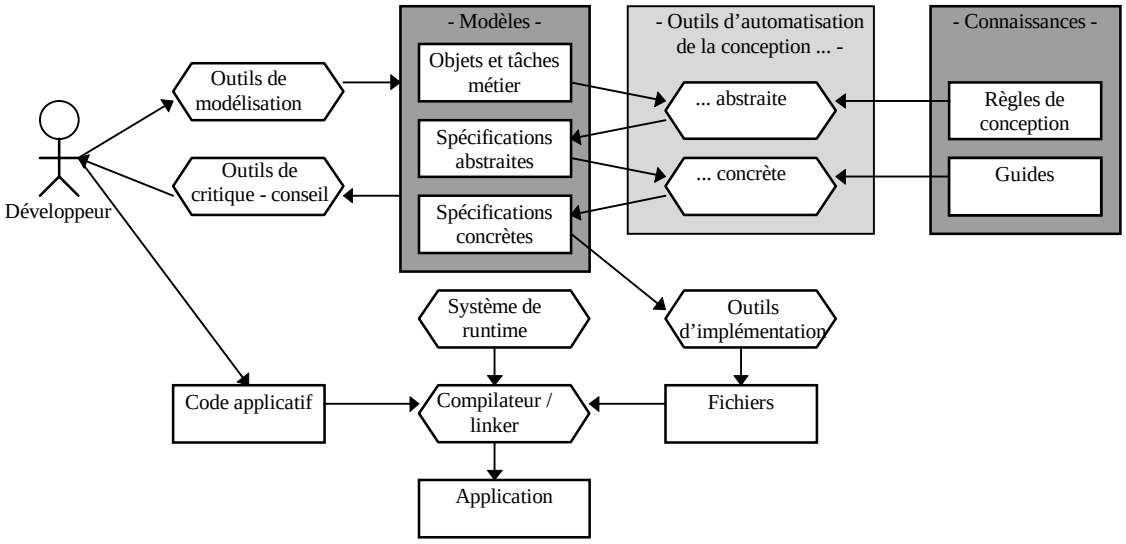

Figure 25 : Architecture générale des MB-IDE. Identification de quatre modèles : les objets et tâches métier, les spécifications abstraites et concrètes. Traduit et adapté de [Szekely 96a] [Szekely 96b].

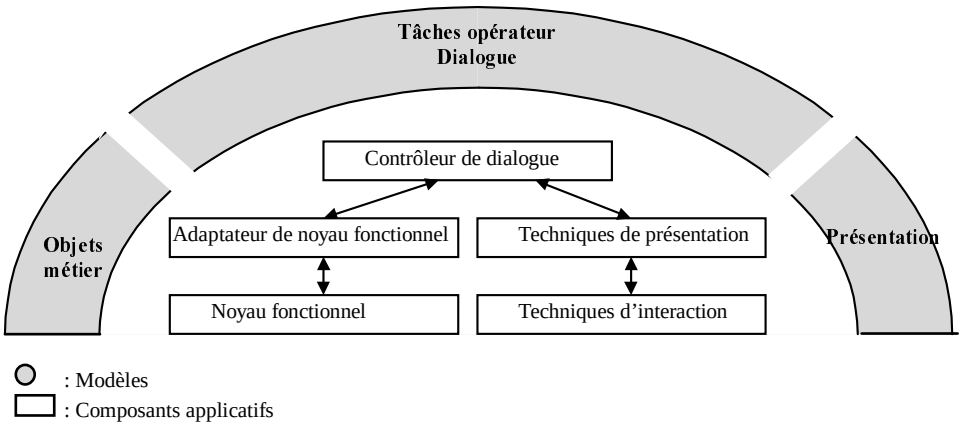

Figure 26 : Projection des modèles sur les composants fonctionnels de Arch.

- en matière de réactivité, le degré d'automatisation distingue :
  - l'observation ;

- l'analyse ;
- la critique ;
- et le conseil.

Nous conservons, en la matière, la terminologie adoptée au chapitre III. Notons que l'observation n'a lieu d'être qu'en évaluation expérimentale. Elle suppose alors une instrumentation du code, dont l'automatisation doit aussi être considérée.

La figure 27 résume, en proactivité et réactivité, les degrés d'automatisation ainsi identifiés.

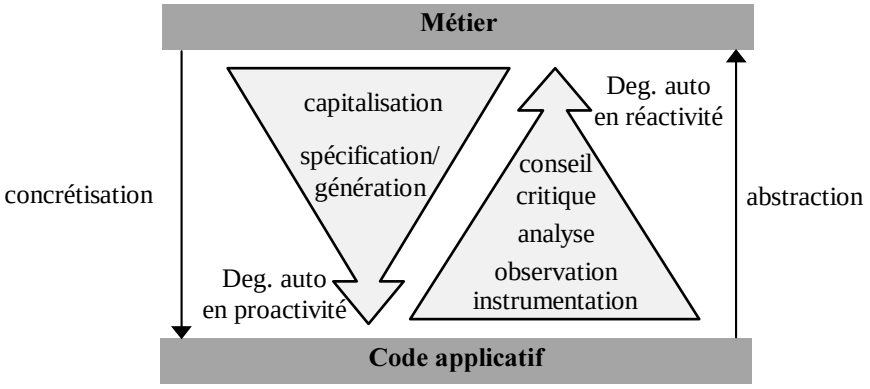

Figure 27 : Degrés d'automatisation en proactivité et réactivité.

Si ce degré d'automatisation permet de mesurer la couverture de l'outil, il n'en reflète aucunement la correction. Aussi, introduisons-nous un référentiel de critères pour exprimer cette notion de contrôle.

2.4.2. Référentiel de critères

Les critères ont ici un rôle de contrôle. Ils cernent :

- en proactivité, les propriétés dont l'outil est garant. Ce sont, par exemple, en génération automatique, les éventuelles règles ergonomiques sollicitées, par l'outil, pour la production de la présentation ;
- en réactivité, les propriétés vérifiées, par l'outil, dans sa prestation d'évaluation.

Nous distinguons deux classes de critères, selon leur généricité :

- d'une part, des critères spécifiques au domaine traité, comme la norme OTAN [OTAN 96] par exemple ;
- d'autre part, des critères génériques consignant, par exemple, les apports de l'ergonomie ou la psychologie. Les propriétés d'utilisabilité mentionnées au chapitre III en font partie [IFIP 96] (section 1.2).

Ainsi s'achève la caractérisation des proactivité et réactivité. Nous intégrons cette réflexion dans l'espace ESPRIT.

3. Espace ESPRIT

Comme en témoigne son acronyme, ESPRIT (Examination Space for Proactive and Reactive Innovative Tools) repose sur ces notions de proactivité et réactivité : il propose une caractérisation des outils selon ces deux volets. S'il adopte, dans le principe, la grille d'analyse précédente en termes de degré d'automatisation et de critères, il l'affine par des décorations supplémentaires :

- il indique, pour les degrés d'automatisation :
  - si des métriques quantifient la contribution de l'outil. Plusieurs métriques sont envisageables : par exemple, la proportion de critères satisfaits ;
  - si des justifications accompagnent les inférences de l'outil ;
- il décore les critères de trois types d'informations :
  - un état : câblé non consultable, câblé consultable ou personnalisable ;
  - un langage d'expression dédié ou non dédié.

Dans cet espace de réflexion, notons :

- qu'autoriser la personnalisation des critères, c'est ouvrir l'outil aux cultures d'entreprise ;
- opter pour un langage dédié, c'est se heurter à un apprentissage supplémentaire mais certainement atteindre un niveau d'abstraction adéquat ;
- opter pour un langage non dédié, c'est, au contraire, s'affranchir de cet apprentissage, bénéficier d'environnements existants et certainement assurer une plus large couverture du processus de développement.

ESPRIT étant ainsi défini, dotons-nous d'un système de notations. La figure 28 y est consacrée.

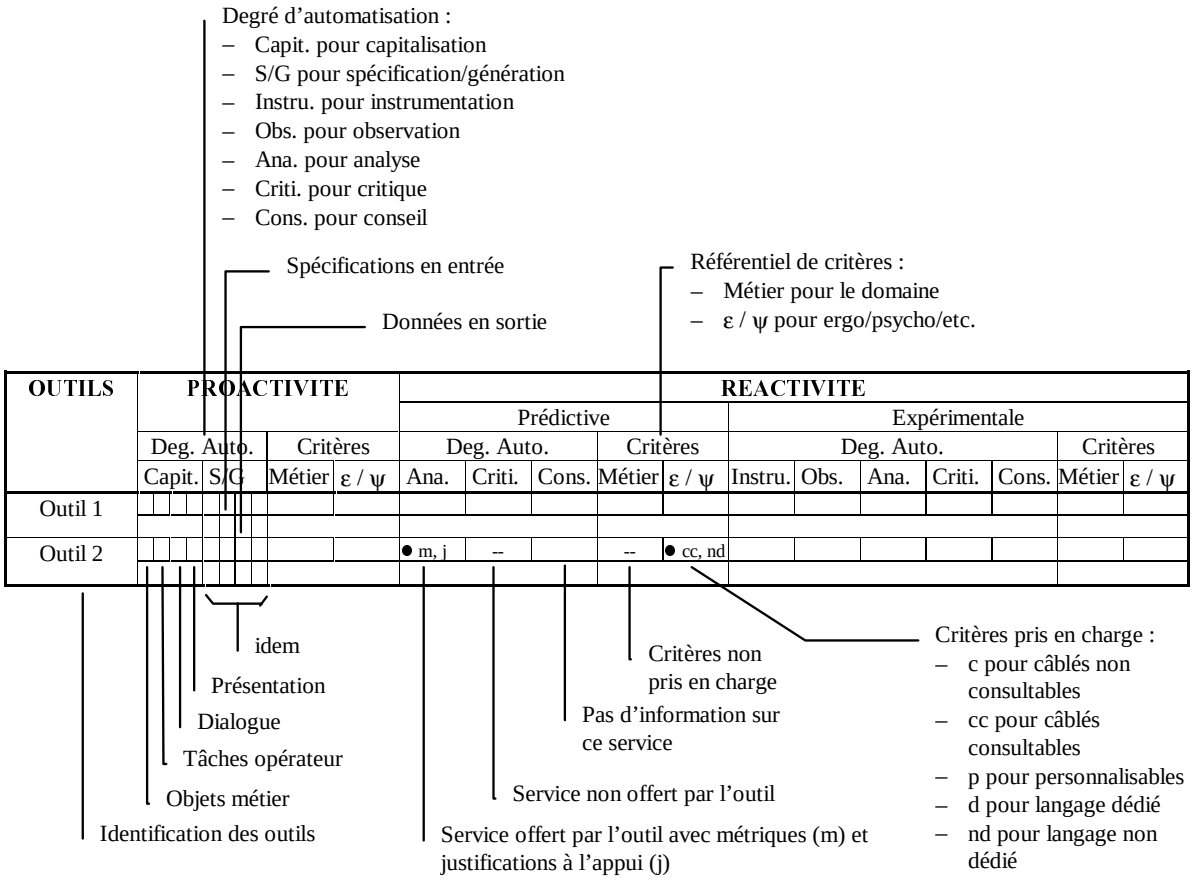

Figure 28 : Notations associées à l'espace ESPRIT.

Avant de procéder à une revue et caractérisation des outils d'origine académique, nous proposons, dans la section suivante, une synthèse de ce chapitre.

4. Synthèse : profil requis

En guise de synthèse, nous projetons le cahier des charges formulé au chapitre III dans cet espace ESPRIT (Figure 29). Rappelons que ce cahier défend :

- la capitalisation des objets métier et savoir-faire opérateur : cette exigence est, par essence, proactive. Elle s'exprime en termes de capitalisation et cible les objets et tâches métier. Des critères métier sont nécessaires aux gestion et évolution des bases de connaissances ainsi constituées ;

- une évaluation expérimentale automatisée et quantifiée de l'interaction homme-machine : cette exigence relève, par nature, de la réactivité expérimentale. Elle cible les niveaux critique et conseil, sur la base des 3Co : correction, complétude et concision opératoires ;
- une mise en œuvre économique de l'outil par :
  - une évaluation prédictive préalable, elle-aussi quantifiée sur la base des propriétés d'utilisabilité ;
  - une instrumentation automatique du code ;
  - et une justification systématique des résultats inférés.

En matière de réactivité, si des observations et analyses sont nécessaires, nous n'exigeons, en la matière, que leur automatisation. Le niveau « critique » étant, en effet, requis, c'est sur ce degré d'automatisation que nous formulons nos exigences : elles s'expriment, en l'occurrence, en termes de métriques, le cahier des charges ne stipulant aucune exigence quant au langage.

OUTILS	**PROACTIVITE**				**REACTIVITE**											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
Outil requis	●●---- ----	---- ----	● p	--	●	● m, j	● m, j	● p	● p	●	●	●	● m, j	● m, j	● p	● p

Figure 29 : Projection du cahier des charges dans l'espace ESPRIT.

Cette réflexion confirme l'adéquation d'ESPRIT pour une analyse critique dirigée des outils aujourd'hui envisagés en recherche. Le chapitre suivant est consacré aux revue et caractérisation de ces approches.

CHAPITRE V : REVUE ET CARACTERISATION DES OUTILS DE CONSTRUCTION

Dans ce chapitre, nous proposons une revue et caractérisation des outils aujourd'hui envisagés, dans le monde de la recherche, pour la construction et l'évaluation d'IHM. Tandis que la revue les décrit de façon factuelle, la caractérisation les positionne au sein de l'espace ESPRIT. Nous structurons cette revue en trois parties suivant le caractère dominant, proactif, réactif prédictif ou réactif expérimental des différentes approches. Pour une plus fine structuration, nous affinons chaque partie selon le degré d'automatisation offert. Nous y décrivons, de façon systématique, chaque outil puis le caractérisons, de façon isolée, dans ESPRIT. Nous commençons par les outils à dominante proactive.

1. Outils à dominante proactive

En matière de proactivité, ESPRIT distingue les proactivités par capitalisation ou spécification/génération. Afin de refléter, dans ce dernier cas, la réelle ontologie de l'outil, nous distinguons les outils de spécification et de génération. La revue est ainsi structurée en trois volets : proactivité par capitalisation, spécification ou génération.

1.1. Proactivité par capitalisation

En pratique, peu d'outils centrent leur prestation sur la capitalisation. S'ils sont sensibles à ces exigences, ils complètent typiquement leur offre par des services de génération ou autres. IDA et EXPOSE en font partie mais leur contribution en la matière mérite néanmoins d'être soulignée.

1.1.1. IDA

Description

IDA [Reiterer 93] (cité dans [Farenc 97]) offre une aide à la conception et l'évaluation d'applications de bureau. Il s'articule autour de trois outils :

- un outil d'aide à la construction met à disposition du développeur des bibliothèques de scripts de dialogue et d'objets d'interaction spécifiques au domaine traité. Ces éléments sont tous définis dans le respect de règles ergonomiques ;
- un outil de contrôle évalue l'interface courante au regard de ces mêmes règles. Cet outil s'apparente à un système expert ;
- enfin, un outil d'aide explique les résultats de ce contrôle de qualité et assiste le développeur dans son utilisation de l'outil de construction.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.			Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.		S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
IDA	--	--	● ● -- -- -- -- -- --	● cc	● c	--	● j	--	--	● c	--	--	--	--	--	--	--

Figure 30 : Caractérisation d'IDA dans ESPRIT.

EXPOSE s'inscrit dans cette même lignée, mais, contrairement à IDA, ne se limite pas à un domaine applicatif donné.

1.1.2. EXPOSE

Description

EXPOSE (Expert system for Phase-Oriented Software Ergonomic counseling) [Gorny 95] (cité dans [Farenc 97]) est un système expert ciblant une conception assistée par ordinateur. Il offre méthodes et conseils en matière de conception. Il exploite, à ces fins, deux types de connaissances :

- des informations empiriques propres au domaine applicatif ;
- et des règles ergonomiques génériques relevant de la psychologie expérimentale et du travail. Ces règles sont issues de guides de style et de recommandations.

EXPOSE induit une méthodologie de conception en quatre étapes. Ces étapes distinguent la réalisation :

- du modèle conceptuel de l'interface ;
- du modèle de la structure du dialogue ;
- des spécifications concrètes de l'interface ;
- et de son prototype.

Le modèle conceptuel de l'interface est établi sur la base d'une spécification détaillée des tâches utilisateur : les rôles opérateur, objets manipulés et arbre des tâches y sont mentionnés. Pour ce faire, le concepteur dispose d'une bibliothèque d'objets du domaine et d'une liste de méthodes permettant de manipuler ces objets. Il lui reste alors à préciser les attributs ergonomiques des tâches ainsi spécifiées.

Si EXPOSE, de par cette capitalisation de connaissances, assiste le concepteur dans cette première phase, il poursuit son assistance en matière de présentation. Grâce à des recommandations ergonomiques embarquées, il guide le développeur dans sa mise en œuvre de l'interface. Les règles sont consultables en blocs ou filtrées selon la nature des objets d'interaction manipulés. Des conseils peuvent lui être promulgués à l'appui, mais sans justification.

Caractérisation

OUTILS	PROACTIVITE							REACTIVITE													
								Prédictive						Expérimentale							
	Deg. Auto.				Critères			Deg. Auto.			Critères			Deg. Auto.						Critères	
	Capit.			S/G			Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
EXPOSE	●	--	--	--	●	--	--	● cc	● cc	--	--	●	--	● cc	--	--	--	--	--	--	--	--
				--	--	--	--														

Figure 31 : Caractérisation d'EXPOSE dans ESPRIT.

Si IDA et EXPOSE amorcent des capitalisations partielles, de couvertures complémentaires, les outils de spécification restent encore insensibles à ces préoccupations.

1.2. Proactivité par spécification

Pour des raisons de concision, nous évinçons de cette revue les outils suivants :

- ITS, The Interactive Transaction System [Boies 88] [Bennet 89] [Wiecha 90], dont nous retiendrons :
  - la reconnaissance explicite du rôle des experts et programmeurs en style pour une personnalisation et capitalisation des règles ergonomiques ;
  - l'opportunité d'une génération de l'interface pour une perception précoce de l'interaction, moyennant le choix d'un style d'interaction quelconque ;
- UIDE, User Interface Design Environment [Foley 88a] [Foley 88b], l'ancêtre des MB-IDE. Nous en retiendrons :
  - un accent résolument mis sur les objets de l'application, ceci au détriment d'un modèle de présentation limité [Szekely 93]. A l'inverse d'ITS, les règles de style y sont prédéfinies et non programmables ;
  - une profusion d'outils connexes déchargeant le développeur d'explorations, réalisations ou évaluations. Par exemple, un outil de transformation balaye, pour un noyau dur de connaissances modélisées, les variantes d'interfaces envisageables. Les différences peuvent porter sur des valeurs par défaut, des pré et post conditions, des sélections, etc. Mentionnons aussi les analyseurs de cohérence, temps d'exécution et apprentissage [Szekely 93] [Foley 88a]. Mentionnons enfin Cartoonist [Sukaviriya 90] et DON [Kim 93] (référencés dans [Szekely 93]) générant respectivement aide contextuelle et boîtes de dialogue ;
- Humanoïd [Szekely 92] [Szekely 93] [Luo 93] un successeur de UIDE. S'il reprend à UIDE la richesse de ses modèles de l'application et du dialogue, il se rapproche d'ITS pour la spécification de la présentation [Szekely 93]. Nous en retiendrons :

- la notion de « templates » et leur organisation au sein d'arbres d'héritage pour factoriser toute intervention sur les éléments de présentation. Par opposition, les générateurs d'interface commercialisés contraignent le développeur à retoucher chaque widget ;
- la notion de « bouchons » pour la construction d'interfaces exécutables, ceci quel que soit le degré d'accomplissement atteint ;
- la possibilité, lors de l'exécution des modèles, de modifier manuellement l'interface et de conserver les évolutions ainsi apportées.

Nous commençons la revue par Mastermind, un successeur de UIDE et Humanoïd.

1.2.1. Mastermind

Description

Mastermind [Szekely 95] prolonge les travaux menés dans le contexte de Humanoïd et UIDE. Il vise à réduire le coût de construction des interfaces et à encourager la construction de produits de qualité. Basé sur une modélisation du domaine, des tâches et de la présentation :

- il facilite les prototypages rapides et développements itératifs ;
- il permet une conception en ligne et un affinement pour une convergence continue vers un produit de qualité.

Il définit, à ces fins, un vocabulaire de concepts et d'attributs, appelé « ontologie de Mastermind ». Désormais adoptée dans de nombreuses approches, cette ontologie fédératrice permet une spécification déclarative des différents modèles, à savoir : le modèle de l'application, des tâches et de la présentation.

- Le modèle de l'application est une extension du modèle objet de CORBA (Common Object Request Broker Architecture) [Geib 98]. Il complète les « interfaces » de cette norme par les notions de préconditions et de notification :
  - les préconditions explicitent le contexte dans lequel l'appel à la méthode est valide. Elles se formalisent par le biais d'expressions ;
  - les notifications permettent, quant à elles, à des objets, y compris tâches et présentations, de se déclarer intéressés par certains changements survenus au sein d'objets.
- Le modèle de tâches décrit les tâches utilisateur. Pour l'essentiel, une tâche est modélisée par :
  - un nom ;
  - un but ;
  - des paramètres d'entrée et de sortie ;
  - des préconditions portant sur ces paramètres d'entrée ou l'état d'autres tâches ;
  - une procédure décomposant la tâche en sous-tâches, jusqu'à dénoter les techniques d'interaction. Quatre opérateurs, dits de connexion, sont disponibles à ces fins : « sequence » et « parallel » pour des exécutions respectivement séquentielles et parallèles; « unrestricted » pour exprimer l'absence de contraintes, autres que celles dérivant des préconditions; « one_of » lorsque l'exécution d'une seule sous-tâche suffit. Notons que les tâches mentionnées dans cette décomposition n'incombent pas forcément à l'opérateur : l'attribut de type, évoqué ci-dessous, précise l'acteur en charge de la tâche ;
  - un type affectant, pour l'essentiel, la tâche à un exécutant ;
  - des attributs concernant le contrôle du dialogue. Parmi eux, les caractères optionnel, interruptible, la reprise sur interruption, la notion de boucle et le caractère réentrant, ceci en cas d'instanciation multiple d'une même tâche ;
  - les effets spécifiant les actions exécutées lors du déroulement de la tâche. Ils consistent typiquement en appels de routines applicatives, présentations d'informations, changement d'états d'autres tâches (déclenchement, interruption, arrêt, exécution) ou modification de paramètres de ces dernières. Notons que ces spécifications peuvent être traduites en code exécutable pour des démonstrations du comportement prévu.
- Le modèle de présentation définit l'apparence visuelle de l'interface. Il s'exprime en termes d'éléments de présentation. Chacun d'eux est, pour l'essentiel, défini par :
  - un nom ;
  - des constituants ;
  - des paramètres standard (fontes, couleurs, etc.) et applicatifs (présentation d'objets du domaine) ;

– des alternatives de présentation pour faciliter d'éventuels portages et notamment faire face aux conflits de place sur écrans exigus.

Ces trois modèles sont définis et stockés sous une forme déclarative. Si, pour des raisons d'efficacité, ils sont, à terme, compilés en code source C++, leur version déclarative reste toujours consultable dynamiquement. D'un point de vue architecture, ils sont au cœur d'un serveur de données et accessibles, via une couche de communication CORBA. Tout client peut ainsi se connecter en cours de session et accéder aux modèles en lecture ou écriture. Les évolutions sont systématiquement propagées vers les clients concernés. Charge à eux, par contre, de se mettre à jour à la volée. Notons que cette architecture distribuée favorise le travail de groupe.

Parmi les outils supports, mentionnons les services de :

- spécification et construction ;
- génération de code;
- génération d'aide ;
- personnalisation de l'interface ;
- formulation de critiques ;
- exploration du code (liste des procédures activables, procédures à déclencher pour en rendre une autre activable, etc.) ;
- et exécution des modèles. A noter alors le système de bouchons, emprunté à Humanoïd, qui permet une exécution précoce de ces modèles, même incomplets, et favorise ainsi un développement incrémental.

En résumé, retenons de Mastermind :

- un fort potentiel en termes de capitalisation de par les modularité et couverture des spécifications d'entrée ;
- l'absence de support à cette capitalisation, peut-être due à l'absence de frontière entre descriptions métier et solution applicative ;
- une opportunité de capitalisation finalement trop implicite et largement baffouée par des objectifs de génération, exploration, critique, etc.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
Mastermind	-- -- -- --	● ● ● ●	--	--		--	●	--	--	●	--	--	--	--	--	--	--
		-- -- ● ●															

Figure 32 : Caractérisation de Mastermind dans ESPRIT.

Mecano poursuit les travaux de Mastermind : il en reprend l'ontologie et en étend l'application.

1.2.2. Mecano

Description

Le projet Mecano [Puerta 96] est né au début des années 1995. Il s'articule autour de deux objectifs :

- définir, et mettre à la disposition de la communauté IHM, un modèle d'interface générique, complet, portable et indépendant de tout système ;
- sur la base de ce modèle, concevoir et réaliser un environnement ouvert permettant, outre la génération de l'interface, la connexion de tout outil respectant ces consignes de formalisation.

C'est donc en termes de structure d'accueil que s'expriment les ambitions de Mecano. Aujourd'hui, seule la première phase est mature : elle se consigne en un langage de modélisation MIMIC. MIMIC est un langage orienté-objet, en pratique implémenté en C++. Il cible la modélisation des tâches utilisateur, du domaine, de la présentation, du dialogue, de l'utilisateur et de la conception. C'est ce dernier composant qui en fait l'originalité :

il explicite les connexions entre les trois modèles : tâches, domaine et présentation. Ces connexions s'apparentent à des règles de mise en correspondance, décorées de conditions de validité.

Si ce langage est aujourd'hui stabilisé, le deuxième objectif de Mecano n'est, en revanche, que partiellement atteint. Notons, par exemple, que Mecano ne couvre aujourd'hui, d'un point de vue génération, que les interfaces graphiques de type formulaire [Puerta 97].

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
Mecano	-- -- -- --	● ● ● ●	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
		-- -- -- --															

Figure 33 : Caractérisation de Mecano dans ESPRIT.

Mobi-D est un successeur de Mecano :

- il conserve les notions de modèles et de liens entre modèles ;
- et traite des aspects méthodologiques.

1.2.3. Mobi-D

Description

Mobi-D (Model-Based Interface Designer) [Puerta 97] s'inscrit dans la lignée de Mecano :

- il propose, de même, un méta-langage de modélisation ;
- et réhabilite les choix de conception.

Le paradigme du MBD n'y est plus une finalité mais un moyen de satisfaire des exigences amont. Mobi-D définit un processus de conception. Il défend une conception participative, avec une implication centrale et régulière des utilisateurs finaux. Les modèles y sont alors garants de modularité et de réutilisabilité, ceci notamment grâce à leur caractère déclaratif.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
Mobi-D	-- -- -- --	● ● ● ●	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
		-- -- -- --															

Figure 34 : Caractérisation de Mobi-D dans ESPRIT.

Ainsi s'achève cette revue consacrée aux outils de spécification. La proactivité y reste « lâche », essentiellement soutenue par une structure d'accueil favorisant la modularité. Il n'en n'est pas de même dans les outils de génération.

1.3. Proactivité par génération

Par nature, les outils de génération s'apparentent à des éditeurs de sémantique. S'ils sont très peu représentés dans l'industrie, ils se multiplient, en revanche, dans le monde de la recherche. ADEPT en est un exemple pionnier.

1.3.1. ADEPT

Description

ADEPT (Advanced Design Environment for Prototyping with Task Models) [Johnson 93] [Kelly 92] [Wilson 95] est un environnement de développement d'interfaces qui s'appuie sur une description explicite des tâches utilisateur. Il couvre toutes les phases allant de cette analyse de tâche à la génération de code exécutable. Son processus s'articule autour de la notion de modèles, chaque phase étant étayée d'outils (Figure 35) :

- un éditeur graphique de tâches permet la construction et la consultation de tâches exprimées selon la méthode TKS [Johnson 91]. La sortie alimente un générateur d'interfaces abstraites ;
- ce générateur d'interfaces abstraites fournit une spécification haut-niveau de l'interaction. Les structures de dialogue et les objets abstraits d'interaction y sont mentionnés. Un éditeur permet au développeur de personnaliser ce modèle. La sortie alimente un générateur d'interfaces concrètes ;
- le modèle d'interface concret est une description de l'interface en termes de techniques d'interaction. Il précise notamment les agencements et comportements des différents tableaux de bord. Un outil permet d'en générer une version par défaut. Il prend en compte le modèle de l'utilisateur (expert/novice, etc.). Un outil permet de personnaliser le modèle ainsi obtenu ;
- enfin, un générateur de code produit du code source, fonction de la plate-forme cible.

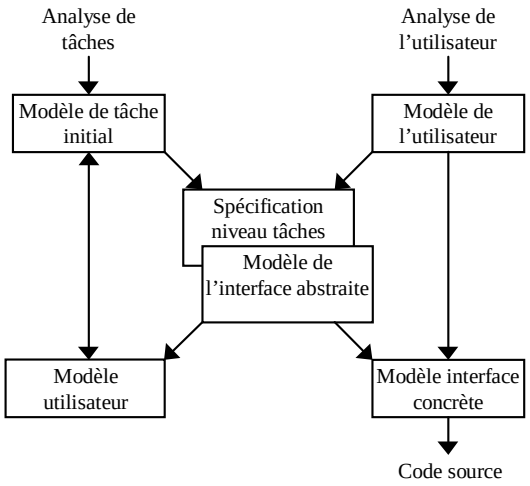

Figure 35 : Les modèles et processus d'ADEPT. Traduit de [Johnson 93].

Ainsi, ADEPT génère automatiquement une interface à partir d'un modèle de tâche et de l'utilisateur. La conformité entre tâches et présentation est ainsi garantie. Le prix de cette autonomie s'exprime en termes de présentation : l'environnement reste fermé et empêche le concepteur de se définir des éléments de présentation personnalisés [Szekely 95].

Caractérisation

OUTILS	PROACTIVITE								REACTIVITE													
									Prédictive						Expérimentale							
	Deg. Auto.				Critères				Deg. Auto.			Critères			Deg. Auto.						Critères	
	Capit.			S/G			Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ		Instru.	Obs.	Ana.	Criti.	Cons.		Métier	ε / ψ
ADEPT	--	--	--	--	●	--	--	●p, nd	--	--	--	--	--		--	--	--	--	--		--	--
				--	--	●	●															

Figure 36 : Caractérisation d'ADEPT dans ESPRIT.

L'outil de DIANE+ vise un objectif similaire : en forçant d'emblée une spécification formelle de l'espace des tâches, il garantit, de même, la conformité du code à l'analyse des besoins fonctionnels.

1.3.2. DIANE+

Description

L'outil de DIANE+ [Tarby 93] propose des règles de génération automatique d'une IHM à partir d'un modèle de tâches DIANE+. DIANE+ [Barthet 88] est une méthodologie de développement d'applications interactives s'appuyant sur la méthode MERISE. Elle distingue trois vues : celles de l'analyste, de l'utilisateur et du programmeur.

- Le point de vue de l'analyste décrit, par une Représentation Conceptuelle, la logique générale du système puis sa logique de traitement, par rapport à un poste de travail donné. C'est une vue opérationnelle ;
- La vue de l'utilisateur, appelée Représentation Externe, correspond au logiciel tel qu'il sera manipulé par l'utilisateur. C'est la traduction de la Représentation Conceptuelle, plus une définition de tous les éléments d'interaction ;
- La vue du programmeur décrit, par une Représentation Interne, la mise en œuvre des deux autres représentations. C'est une vue purement technique.

L'outil de DIANE+ couvre la traduction de la Représentation Conceptuelle vers la Représentation Externe. Il permet :

- la saisie de la spécification du dialogue homme-machine, via les procédures DIANE+ ;
- la génération de l'interface, selon des critères ergonomiques généraux ;
- la gestion de la dynamique et de l'aide.

Il offre enfin des tests à tous niveaux : vérification de la syntaxe des procédures, détection de blocages, etc.

Caractérisation

OUTILS	PROACTIVITE								REACTIVITE													
									Prédictive						Expérimentale							
	Deg. Auto.				Critères				Deg. Auto.			Critères			Deg. Auto.						Critères	
	Capit.			S/G			Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ		Instru.	Obs.	Ana.	Criti.	Cons.		Métier	ε / ψ
DIANE+	--	--	--	--	--	●	--	--	●c	--	--	--	--	--		--	--	--	--	--		--	--
				--	--	--	●															

Figure 37 : Caractérisation de DIANE+ dans ESPRIT.

ALACIE s'inscrit dans cette même lignée : il concilie méthodologie et environnement de développement dans une approche centrée tâche.

1.3.3. ALACIE

Description

ALACIE (Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques) [Gamboa-Rodriguez 98] répond, selon trois perspectives complémentaires, aux difficultés actuelles de l'ingénierie des interfaces homme-machine. La tâche utilisateur, en tant que fruit de l'analyse de l'existant, y est initiatrice :

- d'un point de vue modélisation, les modèles MAD* et SSI permettent respectivement la description des tâches utilisateur et de l'interface sémantique. L'interface s'exprime à deux niveaux d'abstraction par des *schémas de procédures* et *procédures*. Ces éléments peuvent être assortis d'une représentation physique à l'écran, représentation qu'ils gèrent alors selon leur état interne. Ils sont capables de propager cet état aux objets voisins par le biais de mécanismes. Ces mécanismes sont à la base du contrôle de dialogue. MAD* et SSI correspondent respectivement aux début et fin du processus de conception. Des liens explicites connectent le modèle SSI à la modélisation originelle MAD* ;
- d'un point de vue méthodologique, [Gamboa-Rodriguez 98] préconise un processus, non forcément linéaire, en cinq étapes. Ce processus est basé sur la modélisation MAD*. Il conduit, en final, via SSI, à la génération de prototypes ;
- d'un point de vue pratique, ALACIE propose deux outils IMAD* et ISSI implémentant respectivement MAD* et SSI. IMAD* permet l'édition des arbres de tâches MAD* avec notamment : la gestion des objets référencés ; l'édition, la vérification et l'évaluation des pré et post conditions ; la simulation du déroulement des tâches. ISSI aide à la traduction des composants de ces tâches en éléments de l'interface sémantique. Il participe aussi à leur assignation de représentations physiques. Il gère l'exécution dynamique de la maquette obtenue ; génère et gère partiellement une aide. Notons que ces deux outils, IMAD* et ISSI, favorisent les développements itératifs en autorisant des niveaux de formalisation divers. Ils satisfont, en ceci, l'objectif d'ALACIE : faciliter la collaboration entre experts multidisciplinaires et ceci tout au long du processus de conception.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
ALACIE	-- -- -- --	-- ● -- --	--	●c	--	--	--	--	--	--	--	--	--	--	--	--	
		-- -- ● ●															

Figure 38 : Caractérisation d'ALACIE dans ESPRIT.

TADEUS concilie, de même, méthodologie et outils sur une approche progressive centrée tâche.

1.3.4. TADEUS

Description

TADEUS (TAsk-based DEvelopment of USer interface Software) [Elwert 95] [Schlungbaum 96] est à la fois une méthodologie et un environnement de développement. Il défend une progression continue, allant de l'analyse des besoins à la production de l'IHM exécutable. Il s'articule autour de quatre modèles :

- les modèle de la tâche, de l'utilisateur et du domaine du problème consignent le fruit de l'analyse des besoins. Les méthodes de génie logiciel permettent de les élaborer ;
- le modèle du dialogue décrit l'interface de façon statique et dynamique. Il distingue la navigation du traitement :
  - la navigation se réfère aux interactions entre vues ; une vue regroupant des objets de dialogue logiquement ou fonctionnellement connectés ;
  - le traitement relève, quant à lui, d'une description procédurale, spécifique à chaque vue. Il traite de la représentation locale des objets, de la forme du dialogue, des transitions, etc.

D'un point de vue mise en œuvre, TADEUS offre deux niveaux de génération (Figure 39a) :

- l'élaboration d'une version initiale du modèle de dialogue à partir des trois autres modèles ;
- la construction d'une IHM prototypique sur la base de ces descriptions et en regard de règles ergonomiques.

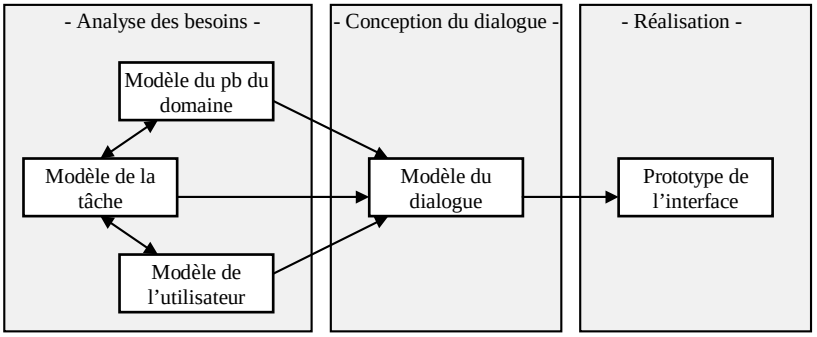

Figure 39a : Principes de TADEUS.

Caractérisation

OUTILS	PROACTIVITE						REACTIVITE												
							Prédictive						Expérimentale						
	Deg. Auto.			Critères			Deg. Auto.			Critères			Deg. Auto.					Critères	
	Capit.		S/G		Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ		Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
TADEUS	--	--	--	--	--	●	--	--	--	--	--		--	--	--	--	--	--	--
			--	--															

Figure 39b : Caractérisation de TADEUS dans ESPRIT.

Avant de clore sur cet outil, soulignons les deux originalités suivantes :

- la réutilisation de concepts et outils existants. Par exemple :
  - les réseaux de Pétri pour le dialogue et, par voie de conséquence, le bénéfice de tous les mécanismes de vérification de correction, détection de blocages, etc., disponibles en la matière ;
  - l'UIMS ISA Dialog Manager pour la génération de l'IHM ;
- la conciliation permanente entre méthodologie et outils : FUSE cible ce même objectif.

1.3.5. FUSE

Description

FUSE (Formal User Interface Specification Environment) [Lonczewski 96] est un environnement de développement à part entière. Il allie méthodologie et outils pour une couverture complète du processus de développement. La méthodologie s'articule autour de trois phases :

- l'analyse des besoins cerne les exigences. Elle les consigne en trois modèles :
  - le modèle de la tâche décrit, de manière hiérarchique, les tâches de l'application ;
  - le modèle du problème du domaine est une restriction du noyau fonctionnel aux objets pertinents pour la présentation ;
  - le modèle de l'utilisateur est une description des propriétés statiques et dynamiques relatives aux groupes d'utilisateurs ou utilisateurs particuliers ;
- la phase de conception requiert, au préalable, la contribution d'ergonomes, en présentation et conversation, pour la définition de guides de style. Quatre outils dédiés, FIRE, FLUID, BOSS et PLUG-IN, assument alors les tâches de transformation :
  - FIRE fournit des éditeurs graphiques pour la spécification de ces trois modèles. Il propose, en outre, un service de génération d'une interface prototypique, en support à la discussion des modèles ainsi définis ;

- FLUID est en charge de la génération des spécifications abstraites : il applique les règles de style définies par les ergonomes pour obtenir les spécifications logiques formelles de l'interface. Le développeur peut modifier le résultat ainsi acquis ;
- BOSS [Schreiber 94] poursuit cette génération par la suggestion de spécifications concrètes ;
- enfin PLUG-IN, sur la base de l'interface ainsi construite, et en regard du modèle de l'utilisateur, génère une aide contextuelle de guidage, au format hypertexte ;

- la phase d'évaluation porte sur l'IHM générée. Elle n'est pas assumée par l'outil. Elle suppose, au préalable, une connexion entre le noyau fonctionnel de l'application et les composants de cette IHM générés par l'outil.

Caractérisation

OUTILS	PROACTIVITE							REACTIVITE											
								Prédictive					Expérimentale						
	Deg. Auto.			Critères				Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.		S/G			Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
FUSE	--	--	--	--	● ●	--	--	● p	--	--	--	--	--	--	--	--	--	--	--	--
					-- -- ● ●														

Figure 40 : Caractérisation de FUSE dans ESPRIT.

TRIDENT s'inscrit dans cette même lignée : il offre un support méthodologique et un ensemble d'outils d'aide à la construction d'interfaces.

1.3.6. TRIDENT

Description

Le projet TRIDENT [Bodart 95b][Vanderdonckt 93][Vanderdonckt 94] apporte, à la fois, une réponse méthodologique et un ensemble d'outils visant à aider le concepteur dans ses compromis quotidiens : modularité, indépendance du dialogue, partage de tâches pour une conception collaborative, etc., sont, en effet, autant de critères dont la conciliation se fait, en pratique, au prix de délicats compromis. TRIDENT propose une alternative à cet artisanat : sur la base d'une modélisation des tâches et des unités de présentation, il propose la génération automatique d'une IHM en en garantissant une architecture logicielle systématique. Cette architecture, de structuration hiérarchique, s'articule autour de trois types d'objets :

- les objets de contrôle gèrent le dialogue. Ils relient les données de l'application et les éléments de présentation, par des diagrammes état/transition. Ils sont en pratique définis par un langage de scripts, à base de règles ;
- les objets de l'application dénotent les fonctions de l'application. Les mentionner explicitement dans l'architecture permet d'en rehausser la sémantique : ce sont ces fonctions qui incarnent le rôle et la raison d'être des tâches ;
- enfin, les objets d'interaction se réfèrent aux techniques d'interaction. Elles concernent les informations d'entrée et de sortie.

Si, de ce point de vue, la filiation de TRIDENT par rapport à PAC et Arch est manifeste, sa force est d'en proposer une représentation interne, compatible d'une mise en œuvre automatique. Mais TRIDENT va plus loin : il cible aussi la génération automatique des objets d'interaction. Il définit, à ces fins, un corpus de règles ergonomiques, qu'il informatise et manipule via deux outils :

- SEGUIA [Vanderdonckt 94] est un système expert qui extrait de ce corpus de règles celles pertinentes pour la génération automatique de l'interface. Cette génération couvre la sélection et l'organisation des objets élémentaires ou composés d'interaction. Elle est potentiellement automatique mais offre aussi un mode semi-manuel ;
- SIERRA [Vanderdonckt 94] [Bodart 95a] est un navigateur hypertexte qui offre un accès structuré à ces règles.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
TRIDENT	--	●	--	●p		--	--	--	--	--	--	--	--	--	--	--	--
		--															

Figure 41 : Caractérisation de TRIDENT dans ESPRIT.

Retenons les deux originalités suivantes :

- d'une part, l'accessibilité des règles ergonomiques ;
- d'autre part, la sensibilité de TRIDENT aux aspects architecturaux.

SIROCO-Guide répond, de même, à un modèle d'architecture explicite.

1.3.7. SIROCO-Guide

Description

SIROCO-Guide [Normand 92] est un prototype de générateur de code. Il travaille en entrée sur une description de l'application en langage SIROCO et fournit, en sortie, du source Guide, un langage à objets répartis. Le langage SIROCO est un formalisme de spécification conceptuelle d'une interface. Il cible les données et opérations des Concepts de l'Application, ceci relativement aux tâches utilisateur :

- un Concept de l'Application est « un objet sémantique du système interactif » (p 98). Il est défini dans une approche à objets ;
- les tâches utilisateur relèvent de la dimension d'utilisation. Elles sont obtenues par l'application de la méthode MAD [Sebillotte 94].

SIROCO concilie ces deux dimensions via les notions d'Espace de Travail et de Perspective :

- un Espace de Travail est un « lieu d'activité virtuel offrant les éléments nécessaires à la réalisation d'une ou plusieurs tâches » (p 99) ;
- une Perspective définit « le mode d'accès à un Concept de l'Application depuis un Espace de Travail. Ce mode d'accès comporte deux grands aspects : d'une part, un filtre sur les éléments du Concept, et d'autre part, la description du mode d'utilisation des éléments retenus » (p 117).

A partir de ces spécifications conceptuelles, SIROCO-Guide génère automatiquement le code orienté-objet de l'IHM graphique. Il est garant d'un modèle d'architecture logicielle explicite. L'une des retombées de cette organisation rationnelle, systématique et orientée-objet est l'opportunité de personnalisations manuelles du code produit. A l'inverse, la structuration en Espaces de Travail est manuelle : cette délégation présente un risque de non conformité vis-à-vis du modèle de tâches. ADEPT, analysé en 1.3.1, en est, au contraire, garant.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
Siroco-Guide	--	●	--	●		--	--	--	--	--	--	--	--	--	--	--	--
		--															

Figure 42 : Caractérisation de SIROCO-Guide dans ESPRIT.

En décorrélant et conciliant les dimensions d'utilisation et dimensions de fonctionnement, c'est finalement le domaine que Siroco réhabilite. Il lui accorde une existence propre et ne le limite plus seulement à la perspective qu'en ont les tâches. AME s'inscrit dans cette même lignée.

1.3.8. AME

Description

AME (Application Modelling Environment) [Märtin 96] est un environnement de développement orienté-objet. Il cible les phases de modélisation, construction et implémentation :

- la modélisation porte sur le domaine. Les méthodes suivies sont OOD (Object Oriented Design) [Shlaer 92] et OMT [Rumbaugh 95]. AME offre, à ce niveau, une aide à l'analyse et à la conception. OODEVELOPTOOL reprend, par exemple, les éléments de modélisation de la méthode OOA/OOD. Il en offre des éditeurs de classes, attributs et méthodes, ainsi qu'un support à OOD pour la conception de l'interface, des pré et post conditions. Il y associe un système de gestion de versions et des mécanismes de documentation logicielle. ODE en est un autre exemple. Il permet notamment l'obtention de modèles OOA à partir de formalisations OMT. Tous élaborent des modèles, traduits par AME en une représentation interne, et transmis à la phase de construction ;
- la construction affine, par génération automatique, les modèles précédents en des spécifications détaillées : objets d'interaction, comportement dynamique et organisation de l'interface y sont considérés. Les spécificités de la plate-forme cible et de l'utilisateur sont ici prises en compte. Notons que le développeur peut intervenir sur les éléments générés ;
- enfin, l'implémentation génère code source et exécutable, sur la base des spécifications ainsi stabilisées.

D'un point de vue méthodologique, ces trois niveaux s'intègrent dans un processus de développement tout aussi formalisé. Il reconnaît cinq activités : l'analyse, les conceptions globale et détaillée, la génération des spécifications du comportement de l'interface, l'adaptation à des utilisateurs et environnements spécifiques puis les génération et compilation d'un programme en langage cible.

- L'analyse consiste en l'élaboration d'un modèle OOA. Pour chaque objet du domaine, sont précisés attributs et méthodes. Des connexions entre classes peuvent enrichir ces descriptions : agrégations, associations ou liens dynamiques. Les outils graphiques OODEVELOPTOOL et ODE facilitent ces spécifications.
- La conception globale définit la structure orientée objet de l'application. Elle relève du niveau construction. Le modèle OOA y est automatiquement étendu en un modèle OOD qui inclut la définition de la structure de la fenêtre applicative ainsi que la hiérarchie de commandes et de menus. Cette traduction dépend de l'environnement. Des interventions manuelles sont toujours possibles : complément de spécifications, optimisation (efficacité ou utilisabilité par exemple), etc.
- La conception détaillée attribue un objet d'interaction abstrait à toute classe du modèle OOD. Cette association s'effectue via un système à base de règles « si-alors-sinon ».
- La spécification automatique de la dynamique est le dernier composant du niveau construction. Il projette les caractéristiques des classes OOD sur les objets d'interaction.
- La génération de l'agencement et du style manipule des objets d'interaction concrets. Les préférences opérateur et contraintes applicatives peuvent orienter cette mise en correspondance. Pour ce qui est des ressources de présentation, à savoir fontes ou couleurs par exemple, elles sont héritées des objets pères, mais restent configurables par le concepteur. Des simulations permettent de visualiser l'interface obtenue.
- La génération de code relève du niveau implémentation. Elle exploite le modèle ainsi capitalisé et en génère un programme C++.

En conclusion, retenons une caractéristique essentielle de cet environnement, à savoir, sa flexibilité : il implique le concepteur dans toute activité de décision et se soumet à son avis.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.			Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.		S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
AME	--	--	--	--	●	--	--	--	--	●	--	--	--	--	--	--	--	--	--

Figure 43 : Caractérisation de AME dans ESPRIT.

Ainsi s'achève les revue et caractérisation des outils de construction à dominante proactive. Nous nous consacrons maintenant aux approches, par nature, réactives. Nous commençons par les instances prédictives.

2. Outils à dominante réactive prédictive

Les outils de cette nature s'affranchissent, par définition, de tout utilisateur. Ils évaluent l'application en regard de données d'entrée. Ces données sont diverses :

- potentiellement subjectives, issues, par exemple, d'évaluateurs experts ;
- ou objectives lorsqu'elles sont, par exemple, inférées par l'outil sur la base du code source. USAGE s'inscrit dans cette lignée : il travaille sur des fichiers générés par l'environnement de construction. C'est un outil d'analyse, que nous étudions dans la section suivante avant d'aborder les outils de critique.

2.1. Outils d'analyse

Pour la plupart, les outils d'analyse dressent une passerelle entre deux mondes trop souvent hermétiques : la psychologie et la technologie. Si, dans le principe, le bienfait de ces échanges est pourtant reconnu, en pratique deux obstacles s'opposent à une culture multidisciplinaire :

- d'une part, la complexité technologique croissante qui accapare les développeurs ;
- et d'autre part, l'opacité des méthodes formelles d'analyse cognitive, qui dissuade d'un investissement supplémentaire.

Pour [Byrne 94], la solution s'exprime en termes d'automatisation : il s'agit d'intégrer les modèles psychologiques dans l'arsenal technique des développeurs. USAGE cible le modèle GOMS et le greffe à l'environnement UIDE. Nous l'étudions dans la section suivante.

2.1.1. USAGE

Description

USAGE (the UIDE System for semi-Automated GOMS Evaluation) [Byrne 94] propose, sur la base du modèle GOMS [Card 83], une évaluation semi-automatique des interfaces élaborées dans l'environnement UIDE. L'évaluation porte sur les temps d'apprentissage et d'exécution : ils sont calculés sur la base de scénarios opérateur dont la spécification incombe à l'utilisateur.

D'un point de vue mise en œuvre, USAGE travaille sur des fichiers ASCII de description des tâches. Ces fichiers sont générés par UIDE et décrits au niveau technique d'interaction. Un traducteur en fournit un modèle NGOMSL (Natural GOMS Language). NGOMSL est un formalisme dédié aux analyses GOMS. Ce modèle alimente ensuite un interpréteur NGOMSL. Sur la base des scénarios opérateur, l'interpréteur évalue les temps précités.

Ainsi, moyennant l'élaboration de modèles NGOMSL concurrents, USAGE permet une comparaison objective des différentes alternatives. Mais l'automatisation reste partielle : le choix des scénarios incombe à l'utilisateur et s'avère déterminant. Disons même qu'a priori, seuls les utilisateurs finaux ou des études statistiques sont à même de les définir de façon représentative.

Enfin, notons quelques limites actuelles d'USAGE :

- l'incapacité de l'outil à gérer des procédures équivalentes pour une même tâche, par exemple : les raccourcis clavier associés aux menus ;
- la fermeture de l'outil en termes de techniques d'interaction : seules celles prises en charge par le traducteur sont candidates à l'automatisation ;
- et enfin la contrainte de formuler les scénarios au niveau d'abstraction du modèle des tâches d'UIDE.

D'un point de vue limites, USAGE se heurte à celles inhérentes au modèle GOMS, à savoir [Coutaz 90] :

- la non prise en compte de l'erreur humaine ;
- la « résolution de problèmes déjà résolus » [Coutaz 90] p 30 ;
- et l'absence de conseils. Ce grief ramène à la proactivité.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
USAGE	--	--	--	--	● m	--	--	--	● c	--	--	--	--	--	--	--

Figure 44 : Caractérisation de USAGE dans ESPRIT.

GLEAN qui relève de la même approche propose, de même, une version informatisée du modèle GOMS.

2.1.2. GLEAN

Description

GLEAN (GOMS Language Evaluation and ANalysis) [Kieras 95] cible globalement quatre objectifs :

- en tout premier lieu, automatiser les évaluations prédictives de type GOMS et publier des résultats d'analyse utiles et intelligibles ;
- garantir une mise en œuvre aisée par des données d'entrée simples et rapides à constituer ;
- adopter une notation lisible et compréhensible par tous, au prix, au pire, d'un faible entraînement ;
- et enfin standardiser les méthodes, en constituer des bibliothèques et ainsi favoriser une réutilisabilité.

D'un point de vue mise en œuvre, GLEAN procède par simulation : sur la base de scénarios de tâches prédéfinis, il simule l'interaction entre un utilisateur simulé et des dispositifs, eux-aussi, simulés. Il recueille ainsi des mesures statiques et dynamiques quant à l'interaction proposée : temps d'apprentissage, d'exécution, évolution de la charge de travail, etc.

Pour ce faire, GLEAN exploite trois types d'informations (Figure 45) :

- une spécification des tâches de référence, dites benchmark, s'apparentant à des scénarios ;
- le modèle GOMS formalisant la connaissance procédurale de l'utilisateur ;
- et une description statique et dynamique de l'interface (localisation, apparence et comportement des objets d'interaction).

GLEAN active alors deux moteurs d'inférence :

- un évaluateur/interpréteur GOMSL établit, en tout premier lieu, des mesures prédictives statiques, comme le temps d'apprentissage par exemple. Il complète ces mesures d'informations dynamiques inférées par simulation : concrètement, en regard des tâches prescrites, il exécute les méthodes idoines et recueille les informations recherchées (temps d'exécution, par exemple). Il entretient aussi, au fil de l'interaction simulée, un modèle de l'utilisateur pour notamment mesurer sa charge de travail. Tout événement généré au cours de cette interaction est transmis à un deuxième moteur : l'interpréteur du comportement de l'interface ;

- ce deuxième interpréteur maintient un état de l'interface (état des objets, position du curseur, etc.) qu'il consigne en une animation temps réel de l'interaction proposée : il reflète ainsi les retours d'information et permet d'en détecter visuellement les insuffisances ou inadéquations.

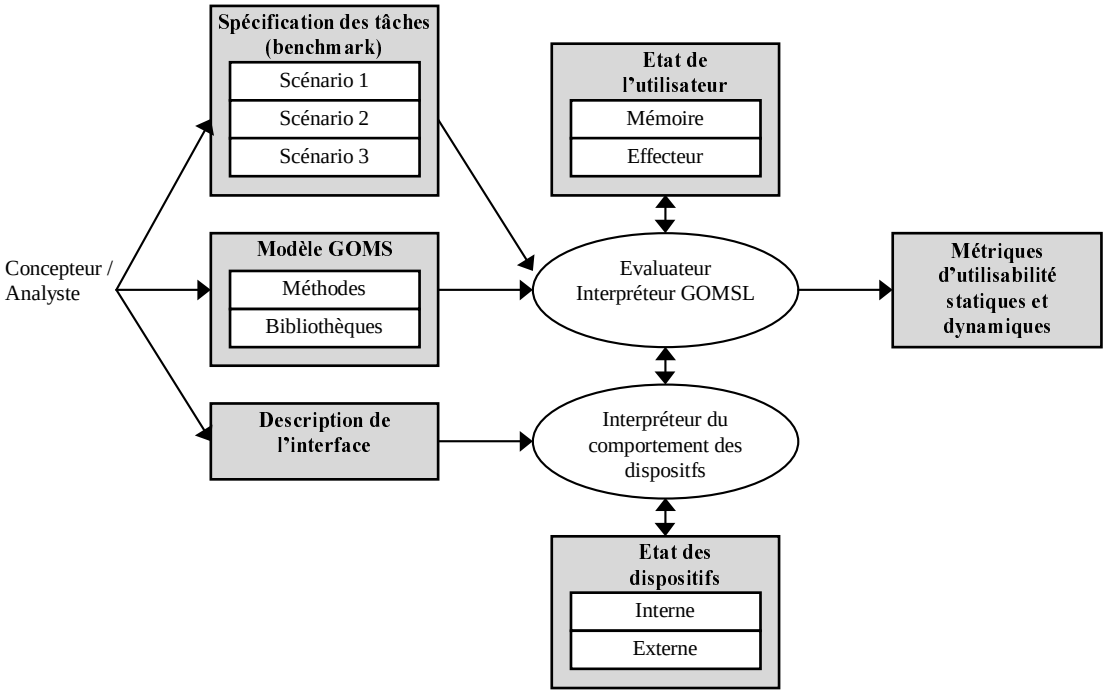

Figure 45 : Structure de GLEAN : le concepteur fournit, en entrée, un modèle GOMS, une description de l'interface et des scénarios de référence ; GLEAN simule alors l'interaction et produit, en retour, une animation de l'interface ainsi que des métriques d'utilisabilité statiques et dynamiques.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
GLEAN	-- ● -- --	-- -- -- --	--	--	● m	--	--	--	● c	--	--	--	--	--	--	--	

Figure 46 : Caractérisation de GLEAN dans ESPRIT.

Si dans USAGE et GLEAN, les critères d'évaluation sont implicites, induits par GOMS, ils sont, au contraire, explicites et accessibles à l'opérateur dans AIDE. Nous en proposons une description dans la section suivante.

2.1.3. AIDE

Description

AIDE (semi-Automated Interface Designer and Evaluator) [Sears 95] propose une assistance au développeur pour l'obtention, à coût réduit, d'interfaces esthétiques, orientées tâche. Il promeut, à ces fins, un prototypage intensif qu'il assortit d'un environnement de développement. Cet environnement allie construction et évaluation sur la base de métriques. Une métrique associe, par définition, une valeur numérique à un critère. Les critères portent ici sur les spécifications externes de l'interface. Concrètement :

- à partir d'une sélection et pondération de métriques, AIDE calcule automatiquement l'agencement optimal des informations à l'écran ;

- le développeur peut alors interagir et modifier la solution proposée ;
- il peut ensuite explicitement demander l'évaluation de son alternative personnelle.

Les métriques aujourd'hui prises en charge sont l'efficacité, l'alignement, l'équilibre et le respect de contraintes :

- l'efficacité dépend de la tâche : elle condamne, au regard de la loi de Fitts [Card 83], les déplacements du curseur imposés par l'organisation des objets d'interaction ;
- l'alignement et l'équilibre contrôlent respectivement les alignements et équilibres horizontaux et verticaux du rendu graphique ;
- les contraintes portent sur l'agencement des objets d'interaction. Elles consignent d'éventuelles exigences en la matière, de la part du développeur. AIDE évalue la satisfaction relative de ces différentes contraintes.

D'un point de vue mise en œuvre, AIDE suppose :

- l'énumération des widgets composant les spécifications externes de l'interface (nom et taille) ;
- des informations de fréquence quant aux déplacements du curseur (transitions entre widgets), ceci en regard d'une analyse de l'activité ;
- la localisation du curseur à l'initialisation ;
- la formulation éventuelle de contraintes d'agencement absolues ou relatives ;
- et la pondération de ces différentes métriques.

Ainsi, AIDE incarne résolument les approches itératives. Il milite pour une détection précoce de problèmes potentiels d'utilisabilité et propose, à ces fins, une structure d'accueil favorisant les cycles de construction, évaluation, modification et réévaluation. Si cette structure est, par essence, réactive, elle est potentiellement proactive, par l'implication de ces métriques en phase de construction. Mais, cette proactivité est du ressort du développeur : l'exploitation du fruit de l'analyse lui incombe. Elle est, en revanche, partiellement automatisée dans les outils à base de critique : nous les étudions dans la section suivante.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
AIDE	--	--	--	--	● m, j	--	--	--	● p	--	--	--	--	--	--	--	

Figure 47 : Caractérisation de AIDE dans ESPRIT.

2.2. Outils de critique

Si les éditeurs de présentation offrent une liberté totale en matière de choix et d'agencement de widgets, ils délèguent, en pratique, au développeur le respect de toute règle ergonomique. Les outils à base de critique amorcent un début de civilisation : ils s'apparentent typiquement à des systèmes à base de connaissances et vérifient la conformité de l'interface en regard des informations embarquées. CritiGUI en est un premier exemple.

2.2.1. CritiGUI

Description

CritiGUI [Nitsche-Ruhland 95] s'attache à l'ergonomie de surface. Il définit, à ces fins, un module d'expertise contenant les règles d'usage en matière de taille et de couleur des objets d'interaction. Un analyseur procède à une analyse lexicale du code de l'application pour récupérer les caractéristiques des différents widgets. Un module de critique intervient alors en regard des règles ergonomiques spécifiées dans le module d'expertise.

Si CritiGUI souffre aujourd'hui d'un faible niveau d'abstraction, il devrait porter demain sur les traits sémantiques : contenu et regroupement d'options au sein de menus par exemple.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
CritiGUI	--	--	--	--	--	● j	--	--	● c	--	--	--	--	--	--	--

Figure 48 : Caractérisation de CritiGUI dans ESPRIT.

Comme CritiGUI, CHIMES traite des couleurs. Mais, à sa différence, il couvre le niveau conseil.

2.2.2. CHIMES

Description

CHIMES [Jiang 92] (extrait de [Farenc 97] pp 36-39) est un système expert dédié à l'évaluation ergonomique d'interfaces développées sous OSF/Motif. Il vérifie la conformité de ces interfaces vis-à-vis du guide de style Motif. Il cible, plus particulièrement, les recommandations relatives à l'usage de la couleur. Il est composé de cinq modules principaux :

- un module d'acquisition des données charge la description de l'interface et la transmet, sous une forme exploitable, au vérificateur de conformité ;
- le vérificateur de conformité convertit ces informations en faits et en alimente la base de connaissances ;
- la base de connaissances contient les faits et règles ergonomiques. Ces dernières portent sur les niveaux lexical et syntaxique ;
- le générateur de conseils infère des axes d'amélioration ;
- l'interface permet à l'utilisateur d'initialiser l'évaluation. Elle publie les résultats sous la forme d'une liste de conseils, référençant les éventuels objets d'interaction concernés.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
CHIMES	--	--	--	--	--	--	● j	--	● c	--	--	--	--	--	--	--

Figure 49 : Caractérisation de CHIMES dans ESPRIT.

KRI présente un profil analogue mais cible les interfaces de type formulaires.

2.2.3. KRI

Description

KRI (Knowledge-based Review of user Interfaces) [Löwgren 90] est un système à base de connaissances pour une évaluation ergonomique d'interfaces opérateur. Il cible les niveaux critique et conseil pour des interfaces de type formulaires où les menus sont accédés via des touches de fonction. Il s'articule, pour ce faire, autour de trois composants principaux :

- une base de connaissances consigne, sous forme de règles, le savoir du système en matière d'évaluation ergonomique. Ce savoir combine connaissances subjectives d'experts et recommandations ergonomiques issues de guides de style. Seuls les niveaux lexical et syntaxique sont considérés ;
- une base de données stocke ces mêmes règles, sous une fome textuelle, académique, dépouillée de toute interprétation ;

- enfin une classification des techniques d'interaction, selon notamment le type de dialogue appliqué, permet, pour l'essentiel, de structurer la base de connaissances. Elle sert aussi au concepteur pour orienter l'évaluation vers des pôles d'intérêt donnés.

D'un point de vue mise en œuvre, KRI travaille sur une représentation formelle de l'interface utilisateur. Il requiert du concepteur une spécification de son plan de résolution. Une fois ces pôles d'intérêt exprimés, l'évaluation proprement dite prend place : les règles de la base de connaissances sont exploitées en chaînage avant. Les anomalies sont consignées sous forme de messages. Des explications complémentaires et axes de progrès sont disponibles. Il est notamment possible de consulter la règle ergonomique à l'origine du conflit : la base de données est alors sollicitée.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
KRI	--	--	--	--	--	● j	● j	--	● cc	--	--	--	--	--	--	--

Figure 50 : Caractérisation de KRI dans ESPRIT.

Dans la même lignée que KRI, SYNOP [Kolski 89] adopte une structuration de la base de connaissances. Mais, à sa différence, il prolonge la critique, non plus en conseils mais en mise en œuvre de corrections : ce service ne cadre pas avec nos objectifs. Etudions, en revanche, ERGOVAL qui implique aussi une base de connaissances mais en soigne particulièrement l'utilisabilité.

2.2.4. ERGOVAL

Description

ERGOVAL [Farenc 97] [Farenc 96] [Liberati 95] [Barthet 94] [Senach 90] est un système d'aide à l'évaluation ergonomique d'interfaces graphiques. Il traite de la présentation statique, aux niveaux lexical et syntaxique. Il s'adresse aux informaticiens et soigne, en conséquence, l'utilisabilité des résultats inférés. L'optimisation porte sur :

- le temps d'accès aux règles ;
- la facilité d'utilisation de l'outil ;
- les complétude et maintenance de la base de connaissances.

Pour ce faire, ERGOVAL implique trois sous-systèmes :

- une base de connaissances contenant les règles ergonomiques issues de la littérature ;
- la décomposition structurelle de l'interface ;
- et une typologie des objets graphiques permettant de factoriser la définition de règles en regroupant les objets candidats à leur application.

Si ERGOVAL est aujourd'hui un outil, c'est aussi, et avant tout, une méthode de structuration des règles ergonomiques. Cette méthode établit une passerelle entre les connaissances ergonomiques et informatiques et constitue une avancée fondamentale pour une connexion à un UIMS. VDDE en est un exemple.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
ERGOVAL	--	--	--	--	--	--	● j	--	--	● cc	--	--	--	--	--	--	--

Figure 51 : Caractérisation de ERGOVAL dans ESPRIT.

2.2.5. VDDE

Description

VDDE (Voice Dialog Design Environment) [Sumner 97] cible la construction et l'évaluation d'applications de type téléphonie :

- la construction couvre :
  - la définition fonctionnelle de l'application ;
  - la spécification des messages audio, à l'intention des appelants ; un module de simulation en offrant un rendu précoce ;
  - la création des messages vocaux, invitant l'utilisateur à presser certaines touches de l'appareil ;
- l'évaluation repose, quant à elle, sur un système de critique. Il intègre des critères et stratégies d'évaluation :
  - les critères portent, d'une part, sur les complétude et cohérence intra et inter solution(s), d'autre part, sur le respect de consignes métier ;
  - les stratégies prévoient les procédures analytiques et comparatives : les premières examinent la solution courante en regard des critères précités, les secondes les opposent, en revanche, à des versions concurrentes.

D'un point de vue mise en œuvre, VDDE offre trois politiques d'intervention, non exclusives. Ces politiques fixent, non seulement, les instants d'activation du système de critique mais aussi sa portée. Cette portée s'exprime en termes de couverture de l'application et de critères pris en charge :

- l'intervention active suppose un système de critique actif en permanence et à l'affût d'interventions opérateur. La critique est alors localisée à ces évolutions, ceci pour une perturbation moindre du concepteur ;
- l'intervention passive inhibe, à l'inverse, le système de critique : il ne s'active que sur déclenchement explicite de la part du développeur. La critique porte alors sur le produit complet ;
- la stratégie à base d'unités conceptuelles porte exclusivement sur les menus vocaux. Le système s'active lorsque le concepteur change de pôle d'intérêt : achèvement d'un menu et passage au suivant.

Un module d'explication visualise, sur sélection d'une critique, la portion de code concernée. Il présente complémentairement la partie du guide de styles à l'origine du conflit, ceci grâce à une connexion à ce corpus de règles. Le concepteur peut alors ignorer une critique ou, au contraire, la traiter. Il peut documenter tout choix de conception.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
VDDE	--	--	--	--	--	--	● j	--	● cc	● cc	--	--	--	--	--	--	--

Figure 52 : Caractérisation de VDDE dans ESPRIT.

Ainsi s'achèvent cette revue et caractérisation des outils d'analyse et de critique offrant une évaluation prédictive de l'interaction proposée. Nous nous attachons maintenant aux approches expérimentales.

3. Outils à dominante réactive expérimentale

Nous organisons, de même, cette section en deux parties selon le service d'analyse ou de critique offert par l'outil. Nous les envisageons dans cet ordre.

3.1. Outils d'analyse

Nous commençons ici la revue par le système pionnier en la matière : Playback.

3.1.1. Playback

Description

Le système Playback [Neal 83] est un système de capture d'actions clavier. Il repose sur un dispositif dédié, le « lab computer », inséré entre le clavier et le système hôte. Ce dispositif détecte et date toute action clavier. Il les enregistre sur disque. Les fichiers sont alors disponibles pour des exploitations manuelles ou informatisées. Playback propose en la matière :

- une analyse statistique des occurences d'actions ;
- un mécanisme de rejeu pour reproduire la session de travail originelle.

D'un point de vue mise en œuvre, Playback souffre :

- d'une capture à faible niveau d'abstraction ;
- de la dépendance en dispositifs matériels dédiés ;
- et de « l'oubli » en 1983 ... des interactions par manipulation directe.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE												
					Prédictive					Expérimentale							
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères		
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ	
Playback	--	--	--	--	--	--	--	--	--	--	●	● m	--	--	--	--	

Figure 53 : Caractérisation de Playback dans ESPRIT.

A l'inverse, le système d'Hammontree couvre les entrées clavier et souris.

3.1.2. Système d'Hammontree

Description

Le système d'Hammontree [Hammontree 92] est un système de capture couvrant les actions clavier et souris. Ces événements sont tous estampillés d'une date. Hammontree complète ces données numériques par un enregistrement audiovisuel analogique. D'un point de vue exploitation, il propose les mécanismes d'assistance suivants :

- un programme de filtrage d'événements permet d'extraire du fichier originel des données pertinentes en regard de critères personnalisés ;
- un analyseur multimedia établit le lien entre événements numériques et enregistrements analogiques : sur la base des datations, il localise tout événement sur la bande et en propose une visualisation ;
- un système d'annotations verbales permet enfin d'associer des commentaires oraux à des segments de la bande. Il procède par numérisation de la parole et s'utilise de pair avec l'analyseur.

D'un point de vue utilisabilité, le système d'Hammontree se heurte à deux difficultés :

- la capture se limite aux widgets prédéfinis ;

- elle nécessite, par ailleurs, une instrumentation manuelle du code.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
Hammontree	--	--	--	--	--	--	--	--	--	--	●	●	--	--	--	--

Figure 54 : Caractérisation de Hammontree dans ESPRIT.

Dans la même lignée qu'Hammontree, MRP nécessite, de même, une instrumentation manuelle du code. Il rehausse, en revanche, le niveau d'abstraction en traitant des aspects syntaxiques.

3.1.3. MRP

Description

Partant de l'hypothèse que les répétitions de schémas d'actions sont manifestes du comportement opérateur, MRP (Maximal Repeating Pattern) [Siochi 91] cible la détection de tels schémas récurrents. Il procède, pour ce faire, à une capture du comportement opérateur : cette capture porte sur les actions physiques et s'effectue sur la base d'une instrumentation manuelle du code. Les fichiers enregistrés sont ensuite analysés pour la détection de motifs récurrents. L'exploitation est alors du ressort du concepteur : il peut, par exemple, rajouter des macros commandes pour augmenter la performance de l'utilisateur et ainsi peut-être réduire les risques d'erreur.

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
MRP	--	--	--	--	--	--	--	--	--	--	●	● m	--	--	--	--

Figure 55 : Caractérisation de MRP dans ESPRIT.

Dans cette même lignée, EMA propose aussi une détection automatique de schémas comportementaux récurrents. Mais, ces schémas, loins d'être anodins, caractérisent, en fait, des déviations comportementales. Aussi, dans les faits, EMA s'apparente-t-il à un outil de critique.

3.2. Outils de critique

Les outils de critique reposent, en pratique, sur une analyse. Les critères alors exploités reflètent et ciblent les défauts traqués. EMA en est un exemple.

3.2.1. EMA

Description

EMA (Evaluation par Mécanisme Automatique) [Balbo 94] propose une évaluation ergonomique automatique à partir de données comportementales numérisées. Si l'évaluation relève, par essence, de l'analyse, elle s'apparente, en pratique, à une critique de polarité négative. Elle cible la détection d'anomalies comportementales et se dote, à ces fins, de quatre critères de déviation : les rupture, annulation immédiate, répétition et invalidité. Elle exploite, pour ce faire, trois types d'informations :

- une base de profils comportementaux génériques, modélisés sous forme de règles ;
- une description formelle des tâches prévues, spécifiques au logiciel testé ;

- les données comportementales, enregistrées au fil de l'interaction, ceci grâce à une instrumentation manuelle du code.

En sortie, EMA fournit des annotations, consignées dans le fichier de capture et le modèle des tâches. Ces annotations localisent les déviations constatées.

Caractérisation

OUTILS	PROACTIVITE					REACTIVITE											
						Prédictive					Expérimentale						
	Deg. Auto.		Critères			Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ		Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
EMA	--	--	--	--		--	--	--	--	--	--	●	●	●	--	--	● p

Figure 56 : Caractérisation de EMA dans ESPRIT.

Cloturons la revue sur NEIMO. Il cible les applications innovantes et fonde son analyse sur les propriétés CARE [Coutaz 95a].

3.2.2. NEIMO

Description

NEIMO [Aublet-Cuvelier 96] est un laboratoire d'utilisabilité numérique d'aide à la capture automatique et à l'analyse de données comportementales pour des situations d'interaction innovante (multimodalité, communication homme-homme médiatisée). Ses objectifs sont de deux natures :

- non pas se substituer au spécialiste mais éliminer toute tâche mécanique de capture, dépouillement de données et signalement de points singuliers en regard de cet expert ;
- offrir un support à l'évaluation formative d'interfaces prospectives, partiellement implémentées.

Pour satisfaire ce premier objectif, NEIMO propose des services de capture, d'analyse et de critique :

- la capture s'effectue à différents niveaux d'abstraction : actions physiques sur les dispositifs d'entrée, tâches du domaine élémentaires ou composées. Le niveau est choisi à la configuration de la sesssion d'expérimentation. Il nécessite une instrumentation manuelle du code ;
- l'analyse adopte les propriétés CARE [Coutaz 95a] comme critère de structuration. A la configuration, les expérimentateurs décident des modalités équivalentes pour une tâche donnée, des complémentarité et redondance autorisées pour cette même tâche ;
- la critique portera sur la détection de points singuliers, mais cette fonction n'est pas offerte à ce jour.

L'outil offre des services de rejeu numérique, augmentés de capacités de déplacement motivés par la tâche de l'analyste : changement de scénario, sélection d'un point singulier, etc. Il fournit complémentairement des renseignements quantitatifs et qualitatifs : la durée des scénarios et un indicateur quant à leur réussite.

D'un point de vue mise en œuvre, NEIMO couvre les évaluations du type Magicien d'Oz. Des compères, situés dans une salle autre que celle du sujet, simulent les fonctions manquantes du système et annotent les scénarios d'observations personnelles.

Si NEIMO est aujourd'hui générique par la variété des interfaces ciblées, le branchement de ces interfaces à l'outil de capture reste hélas encore une affaire de programmeurs. Il faudrait, à l'avenir, automatiser cette activité et définir un format de capture universel [Carraux 96].

Caractérisation

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
NEIMO	-- -- -- -- -- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	●	● m	--	--	--

Figure 57 : Caractérisation de NEIMO dans ESPRIT.

Ainsi s'achève notre revue et caractérisation des outils d'origine académique. Nous en proposons une analyse critique au chapitre suivant.

CHAPITRE VI : SYNTHESE DE L'ETAT DE L'ART ACADEMIQUE

L'analyse critique ici proposée porte sur l'état de l'art présenté au chapitre précédent. Tandis qu'une première section dresse des constats, une seconde discute des orientations phares. Une synthèse cloture enfin cette partie par l'ébauche d'une solution.

1. Constats : les outils, encore et toujours ...

Le principal sentiment qui se dégage de ces revue et caractérisation est celui d'une énergie colossale consacrée à la mise à disposition d'outils de construction et d'évaluation. Ces outils apparaissent comme un point de passage obligé, un goulot d'étranglement qui conditionne la réalisation de toute application. Nous développons ce point dans la section suivante.

Nous poursuivons alors cette prise de recul par une vision globale des outils, jusqu'ici étudiés de façon isolée : nous en dressons une carte des profils et en soulignons tendances et lacunes.

1.1. Sujet fédérateur

Les outils constituent aujourd'hui un sujet fédérateur. L'histoire d'ITS [Boies 88] laissait déjà présager ce destin : il s'agissait, au départ, de regrouper des chercheurs d'horizons divers (notamment logiciel, matériel et ergonomie) pour la réalisation d'une application innovante. Très vite, deux constats s'imposèrent et en changèrent l'orientation :

- d'une part, la nécessité de disposer de nouveaux outils logiciels performants pour accroître la productivité et garantir une haute qualité aux applications développées ;
- d'autre part, l'existence d'invariants tant en termes de rôles dans le processus de développement que de points communs dans l'interaction, et ceci malgré la centaine de sujets candidats (par exemple, le remplissage de formulaires, le choix, la manipulation de listes, la lecture de blocs d'informations, etc.).

Ainsi, ITS se réorienta très vite vers l'étude de ces aspects fondamentaux. La conjoncture économique défavorable, alliée à une effervescence technologique, ravive aujourd'hui ces préoccupations. Les outils sont ainsi devenus sujet de rassemblement : ils conditionnent productivité et innovation (Figure 58).

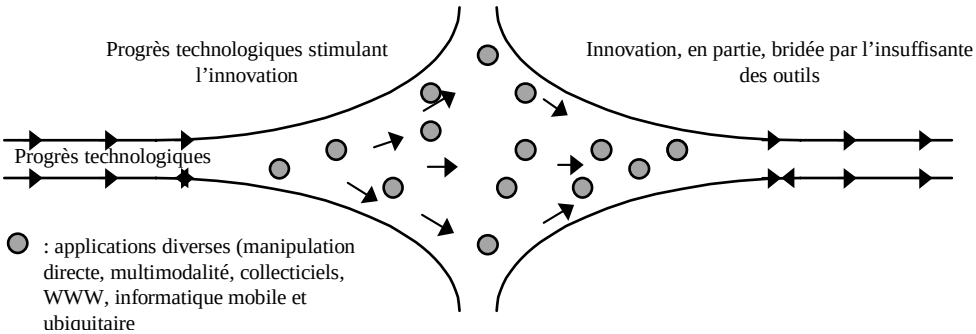

Figure 58 : Les outils ou le point de passage obligé...

Cette prise de conscience du poids des outils se traduit, en pratique, par une abondance de propositions. Nous en proposons une analyse globale et comparative dans la section suivante.

1.2. Carte des profils

Nous regroupons ici en un même tableau les profils individuels établis au chapitre précédent (Figure 59). Il en jaillit un sentiment de quasi-assignation des approches en matière de proactivité, réactivité prédictive et expérimentale :

- ITS (1988), Humanoïd (1992), ADEPT (1992), SIROCO-Guide (1992), DIANE+ (1993), TADEUS (1995), TRIDENT (1995), Mecano (1996), FUSE (1996), AME (1996), Mobi-D (1997) et ALACIE (1998) sont purement proactifs ;
- KRI (1990), CHIMES (1992), USAGE (1994), AIDE (1995), CritiGUI (1995), ERGOVAL (1997) et VDDE (1997) ciblent leur prestation sur de la réactivité prédictive ;

- Playback (1983), MRP (1991), Hammontree (1992), EMA (1994) et Neimo (1996) se consacrent à la réactivité expérimentale ;
- enfin, UIDE (1988), IDA (1993), Mastermind (1995), EXPOSE (1995) et GLEAN (1995) concilient timidement proactivité et réactivité prédictive.

La figure 59 illustre cette distribution sur support ensembliste. Cette représentation souligne l'absence d'outil couvrant notre cahier des charges.

OUTILS	PROACTIVITE				REACTIVITE											
					Prédictive					Expérimentale						
	Deg. Auto.		Critères		Deg. Auto.			Critères		Deg. Auto.					Critères	
	Capit.	S/G	Métier	ε / ψ	Ana.	Criti.	Cons.	Métier	ε / ψ	Instru.	Obs.	Ana.	Criti.	Cons.	Métier	ε / ψ
Outil requis	● ● -- --	-- -- -- --	● p	--	●	● m, j	● m, j	● p	● p	●	●	●	● m, j	● m, j	● p	● p
IDA	-- -- ● ●	-- -- -- --	● cc	● c	--	● j	--	--	● c	--	--	--	--	--	--	--
EXPOSE	● -- -- --	-- ● -- --	● cc	● cc	--	--	●	--	● cc	--	--	--	--	--	--	--
Mastermind	-- -- -- --	● ● ● ●	--	--	--	●	--	--	●	--	--	--	--	--	--	--
Mecano	-- -- -- --	● ● ● ●	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Mobi-D	-- -- -- --	● ● ● ●	--	--	--	--	--	--	--	--	--	--	--	--	--	--
ADEPT	-- -- -- --	● -- -- --	--	●p, nd	--	--	--	--	--	--	--	--	--	--	--	--
DIANE+	-- -- -- --	-- ● -- --	--	●c	--	--	--	--	--	--	--	--	--	--	--	--
ALACIE	-- -- -- --	-- ● -- --	--	●c	--	--	--	--	--	--	--	--	--	--	--	--
TADEUS	-- -- -- --	● ● -- --	--	●	--	--	--	--	--	--	--	--	--	--	--	--
FUSE	-- -- -- --	● ● -- --	--	● p	--	--	--	--	--	--	--	--	--	--	--	--
TRIDENT	-- -- -- --	-- ● ● ●	--	●p	--	--	--	--	--	--	--	--	--	--	--	--
Siroco-Guide	-- -- -- --	● ● -- ●	--	●	--	--	--	--	--	--	--	--	--	--	--	--
AME	-- -- -- --	● -- ● --	--	●	--	--	--	--	--	--	--	--	--	--	--	--
USAGE	-- -- -- --	-- -- -- --	--	--	● m	--	--	--	● c	--	--	--	--	--	--	--
GLEAN	-- ● -- --	-- -- -- --	--	--	● m	--	--	--	● c	--	--	--	--	--	--	--
AIDE	-- -- -- --	-- -- -- --	--	--	● m, j	--	--	--	● p	--	--	--	--	--	--	--
CritiGUI	-- -- -- --	-- -- -- --	--	--	--	● j	--	--	● c	--	--	--	--	--	--	--
CHIMES	-- -- -- --	-- -- -- --	--	--	--	--	● j	--	● c	--	--	--	--	--	--	--
KRI	-- -- -- --	-- -- -- --	--	--	--	● j	● j	--	● cc	--	--	--	--	--	--	--
ERGOVAL	-- -- -- --	-- -- -- --	--	--	--	● j	--	--	● cc	--	--	--	--	--	--	--
VDDE	-- -- -- --	-- -- -- --	--	--	--	● j	--	● cc	● cc	--	--	--	--	--	--	--
Plavback	-- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	● m	--	--	--	--
Hammontree	-- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	●	--	--	--	--
MRP	-- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	● m	--	--	--	--
EMA	-- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	●	●	--	--	● p
NEIMO	-- -- -- --	-- -- -- --	--	--	--	--	--	--	--	--	●	●	● m	--	--	--

Figure 59 : Compilation des caractérisations individuelles pour une perception globale et comparative de l'offre.

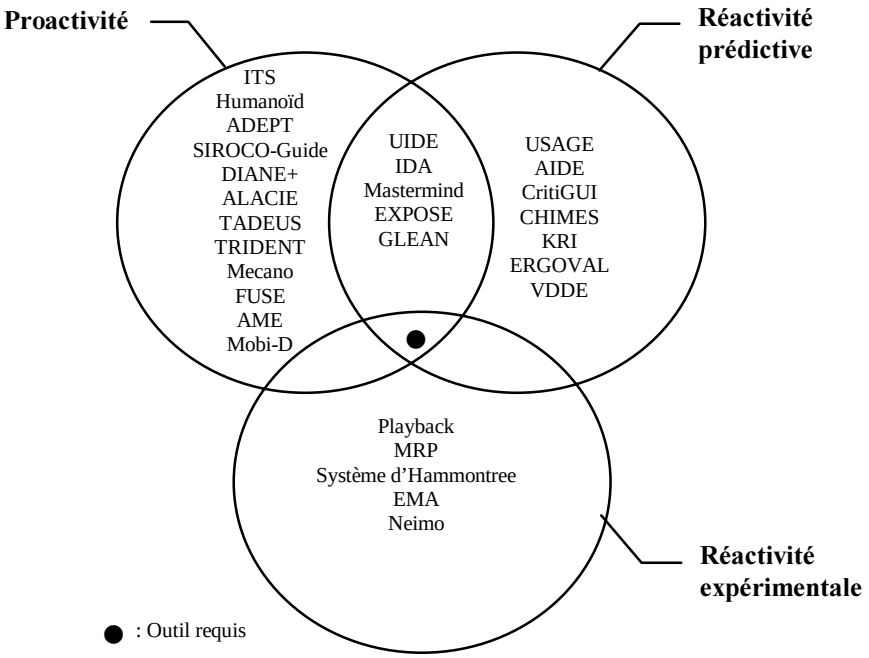

Figure 60 : Distribution des outils selon leur prestation proactive, réactive prédictive ou réactive expérimentale.

S'il est maintenant certain que l'outil requis n'existe pas, il est néanmoins intéressant de cerner l'évolution des préoccupations dans le temps. Nous estampillons, à ces fins, chaque outil d'une date, ceci sur la base des publications recueillies. Notons que cette tâche est des plus difficiles, ceci surtout lorsque l'outil se veut générique, conçu dans une perspective long-terme (tel Mastermind ou ERGOVAL). Nous nous y hasardons néanmoins. La figure 61 en consigne les résultats.

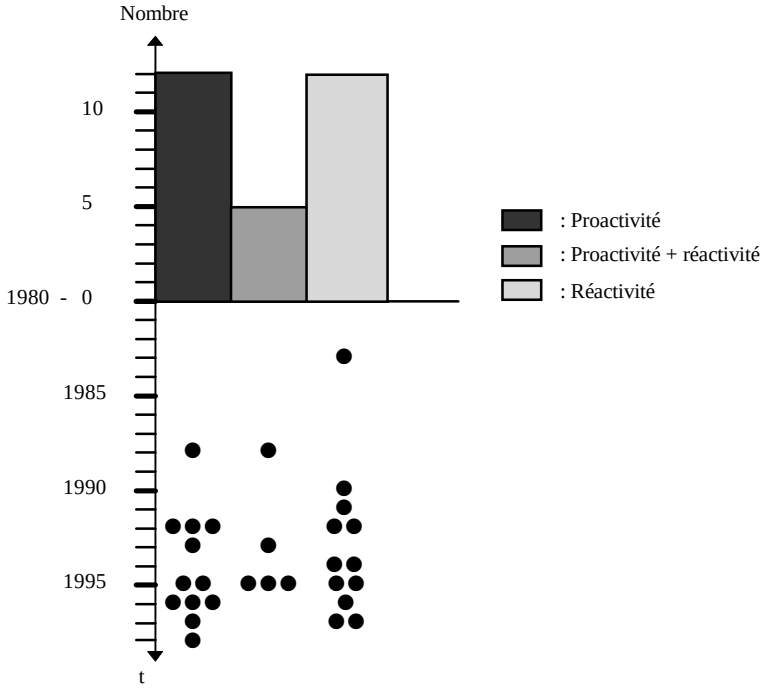

Figure 61 : Profil des recherches dans le temps.

Cette figure suscite les réflexions suivantes :

- le sentiment d'une effervescence, avec une accélération nette dans les années 92 ;

- l'avènement de la proactivité, mais tacite, dans ces années ;
- une mobilisation, à l'inverse, continue et régulière en réactivité ;
- la perception d'une frontière quasi-hermétique entre proactivité et réactivité ;
- des tentatives récurrentes, et flagrantes en 95, pour briser cette frontière et favoriser, au contraire, une osmose (Figure 62).

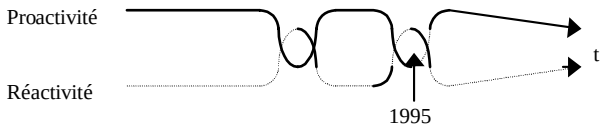

Figure 62 : Des tentatives récurrentes d'osmose entre proactivité et réactivité.

De ce paysage contrasté, nous retiendrons des approches phares :

- Mastermind qui, par la richesse de ses modélisations, fonde un socle stable de connaissances ; Mastermind qui recourt à CORBA pour une intégration explicite et standard du travail de groupe ; enfin Mastermind qui intègre l'exigence d'itérativité par un système de bouchons ;
- Mecano qui exhibe les liens entre modèles, au contraire diffus dans Mastermind ;
- TRIDENT qui réhabilite les règles ergonomiques ; TRIDENT qui garantit une architecture logicielle explicite ;
- ERGOVAL qui soigne l'utilisabilité de l'outil et structure, à ces fins, les règles ergonomiques ;
- AIDE qui associe des métriques à son évaluation ;
- AME qui assure une continuité au processus de développement, allant de l'analyse à l'implémentation ;

Soulignons enfin dans ADEPT, AME, SIROCO-Guide, FUSE et TADEUS la mise à profit de la sacro-sainte réutilisabilité : y sont impliqués TKS, OOA/OOD, OMT, MAD ou encore les réseaux de Pétri.

Nous discutons de ces orientations dans la section suivante.

2. Discussion : de l'âge de pierre à la maturité ?

Si déjà en 1986, Myers dénonçait une décorrélation trop franche entre construction et évaluation[44], force est de constater que, dix ans plus tard, la scission persiste :

- le rôle central de l'évaluation[45] est certes reconnu ; les processus de développement en attestent (cf partie I) ;
- mais aucun environnement n'accueille, de façon équilibrée, ces deux composantes.

Pourtant, des efforts sont investis en ce sens, comme en témoigne la figure 61. Sa quasi-symétrie est symptomatique des deux stratégies mises en œuvre :

- proactivité et réactivité constituent deux points d'ancrage ;
- elles mobilisent des efforts comparables ;
- et, même si elles tentent une osmose avec le courant adverse, ces tentatives restent, en pratique, marginales.

C'est, dans les faits, une supplantation de la réactivité par une proactivité intense qui est tentée. Les modélisation, génération et critique en sont des vecteurs. Nous en discutons dans les sections suivantes.

2.1. Modélisation : pour ou contre ?

C'est du paradigme du MBD dont il s'agit ici. Tandis que certains y voient un succès technologique, d'autres, plus pessimistes, ne sont toujours pas convaincus du bien fondé de l'approche. S'ils ne sont pas sceptiques, ils

[44] « UIMSs do not support evaluation. Very few user-interface tools provide any support for evaluating the user interface » [Myers 86] p 23 cité dans [Löwgren 90].
[45] « evaluation plays a major role in design, because each successive evaluation step guides the course of design activity » [Sumner 97].

attendent, en fait, des réponses aux problèmes restés ouverts : parmi eux, l'architecture et l'implication dans le processus de développement. Dans cette confusion, les practiciens restent prudents et adoptent comme seul critère le rapport coût/bénéfice. Foley, lui, est résolument convaincu d'une percée future de ce paradigme[Sukaviriya 94] :

- il capture la sémantique de l'application à différents niveaux de conceptualisation : conception haut niveau et détails d'implémentation. Il rompt ainsi avec les traditionnels générateurs d'interface qui se concentrent, à l'inverse, sur l'implémentation de la présentation [Szekely 93] ;
- il définit un socle stable et structuré de connaissances, structure d'accueil à la connexion d'outils d'exploration et d'exploitation [Märtin 96] : parmi ces derniers, les générateurs de critiques, de conseils ou de code et les services d'aide contextuelle et de personnalisation. Ces outils s'adressent alternativement aux concepteurs ou aux utilisateurs.

Les retombées sont multiples [Szekely 95] :

- un gain en cohérence et une réutilisabilité favorisés par un partage de connaissances sémantiques entre tous les composants du système ;
- une meilleure intelligibilité, et par voie de conséquence, maintenabilité de l'application ;
- une extensibilité facilitée par la forme déclarative des spécifications ;
- un support et donc une incitation à une conception précoce ;
- la promotion des développements itératifs par le caractère exécutable des modèles : même incomplets, ceux-ci offrent aux concepteurs un terrain d'expérimentation concret. Ils permettent la détection précoce de points faibles, voire d'anomalies de conception. Les rétroactions s'en trouvent facilitées.

Malgré ces points forts, le paradigme ne fait pas l'unanimité. Des inconvénients récurrents lui sont reprochés. Notamment :

- des modèles trop souvent partiels, privilégiant toujours une facette particulière [Märtin 96] : les tâches dans ADEPT, le domaine dans Puerta, la présentation dans Trident et l'application dans Mastermind ;
- des modèles trop souvent spécifiques ou dépendants des plates-formes compromettant, en final, tout espoir de réutilisation [Märtin 96] ;
- un manque de flexibilité dû à une expressivité limitée des langages de modélisation : les développeurs ne disposent pas de moyens adéquats pour contrôler toutes les caractéristiques de leurs interfaces. Si Mastermind propose, en réponse, une ouverture de son langage à des composants étrangers, la solution reste imparfaite : la modélisation de ces composants étant potentiellement incomplète, Mastermind ne pourra raisonner, à leur sujet, que de façon incomplète. De même, si le langage revendique une extensibilité, pour la faculté qu'il offre à rajouter attributs et objets, celle-ci reste lourde de mise en œuvre. Tout outil connexe devra évoluer, par ailleurs, pour exploiter ces modifications ;
- les performances limitées dues à une modélisation trop riche et à l'interprétation de celle-ci ;
- et les difficultés d'utilisation liées à l'apprentissage d'un langage de spécification dédié. Dans certains cas, la spécification s'apparente même à de la programmation.

D'où le sentiment suivant, à savoir :

- une ontologie légitime et louable, consistant à expliciter une sémantique trop souvent enfouie ;
- mais une mise en œuvre insatisfaisante et notamment pénalisante.

Nous discutons, dans la section suivante, d'un service particulier : la génération automatique.

2.2. Génération : pour ou contre ?

Si la génération automatique apparaît, de façon récurrente, dans le temps, c'est en réponse à des motivations séduisantes :

- décharger le concepteur de tâches fastidieuses pour lui permettre de se consacrer aux données d'entrée sémantiques ;
- garantir simultanément cohérence et conformité en regard de ces spécifications.

Très vite, le manque de flexibilité apparaît comme un inconvénient majeur :

- les environnements restent fermés et inadaptés à la présentation de concepts originaux ;
- ils imposent des solutions, au risque de susciter sentiments de dépossession et frustrations.

Si des outils y répondent par une modifiabilité manuelle, ils se heurtent à des risques d'incohérence. Aussi, la génération automatique, plus qu'une finalité, devient raisonnablement une commodité d'exploration de l'espace de conception [Luo 93].

Nous discutons, dans la section suivante, du bienfait des critiques.

2.3. Critique : pour ou contre ?

L'expérience montre que :

- les critiques favorisent la réflexion, notamment en termes de compromis, pour des ingénieurs confirmés[46]. Elles influencent leur comportement et les incitent, en particulier, à une anticipation[47] ;
- elles ne peuvent être assumées par un seul individu et ceci à plusieurs titres : il est, non seulement, impossible de détenir toutes les connaissances requises à l'évaluation, mais aussi d'évaluer une même solution selon différentes perspectives, parfois même incompatibles[48].

Si l'automatisation semble répondre à ces exigences, [Sumner 97] émet néanmoins deux mises en garde :

- le risque d'un détournement du système pour obtenir des esquisses de solution ;
- et la nécessité absolue de préserver la créativité humaine : cette exigence va à l'encontre de la génération automatique.

A la lumière de ces enseignements, nous définissons, dans la section suivante, de nouveaux axes de recherche : les fils directeurs d'une solution ?

3. Synthèse : esquisse d'une solution ?

En synthèse, et en référence aux propriétés CARE, rappelons que notre objectif s'exprime en termes de complémentarité, et non plus d'assignation, des outils en proactivité et réactivité. Nous retenons pour ce faire, et à la lumière de l'état de l'art, les fils directeurs suivants :

- le paradigme du MBD propice aux capitalisations, même si cet atout n'a pas encore été mis à profit ;
- l'extraction, dans Mecano, des liens entre modèles, au contraire diffus dans Mastermind ;
- la continuité du processus de développement, sous-tendue dans AME par l'intégration de la méthode OOA/OOD ;
- l'évaluation quantifiée, proposée par AIDE, sur la base de métriques ;
- la connexion à des guides ergonomiques voire l'interopérabilité avec des outils tels ERGOVAL ou Trident ;
- l'extension des évaluations ergonomiques aux évaluations métier, sur la base des connaissances capitalisées, ceci pour une mesure expérimentale de l'adéquation homme-machine

Dans un effort d'abstraction, c'est finalement une indépendance entre connaissances et plan de résolution que nous préconisons. Nous rejoignons en cela [Détienne 95a] :

---

[46] « critics do appear to promote reflection during design, through the quality of reflection, particularly in the area of considering trade-offs, appears to be deeper for more experienced designers. » [Sumner 97]

[47] « our protocol analyses indicate that such systems do influence the behaviour of designers, but often indirectly. [...]. Designers were observed anticipating the activity of the system and taking preventive steps to avoid it. Differential effects depending on the designers' level of domain experience were also observed. Overall, the system was better suited to the needs of highly experienced designers » [Sumner 97]

[48] « evaluation of design solutions is difficult for both experienced and inexperienced designers because : - in complex domains, no single person can know all the relevant criteria and constraints ; - design solutions must be evaluated from multiple, and sometimes conflicting, perspectives ; - and designers do not always recognise problematic solutions » [Sumner 97].

- les connaissances sont universelles, factuelles, indéniables, invariantes : elles définissent une norme ;
- la solution reste, à l'inverse, une proposition discutable, perfectible, orthogonale aux connaissances précitées.

Cette indépendance présente de nombreux avantages, notamment :

- l'opportunité du maintien d'une modélisation à fort niveau d'abstraction, dénuée de toute considération de l'ordre de l'implémentation ;
- par voie de conséquence, une réutilisabilité accrue de ces connaissances, dénuées de toute spécificité applicative ;
- la compatibilité entre réutilisabilité et formes de conception applicatives. Cette qualité est essentielle, l'émergence des « design patterns » [Gamma 94] en atteste : des consignes se font désormais pressantes en la matière ;
- et donc une applicabilité forte de l'outil.

Souhaitons que, par ces fils directeurs, nous satisfaisions Monsieur Schneider-Hufschmidt[49] qui déplore un milieu industriel bien trop frileux, à l'égard des approches à base de modèles [Sukaviriya 94]. Ce paradigme est, selon lui, victime d'un cercle vicieux. Sans la preuve de la pertinence et de l'efficacité de cette technique sur des applications industrielles, personne n'ose investir dans l'infrastructure nécessaire. Or, en l'absence d'un environnement adjoint, il est difficile de convaincre le milieu industriel et de l'inciter à investir dans le domaine. Inversement, comment prouver le bien-fondé de ce paradigme sans le support industriel ? Nous rompons ici ce cercle vicieux en misant sur une complémentarité proactivité/réactivité, alimentée par une capitalisation métier. Notre contribution logicielle intègre ces principes : nous lui consacrons la partie suivante de ce mémoire.

[49] de la société Siemens SA à Munich.

PARTIE III : CONTRIBUTION LOGICIELLE : CATCHIT

« L’homme n’est heureux que de vouloir et d’inventer » Alain.

En l’absence de solution industrielle ou académique répondant à notre cahier des charges, nous proposons CatchIt, un environnement de développement basé sur une modélisation des connaissances métier. Cet acronyme signifie « Critic-based Automatic and Transparent tool for Computer-Human Interaction Testing ». Il exprime la priorité de l’outil, à savoir l’évaluation de l’interaction Homme-Machine. Si CatchIt propose une aide à la conception et un support à l’évaluation prédictive, c’est sur l’évaluation expérimentale qu’il met l’accent. Sa finalité consiste à détecter et expliquer tout problème potentiel d’utilisabilité, observé sur des experts du domaine, placés en conditions opérationnelles. Autrement dit, il se propose d’enrichir, d’un point de vue opérationnel, les évaluations logicielles, trop souvent menées de façon purement fonctionnelle.

Après en avoir respectivement décrit les principes et la mise en œuvre dans les chapitres VII et VIII, nous illustrons l’outil, sur une application réelle, au chapitre IX. Nous proposons alors, en conclusion, une analyse critique de l’approche : nous en soulignons originalités et limites. Commençons par la description des principes de CatchIt.

CHAPITRE VII : PRINCIPES

Si CatchIt jaillit de la confrontation entre, d'une part, notre cahier des charges et, d'autre part, les enseignements des recherches académiques, sa philosophie reste aujourd'hui unique et peut-être, en cela, déroutante. Aussi, et en tout premier lieu, présentons-nous, de façon globale et synthétique, la philosophie de l'outil : c'est l'objet de la première section. Nous nous consacrons alors à la description de ses proactivité et réactivité.

1. Philosophie de CatchIt

CatchIt [Calvary 97c] s'inscrit dans la lignée du paradigme du «model-based user interface design». Il propose une structure d'accueil générique centrée sur une modélisation réutilisable des connaissances métier. Mais s'il adopte ce paradigme, ce n'est pas, en premier lieu, dans une perspective de génération de l'application à partir de spécifications formelles : il privilégie la souplesse d'utilisation et rejette tout carcan privant le concepteur d'une quelconque liberté. Aussi, au lieu d'y voir une opportunité en termes de génération, il considère cette modélisation comme une référence représentative du domaine opérationnel traité, référence candidate à la réutilisation logicielle, d'une part, et, faisant autorité en matière d'évaluation, d'autre part. Ainsi, support au caractère proactif et réactif de CatchIt, cette référence fait office de norme réutilisable : elle est dite «modèle normatif ». La section 2 lui est consacrée.

La proactivité de l'outil provient du cadre de réflexion offert au concepteur par la mise à disposition de connaissances métier réutilisables. Modélisées dans un langage orienté-objet, ces connaissances formalisent non seulement la description du domaine opérationnel considéré, mais expriment aussi le savoir-faire des opérateurs. Documentées pour une meilleure traçabilité et réutilisabilité, elles procurent au concepteur une grille d'analyse auto-explicative. Libre à lui de réutiliser ces unités de compétence ou de les redévelopper selon ses contraintes applicatives : langage, plate-forme, etc. Par analogie, l'application ainsi obtenue est dite « modèle applicatif ».

Si la proactivité reste informelle, la réactivité s'intègre, au contraire, de façon explicite et organisée dans la structure d'accueil. Elle repose sur une connexion déclarative entre l'application à tester[50] et le modèle normatif du domaine[51] traité. Cette mise en correspondance incite le développeur à une prise de recul par rapport à sa conception : il doit concrètement identifier dans l'application les objets et méthodes implémentant les éléments opérationnels stipulés dans la référence. Tout écart suscite réflexion et prise de décision : les choix de conception peuvent être maintenus, mais alors assumés en connaissance de cause, avec justificatif à l'appui. Bien entendu, si l'application dérive de la référence, la connexion est immédiate, quasi-bijective. La complexité du tissage de liens reflète ainsi l'écart entre les deux modèles : CatchIt la mesure et quantifie, sur cette base normative, la qualité du code applicatif obtenu. Si cette analyse favorise l'autocritique et augmente l'application de sa sémantique opérationnelle, elle ne reste qu'une étape transitoire dans la mise en œuvre d'une évaluation expérimentale : l'ultime objectif de CatchIt consiste à vérifier l'adéquation homme-machine sur des experts du domaine placés en conditions opérationnelles.

Dans un objectif de transparence et d'automatisation, CatchIt exploite, de façon autonome, les points d'ancrage entre l'application à tester et la référence métier, c'est-à-dire entre les modèles applicatif et normatif (Figure 63). Si une première évaluation prédictive vérifie les correction et complétude des liens tissés, l'évaluation expérimentale s'intéresse au comportement d'experts du domaine en interaction avec l'application. Tout événement applicatif est propagé, de façon automatique et invisible vers la référence qui, selon la situation courante, infère le comportement attendu. CatchIt confronte alors ces comportements prescrit et effectif et établit des critiques de l'interaction proposée. Autrement dit, CatchIt médicalise l'évaluation de l'interaction Homme-Machine (Figure 63) : il propose une auscultation externe de l'application. L'espionnage de l'interaction, et notamment l'instrumentation du code, sont assumés par CatchIt, de façon totalement transparente (cf chapitre III).

[50] Ou modèle applicatif.
[51] C'est-à-dire le domaine dont émane l'application.

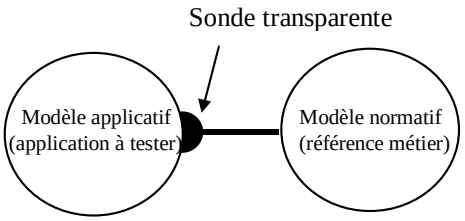

Figure 63 : Tel un électrocardiogramme, CatchIt sonde l'application, de façon externe, pour y détecter toute anomalie potentielle. La seule tâche incombant au concepteur est la connexion déclarative entre les modèles applicatif et normatif. Cette tâche s'apparente, en termes médicaux, à une apposition de ventouses. Il suffit alors de déclencher l'appareil, les mesures et diagnostic sont automatiques.

Par ces observations discrètes, CatchIt établit un bilan de santé de l'application. Il soumet ces informations au concepteur et achève là sa mission. Seul ce dernier est apte à modifier le code applicatif. CatchIt s'apparente, sous cet angle, à un laboratoire d'analyses qui, s'il révèle d'éventuelles carences, peut, tout au plus, suggérer un éventuel traitement, mais, en aucun cas, le prescrire ou l'administrer : l'équipe médicale reste exclusivement constituée des concepteurs et développeurs impliqués dans l'affaire. Un suivi du patient s'impose ensuite pour juger de l'efficacité du traitement.

Sur cette métaphore empruntée au registre médical, nous venons d'illustrer la philosophie de l'outil et d'en cerner les attributions. Détaillons en maintenant le poumon, à savoir le modèle normatif, ainsi que les mécanismes d'exploitation pour une contribution proactive et réactive de l'outil.

2. Proactivité

CatchIt fonde sa proactivité sur la mise à disposition de modèles normatifs, représentatifs des domaines traités. Consultables pour une meilleure appréhension de ces domaines, ils sont aussi disponibles à la réutilisation. Langage de spécification et mécanismes d'exploitation sont donc nécessaires : nous introduisons, à ces fins, deux espaces de réflexion, CoTaS et DEGRE, que nous présentons maintenant. Nous proposons alors un ensemble de critères et métriques permettant de mesurer le taux de réutilisabilité : l'espace CriMePro y est consacré.

2.1. Espace CoTaS

L'espace CoTaS (COncepts -TAsks - Strategies) identifie trois pôles de connaissances : les concepts, tâches et stratégies. Tandis que les concepts dénotent les objets métier, les tâches et stratégies se réfèrent au comportement opérateur. Elles se distinguent par une nuance de variabilité : les tâches deviennent procédurales, déportant vers les stratégies le savoir-faire opérateur contextuel. Les stratégies encapsulent, en conséquence, les degrés d'autonomie accordés à l'opérateur.

Ainsi, l'espace CoTaS apparaît-il comme un affinement de la plus classique dualité tâche/objet. Nous le décrivons ici suivant ses trois pôles structurants.

2.1.1. Concepts

Les concepts dénotent les objets sémantiques du domaine. Ce sont les entités typiquement consignées dans les modèles objet de formalismes tels OMT. Mais, à la différence de [Normand 92], ces concepts ne se limitent pas aux « entités fonctionnelles du système susceptibles d'intervenir directement dans la réalisation des tâches utilisateur ». Ils concernent tout objet métier, indépendamment de l'exploitation qui en est faite dans les diverses applications. Hormis cette nuance de couverture de domaine, nous adhérons à la définition de [Normand 92] : un concept s'apparente à « un objet comportant un ensemble d'attributs de données et sur lequel peut être appliqué un ensemble d'opérations » (p 98).

Si cette dernière définition est fortement empreinte de considérations de l'ordre de l'implémentation, elle reflète parfaitement les deux perspectives selon lesquelles les concepts peuvent être envisagés : à savoir les données et les traitements. Tandis que les données tendent à dimensionner les concepts, les traitements explicitent les opérations offertes par ces entités. Nous illustrons ces deux aspects sur les concepts de véhicule et

de piste, respectivement extraits des domaines de la conduite automobile et de la Guerre Electronique (Figure 64).

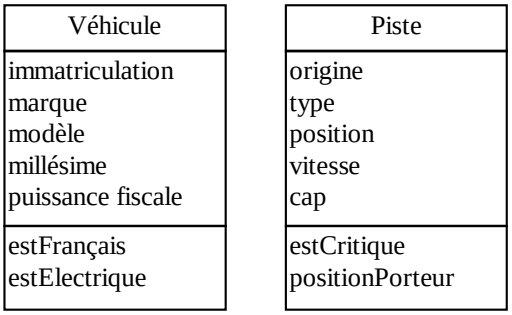

Figure 64 : Illustration des concepts de CoTaS sur les notions de véhicules et de pistes. Un véhicule est, en partie, défini par une marque ; il répond à la question testant la nationalité française de son constructeur. Une piste est localisée, dans l'absolu, par une position géographique. Elle offre un calcul de position relative vis-à-vis du porteur.

Ainsi, les concepts représentent des unités de compétence. Ils sont naturellement modélisés par des classes, au sens orienté-objet. Ils sont composés d'attributs et de méthodes. Tandis que les premiers en véhiculent les traits sémantiques, les seconds explicitent la compétence fonctionnelle de l'objet :

- un attribut est un media fédérateur permettant d'exprimer les traits sémantiques d'un objet et d'en véhiculer les valeurs. S'il banalise ces caractéristiques en en gommant notamment la sémantique, c'est pour gagner en généricité. Il peut ainsi exprimer bon nombre de configurations et notamment des liens entre concepts : qu'ils soient d'ordre structurel, sémantique ou logique [Normand 92], ils s'intègrent tous dans cette notion fédératrice d'attribut, au prix d'une sémantique limitée. Il est alors intéressant de la rehausser via un commentaire textuel. Aussi, l'attribut est-il ici défini par :
  - un nom ;
  - un type ;
  - et un commentaire ;
- les méthodes explicitent les compétences ou capacités fonctionnelles d'un concept. Elles sont définies par :
  - un nom ;
  - des paramètres d'entrée ;
  - un corps ;
  - des paramètres de sortie ;
  - et un commentaire que nous rajoutons pour rehausser la méthode de sa sémantique.

La figure 65 illustre, selon les notations OMT, la modélisation ainsi obtenue.

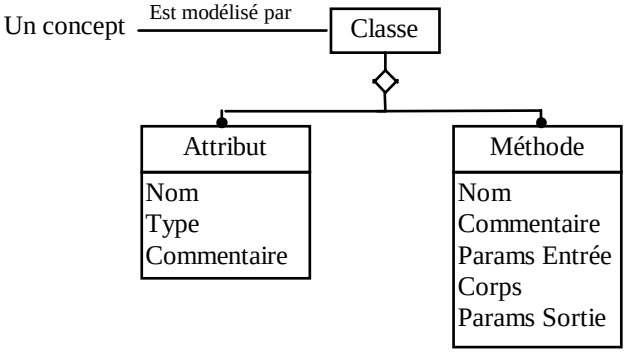

Figure 65 : Modélisation OMT des concepts de CoTaS.

Les concepts étant ainsi définis et modélisés, consacrons-nous aux tâches de l'espace CoTaS.

2.1.2. Tâches

L'objectif ici est de consigner un modèle de tâche représentatif du savoir-faire opérateur. C'est donc plus exactement d'un modèle de l'activité dont il s'agit : ce modèle se réfère aux tâches effectives de [Barthet 88] et stipule, pour l'avenir, les tâches prévues.

Si cette notion générique de « tâche prévue » permet d'embrasser l'activité humaine dans sa globalité, elle en occulte, par contre, les niveaux. Or, un consensus émerge aujourd'hui pour admettre l'existence d'une structuration en niveaux d'activité. Le modèle de [Rasmussen 86] en est un représentant. Mentionnons aussi [Allen 71] (cité dans [Neboit 94]) qui traite de la conduite automobile et propose une hiérarchie de trois sous-classes d'activités :

- « au niveau le plus élémentaire, la classe des activités de contrôle (contrôle de trajectoire et contrôle de vitesse) qui se caractérisent par un certain degré d'automaticité et de rapidité dans leur exécution ;
- au niveau intermédiaire, des activités de guidage mises en jeu dans des situations (conduite en file, dépassement, croisement, franchissement d'intersection...) ;
- et, au niveau supérieur, la navigation, c'est-à-dire la préparation de l'itinéraire à partir de cartes et la recherche de la direction à prendre en situation » [Allen 71].

Si cette étude rappelle la connotation procédurale, consciente ou inconsciente, des deux premiers niveaux, elle souligne aussi l'imprégnation contextuelle du dernier : selon les aléas de la circulation (embouteillages, accidents, travaux ou autres), la navigation adaptera son itinéraire pour une prestation optimisée. C'est cette nuance, certes subtile mais fondamentale, que nous souhaitons ici capturer via la distinction entre tâches et stratégies. Les tâches se limitent aux tâches procédurales, invariantes, c'est-à-dire insensibles au contexte.

Pour leur modélisation, nous nous référons aux travaux de [Balbo 94] :

- une tâche désigne un but assorti d'une procédure ;
- une procédure s'apparente à un ensemble organisé d'opérations, via des relations temporelles ou structurelles ;
- les relations temporelles traduisent séquentialité, interruptibilité ou parallélisme ;
- les relations structurelles expriment des compositions logiques ou alternatives.

Mais, à la différence de [Balbo 94], les opérations se restreignent ici aux tâches : nous excluons délibérément les actions qui, par leur dépendance envers les dispositifs d'entrée et de sortie, nuisent à la réutilisabilité des descriptions. Aussi, l'arbre des tâches reste-t-il homogène, exclusivement composé de tâches. A ses feuilles, s'accumulent les tâches élémentaires de [Balbo 94], c'est-à-dire des tâches dont les procédures s'exprimeraient exclusivement en termes d'actions physiques : ces procédures sont ici tues. En conséquence :

- une tâche est une tâche composée ou élémentaire ;
- toutes sont définies par un objectif ;
- mais seules les tâches composées sont assorties d'une procédure.

Nous illustrons cette définition des tâches sur les figures 66 et 67.

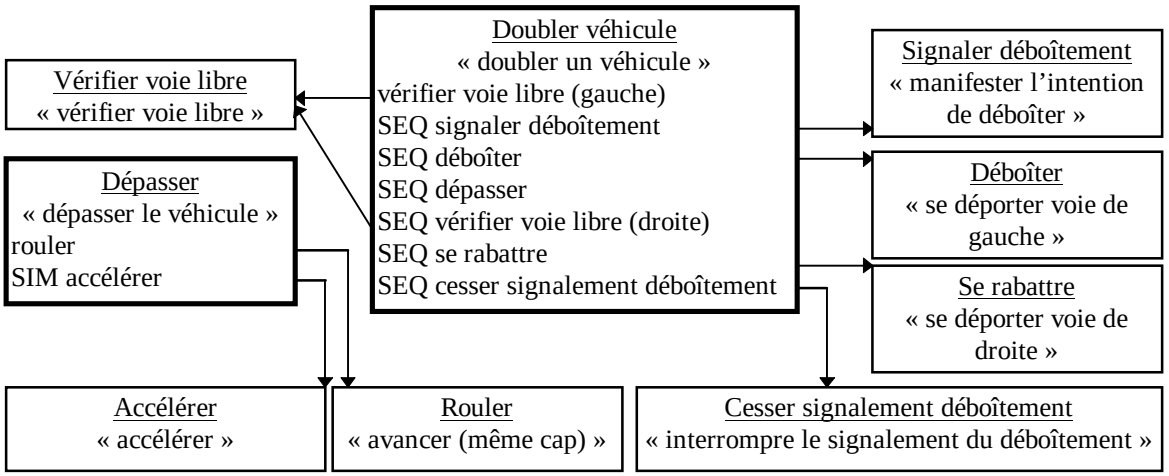

Figure 66 : Illustration des tâches dans le domaine de la conduite automobile : les tâches composées apparaissent en gras, à la différence des tâches élémentaires. Ces dernières sont dépourvues de procédures pour éviter toute référence aux rétroviseur, volant, clignotant ou accélérateur.

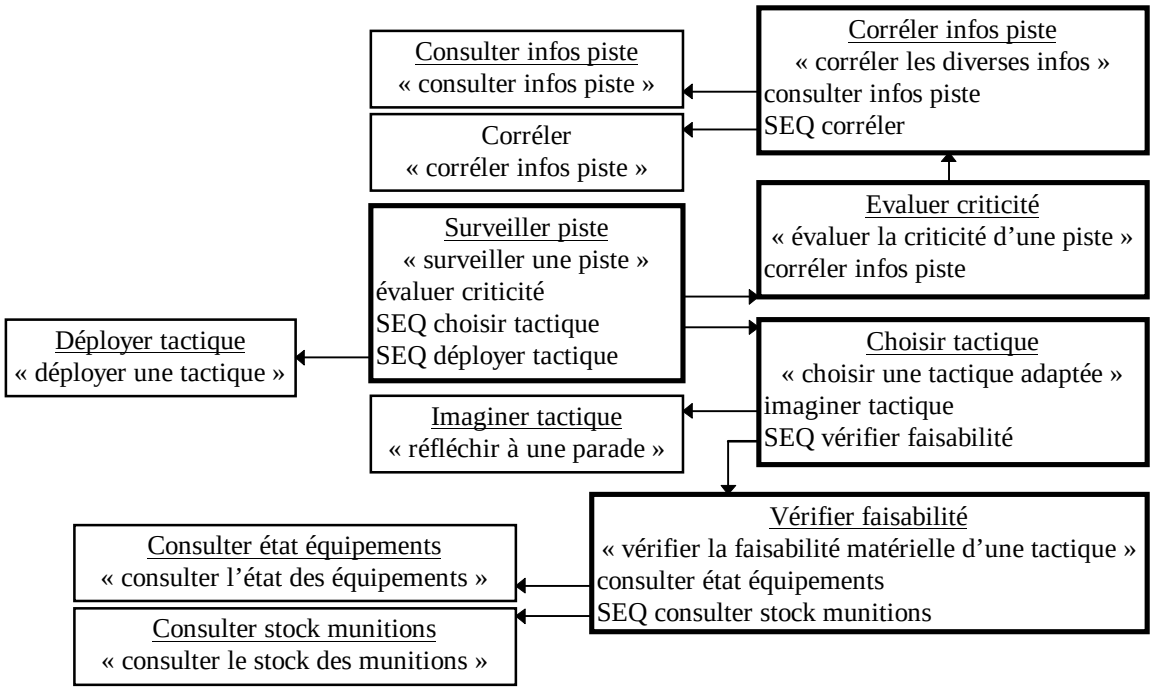

Figure 67 : Illustration des tâches dans le domaine (largement simplifié) de la guerre électronique : les tâches composées apparaissent en gras, à la différence des tâches élémentaires. Ces dernières sont dépourvues de procédures pour éviter toute référence aux dispositifs d'entrée et de sortie.

Les tâches étant ainsi définies, nous nous employons maintenant à leur modélisation. De nombreux formalismes existent en la matière. En référence à l'étude comparative de [Jambon 96], nous nous orientons vers MAD [Sebillotte 94]. Initialement conçu par des ergonomes pour l'analyse des tâches, MAD permet l'expression de propriétés structurelles et temporelles entre tâches. Représentées par un objet générique, appelé objet-tâche, les tâches y sont décrites selon trois volets :

- l'identification repose sur l'attribution, à chaque tâche effective, d'un numéro et d'un nom ;

- les éléments de la tâche regroupent la description de ses but, états initial et final, pré et post conditions et corps ;
- les attributs décorent la tâche de caractéristiques telles que l'interruptibilité, la priorité, les caractères facultatif, itératif, etc.

Nous enrichissons cette description d'un quatrième volet dédié à la spécification de l'acteur en charge de la tâche. Celle-ci pouvant être déléguée au système, nous généralisons ce terme d'acteur en « effecteur ». Une tâche est ainsi décrite selon quatre perspectives :

- l'identification repose sur un nom et un numéro interne :
  - le *nom* résume, au mieux, l'objectif de la tâche. Par exemple « Doubler véhicule » ou « Surveiller piste » ;
  - le *numéro* permet d'en distinguer les occurrences ;
- les éléments fixent le contenu :
  - le *but* revêt un caractère explicatif. Dépourvu de toute formalité, il exprime, en langage naturel, l'objectif de la tâche. Par exemple « doubler un véhicule » ou « évaluer la criticité d'une piste » ;
  - l'*état initial* cerne les objets du monde accédés par la tâche. En termes informatiques, il correspond aux paramètres d'entrée de la tâche. Pour notre exemple, l'état initial contient respectivement le véhicule à doubler et la piste à surveiller. Afin d'intégrer l'importance relative des différents paramètres, nous leur associons ici un poids. De type énuméré, ce poids reproduit les notions de visions centrale, périphérique et champ de vision limité. Ses valeurs sont : central, périphérique et masqué ;
  - les *préconditions* portent sur cet état initial. Elles expriment des contraintes sur les parties du monde ainsi circonscrites. Leur satisfaction est nécessaire au déclenchement de la tâche. On distingue trois types de préconditions :
    - les préconditions dites « état du monde » qui décrivent un état du monde nécessaire à l'exécution de la tâche ;
    - les préconditions dites « déclenchantes » qui confèrent un caractère préemptif à la tâche considérée. Testées en permanence, ces préconditions déclenchent l'exécution de la tâche, dès lors qu'elles sont satisfaites et, bien entendu, dans la mesure où la tâche courante est interruptible ;
    - les préconditions dites « opérateur » qui portent ici sur la disponibilité de l'effecteur de la tâche ;
  - le *corps* de la tâche en explicite la procédure, c'est-à-dire le composant exécutif. Il s'apparente à un ensemble d'opérations organisé par des relations temporelles ou structurelles. Rappelons que, pour assurer l'indépendance de la modélisation vis-à-vis des dispositifs physiques, et gagner ainsi en généricité, seul le corps des tâches composées est ici spécifié. Parmi les constructeurs disponibles [Sebillote 94], nous retenons :
    - la séquentialité (SEQ) pour exprimer un enchaînement temporel entre sous-tâches ;
    - le parallélisme (PAR) pour refléter l'absence de contraintes sur l'ordre d'exécution des tâches ;
    - la simultanéité (SIM) pour spécifier cette fois un parallélisme vrai entre sous-tâches, ce qui suppose des effecteurs différents ;
    - l'alternative (ALT) pour un choix exclusif entre sous-tâches ;
  - l'*état final* énumère les objets du monde modifiés ou créés par la tâche. En termes informatiques, il correspond aux paramètres de sortie de la tâche. Nous les décorons, de même, d'un poids pour refléter leur importance dans l'évaluation du nouvel état ;
  - les *postconditions* portent sur l'état final. Elles expriment des conditions sur les objets de l'état final, ces contraintes devant être satisfaites en fin de tâche. De même que pour les préconditions, on en distingue trois types :
    - les postconditions dites « résultat » identifient l'effet d'événements internes survenus au cours de l'exécution de la tâche ;
    - les postconditions dites « état du monde » répertorient toute modification de l'état du monde, étrangère à l'activité de l'opérateur ;

. les postconditions « fin de tâche » formulent des conditions suffisantes à l'interruption de la tâche. Testées en permanence, elles suspendent définitivement la tâche, dès lors qu'elles sont satisfaites.

- les attributs décorent les tâches de propriétés booléennes. Nous retenons les caractères optionnel (FAC), itératif (@), prioritaire (PRIOR) et interruptible (INTER) ;
- enfin, l'effecteur est relatif à l'affectation des tâches. Il relève de la coordination. Il spécifie les acteurs prévus et potentiels :
  - un acteur prévu est un effecteur (système ou opérateur) privilégié pour la réalisation de la tâche ;
  - un acteur potentiel est un effecteur apte à sa mise en œuvre. Il est candidat à la délégation en cas de migration.

La figure 68 résume la modélisation ainsi obtenue.

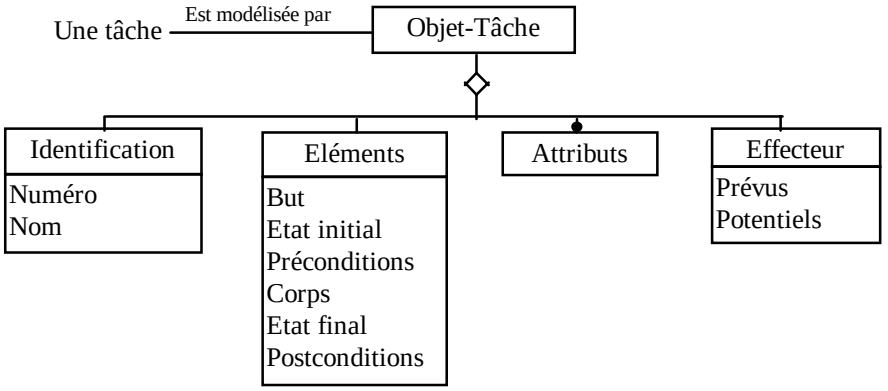

Figure 68 : Modélisation OMT des tâches de CoTaS.

Les tâches étant ainsi définies et modélisées, nous nous consacrons, dans la section suivante, aux stratégies de CoTaS.

2.1.3. Stratégies

Si les tâches explicitent des procédés acquis, a priori rationnels et indubitables, les stratégies expriment, à l'inverse, un savoir-faire opérateur très contextuel. Ces connaissances sont des données empiriques, typiquement établies par l'observation ou l'expérience : elles ne reposent sur aucun fondement théorique, mais une capitalisation empirique. Elles relèvent de deux ordres : d'une part, de l'occurence contextuelle de la tâche ; d'autre part, de son degré de nécessité.

D'un point de vue modélisation, les stratégies s'apparentent à des règles valuées du type « Si *condition* alors *action* » où :

- la condition porte sur le contexte d'interaction ;
- l'action désigne une tâche ou composition de tâches de CoTaS. Elle s'apparente à un corps de tâche ;
- la valuation spécifie le caractère autorisé, conseillé, déconseillé, impératif ou interdit de la tâche.

En voici des exemples concrets :

- en conduite automobile :
  - Si verglas alors doubler véhicule déconseillé
  - Si véhicule précédent lent alors doubler véhicule conseillé
  - Si mauvaise visibilité alors doubler véhicule interdit
- en guerre électronique :
  - Si une nouvelle piste apparaît alors surveiller cette piste impératif
  - Si un équipement est en panne alors passage en mode opérationnel déconseillé
  - Si un équipement est en panne alors identifier panne impératif
  - Si équipements opérationnels alors passage en mode opérationnel autorisé

- Si un équipement est en panne alors passage en mode maintenance conseillé
- Si une piste est hostile alors passage en mode maintenance interdit
- Si danger imminent alors déployer armes dures autorisé
- Si stock de munitions limité alors déployer armes dures déconseillé.

Si cette modélisation confirme la possibilité de banaliser les stratégies en tâches, par le biais des préconditions et attributs, les exemples soulignent le caractère empirique des stratégies : elles relèvent de l'expertise métier et incarnent le savoir-faire opérateur. Tandis que les tâches fixent un grain d'analyse à connotation procédurale, les stratégies en proposent un habillage empirique, opérationnel.

CoTaS étant ainsi défini, nous nous consacrons maintenant à l'espace DEGRE : ils agissent en complémentarité pour une meilleure proactivité.

2.2. Espace DEGRE

L'espace DEGRE (Domain Encapsulation by Genericity Refinement and Evolutivity) relève de la capitalisation : il s'attache à la gestion des modèles pour une réutilisation maîtrisée. Il s'articule autour de trois dimensions : les généricité, affinement et évolutivité que nous étudions ci-dessous.

2.2.1. Généricité

Si le grain d'analyse est resté jusqu'ici binaire, opposant applications et domaines, il oscille, dans la réalité, entre un continuum de valeurs identifiant des sous-domaines d'activité. La guerre électronique marine, par exemple, constitue un sous-domaine de la guerre électronique : elle filtre, voire spécialise, des informations plus génériques issues de ce domaine père. Ainsi, les domaines s'insèrent-ils au sein d'une hiérarchie (Figure 69) : leur niveau de profondeur reflète leur généricité (ou spécificité).

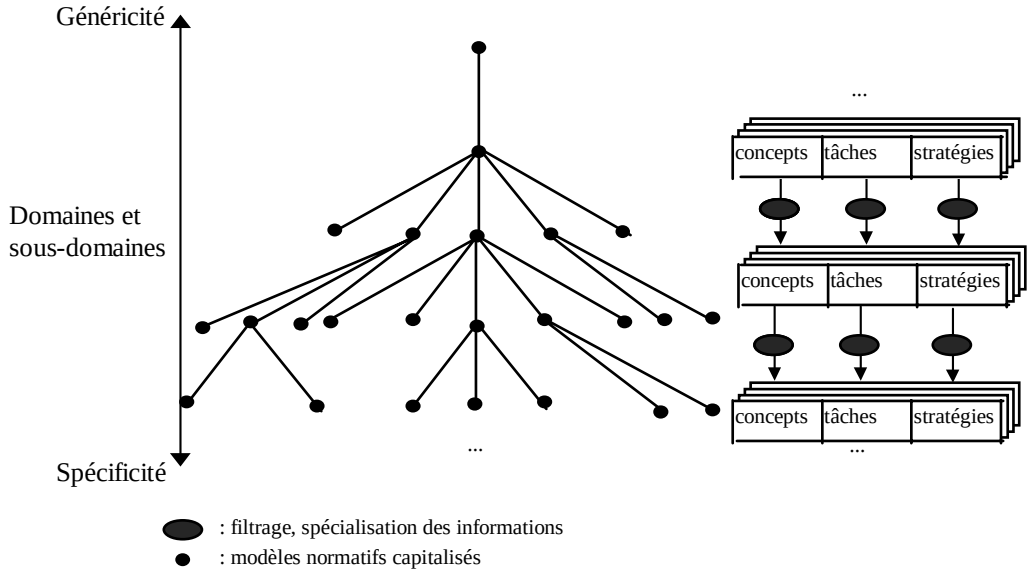

Figure 69 : Organisation hiérarchique des domaines et sous-domaines capitalisés. Leur profondeur dans l'arbre reflète leur généricité/spécificité.

Si un domaine est, en pratique, capitalisé via ses concepts, tâches et stratégies, la question de l'ancrage de cette capitalisation dans le processus de développement se pose. Nous en traitons dans la section suivante, consacrée à la deuxième dimension de DEGRE : le niveau d'affinement des connaissances au regard du processus de développement.

2.2.2. Affinement

L'affinement traite du niveau de description des entités de CoTaS au regard du processus de développement : concepts, tâches et stratégies sont-ils, en clair, capitalisés niveau spécification, conception ou implémentation ?

Pour une meilleure continuité du processus de développement, nous préconisons ici la mise en œuvre d'une structure d'accueil générique :

- couvrant les trois niveaux de description - spécification, conception, implémentation - pour une progression continue ;
- permettant la navigation entre ces niveaux, via peut-être un système de lentilles magiques [Bier 93] ;
- et gérant les reconceptions inhérentes aux approches itératives.

Si CatchIt cible, à terme, l'intégration de cette structure d'accueil, CoTaS est aujourd'hui envisagé au niveau implémentation : concepts, tâches et stratégies sont décrits en langage informatique orienté objet.

La complexité des domaines traités compromettant une modélisation immédiate, complète et correcte, l'évolutivité des modèles est un mal nécessaire. Aussi, complétons-nous la structuration précédente par un critère orthogonal, dédié à la gestion de configurations : l'évolutivité.

2.2.3. Evolutivité

CatchIt traite l'évolutivité des modèles par la notion fédératrice de version : tout modèle capitalisé, c'est-à-dire inscrit dans l'arborescence des domaines, se voit désormais estampillé d'un numéro de version. Par exemple, deux modèles V0.1 et V0.2 peuvent normaliser la guerre électronique marine.

Les espaces CoTaS et DEGRE, supports à la capitalisation, étant ainsi spécifiés, nous proposons, dans la section suivante, des critères et métriques pour en mesurer la réutilisation.

2.3. Espace CriMePro

L'espace CriMePro (Critères et Métriques en Proactivité) propose des critères et métriques pour mesurer la contribution proactive de CatchIt. Il se limite aujourd'hui à la réutilisation logicielle. Il mesure :

- la profondeur, dans l'arborescence, du domaine réutilisé ;
- et le taux de réutilisation au sein de ce domaine.

Ainsi s'achève la description de la prestation proactive de CatchIt. Avant d'envisager la réactivité de l'outil, nous en proposons ci-dessous une synthèse.

2.4. Synthèse et discussion

En résumé, CatchIt fonde sa proactivité sur une capitalisation structurée de connaissances métier. Cette structuration se manifeste à deux titres : par l'identification, d'une part, et la gestion, d'autre part, de pôles de compétences respectivement formalisés et administrés dans les espaces CoTaS et DEGRE.

CoTaS préconise une structuration tripartite des connaissances métier. Il favorise une définition déclarative des interfaces. Il milite pour une base de connaissances décorrélée des plans de résolution [Détienne 95a]. Il propose, pour ce faire, trois plans d'analyse (Figure 70) :

- les concepts incarnent les objets métier ;
- les tâches représentent le savoir-faire procédural des opérateurs : elles s'appuient sur les concepts ;
- les stratégies capturent enfin le savoir-faire contextuel des opérateurs : elles éclairent tâches et concepts.

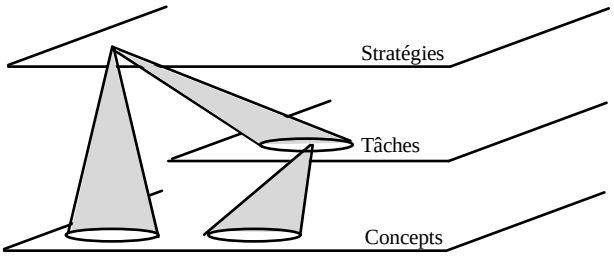

Figure 70 : CoTaS pour une structuration tripartite des connaissances métier : les tâches éclairent les concepts ; les stratégies éclairent tâches et concepts.

Il est alors intéressant, dans un effort d'abstraction, d'étudier l'éventuelle filiation de CoTaS par rapport à GOMS [Card 83]. GOMS est un modèle de description du comportement humain. Il travaille à différents niveaux d'abstraction, depuis la tâche jusqu'aux actions physiques. Il introduit, à ces fins, quatre ensembles pour représenter l'activité cognitive d'un individu engagé dans la réalisation d'une tâche : les buts (Goals), Opérateurs, Méthodes et règles de Sélection. Les buts sont organisés de manière hiérarchique jusqu'à mentionner des buts dits élémentaires, réalisés par le biais d'opérateurs. Ces opérateurs composent les méthodes qui décrivent les procédures permettant d'atteindre les buts fixés. En cas de méthodes concurrentes, ce sont les règles de sélection qui gèrent le conflit : elles déterminent, dans un contexte donné, la méthode à privilégier compte tenu de cette situation courante. Quatre remarques à cela :

- GOMS est un modèle résolument centré tâche, qui occulte la modélisation des concepts de CoTaS ;
- GOMS se limite à la modélisation de l'activité opérateur et exclut les tâches système ;
- GOMS ne traite pas des degrés de priorité/nécessité entre tâches (conseillé, impératif, etc.) ;
- GOMS est un modèle prédictif, quantitatif de performance : il ignore l'erreur humaine, contrairement à CoTaS dont l'objectif est de fournir à CatchIt toute information permettant de les détecter.

Ainsi, la modélisation CoTaS constitue, sous certains aspects, une extension à GOMS : les concepts y sont modélisés de façon explicite, les tâches sont étendues aux attributions système et les stratégies y sont valuées. A l'inverse, la généricité du modèle y est restreinte, le niveau d'abstraction de l'analyse étant fixé. Dans GOMS, au contraire, la granularité de l'opérateur peut épouser quatre valeurs : les niveaux tâche, fonctionnel, argument et physique. CoTaS adopte résolument le niveau tâche : il structure l'espace de travail en une hiérarchie de sous-tâches dont la nature dépend uniquement du domaine.

Si CoTaS cible la description des connaissances métier, DEGRE traite de leur administration. Il identifie trois dimensions complémentaires :

- la généricité s'applique au domaine considéré : elle permet de le situer dans un arbre d'héritage selon ses spécificités ;
- l'affinement traite de la couverture du processus de développement en termes de spécification, conception et implémentation des entités de CoTaS ;
- l'évolutivité relève enfin de la gestion de configurations de ces modèles.

Si CatchIt 98 fixe l'affinement au niveau implémentation, il traite, en revanche des généricité et évolutivité. Nous en schématisons le principe par un organigramme stipulant la démarche préconisée pour construire une nouvelle application, un nouveau domaine ou une nouvelle version de ce même domaine (Figure 71). Les critères et métriques de CriMePro mesurent la réutilisation effective.

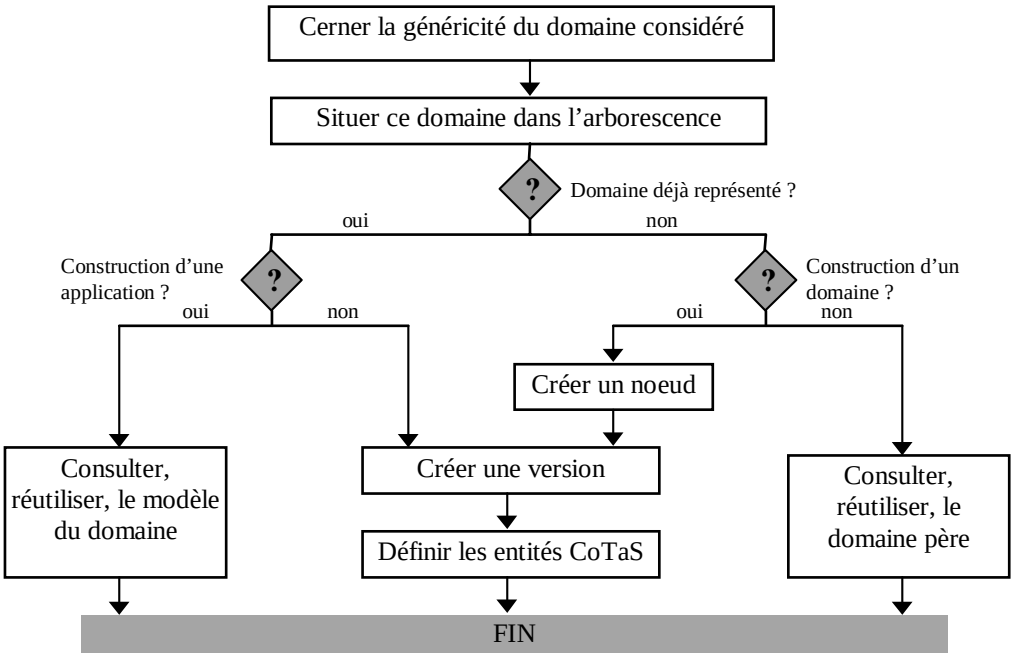

Figure 71 : Procédure préconisée dans une démarche constructive pour, soit créer une nouvelle application, soit enrichir la base normative d'un nouveau modèle ou d'une nouvelle version.

Ainsi s'achève la description de la proactivité de CatchIt. Nous nous consacrons maintenant à sa réactivité.

3. Réactivité

CatchIt fonde sa réactivité sur un étalonnage de l'application par rapport à une référence. Cette référence est le modèle normatif associé au domaine traité : elle s'exprime en termes d'entités CoTaS et s'inscrit dans l'arborescence DEGRE. Il s'agit ici de mesurer la distance entre les modèles applicatif et normatif. CatchIt propose, pour ce faire :

- une connexion déclarative entre ces deux protagonistes : l'espace PATCH y est consacré ;
- un référentiel de critères et métriques, consignées dans CriMeRéact ;
- ainsi que des mécanismes d'exploitation.

Nous les étudions dans cet ordre.

3.1. Espace PATCH

PATCH (Presentation and Abstraction for Task and Concept Hooks) relève de la connexion déclarative entre les modèles applicatif et normatif. Il s'agit concrètement de projeter, dans l'application, toute entité CoTaS du modèle normatif. PATCH propose, pour ce faire, une grille d'analyse : elle constitue une taxonomie de liens. Inspirée du modèle PAC [Coutaz 90], elle distingue, pour toute entité, abstraction et présentation :

- l'abstraction dénote les définition et compétences de l'objet ;
- la présentation définit son image tant en entrée qu'en sortie.

Dans un effort d'unification, et compte tenu du statut des stratégies, à savoir augmenter les tâches de préconditions opérationnelles, tâches et stratégies sont ici banalisées. Aussi, est-ce finalement quatre types de liens que suggère PATCH (Figure 72) :

- les PatCH et pAtCH associent respectivement présentation et abstraction aux concepts de l'application ;
- les PaTcH et pATcH effectuent ces mêmes projections pour les tâches et stratégies.

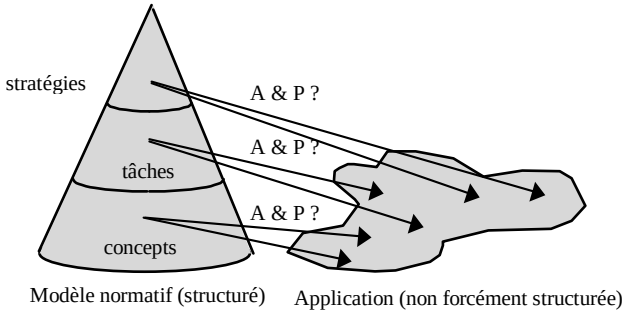

Figure 72 : Connecter les modèles normatif et applicatif signifie localiser dans ce dernier la trace de tout concept, tâche et stratégie métier. Cette trace se décline selon deux composantes : l'abstraction et la présentation de l'entité considérée.

Les liens de type abstraction ne sont aujourd'hui que partiellement traités :

- les pAtCH se limitent à la définition des concepts, occultant totalement la perspective fonctionnelle : seuls les attributs sont reliés ;
- les pATcH ne sont pas implémentés.

D'un point de vue modélisation, nous proposons un cadre fédérateur distinguant deux niveaux de description (Figure 73) :

- au niveau HL (High Level), les liens sont établis à la granularité des entités CoTaS ;
- au niveau LL (Low Level), ils portent, à l'inverse, sur les constituants de ces entités. Tandis que ces constituants s'expriment en termes d'attributs pour les concepts, ils relèvent, pour les tâches, des pré et post conditions, états initial et final.

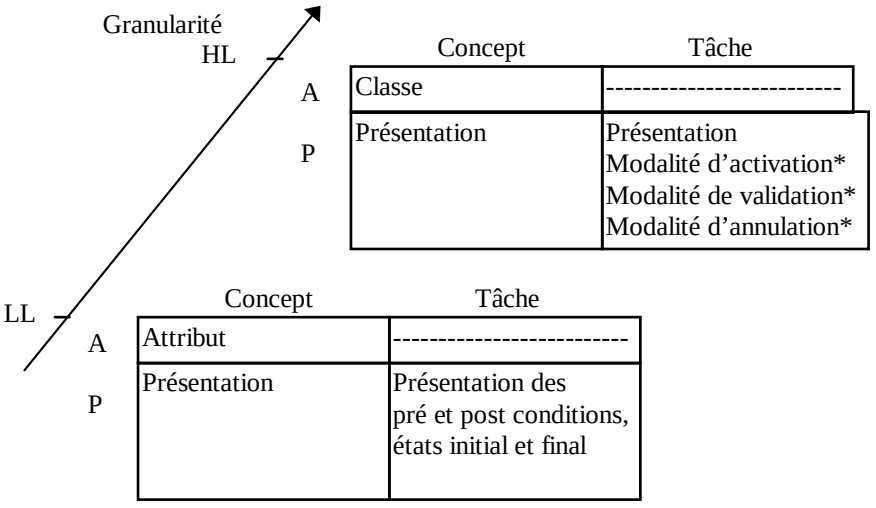

Figure 73 : Déclinaison de l'espace PATCH à deux niveaux de granularité : HL et LL.

Nous étudions ici ces liens au cas par cas.

Liens pAtCH

Les liens pAtCH supposent l'application orientée objet. Ils connectent les classes applicatives aux concepts normatifs. Tandis qu'au niveau HL, ces liens s'expriment entre classes, ils se formulent, au niveau LL, en termes d'attributs. Supposons, par exemple, dans le domaine GE, que :

- l'application dédie la classe *PorteurA* au porteur ;
- dans une quête de généricité, le modèle normatif le banalise en un objet tactique, modélisé par la classe *ObjetTactiqueN* ;
- les classes *PorteurA* et *ObjetTactiqueN* mettent en œuvre une vitesse.

La connexion spécifiera :

- au niveau HL, un lien entre les classes *PorteurA* et *ObjetTactiqueN* ;
- au niveau LL, un lien entre les attributs *vitesse* de *PorteurA* et *ObjetTactiqueN*.

Liens PaTcH

Les liens PaTcH associent, aux tâches normatives, une présentation dans l'application. Nous adoptons la terminologie de [Normand 92] (cf SIROCO au chapitre V) : une tâche est hébergée dans un espace de travail. Le lien PaTcH HL identifie cet espace et précise complémentairement les éventuelles modalités d'activation, validation ou annulation associées (Figure 73). Ces modalités se réfèrent respectivement aux actions physiques nécessaires pour :

- afficher l'espace de travail considéré ;
- entériner les manipulations opérateur menées dans cet espace ;
- ou, au contraire, les rendre caduques.

Ces modalités s'expriment en termes d'éléments de présentation ou de raccourcis clavier / souris. Ces éléments peuvent être prédéfinis, définis ou dynamiques. C'est respectivement le cas des widgets, des images et enfin des dessins construits à la volée. Tous ces éléments sont hébergés dans des espaces de travail.

Dans le cas de tâches élémentaires limitées à une seule action physique, nous affinons cette projection en identifiant l'élément de présentation concerné.

Les liens LL de cette nature associent des présentations aux pré et post conditions, états initial et final de ces tâches (Figure 73). Supposons, par exemple (Figure 74) :

- que le modèle normatif stipule une tâche « *Configurer brouilleur* » ;
- que, parmi l'état initial de la tâche, figure le concept *brouilleur* ;
- qu'une précondition de cette tâche porte sur l'état du brouilleur ;
- qu'un widget « *Etat brouilleur* », dans la fenêtre principale, reflète l'état OK ou NOK de ce brouilleur ;
- que, dans l'application, un bouton intitulé « *Configurer brouilleur* » offre l'accès à un espace de travail dédié à la configuration du brouilleur ;
- qu'au sein de cet espace « *Configuration brouilleur* », les boutons *Valider* et *Annuler* permettent de valider et annuler les actions opérateur.

Les liens HL connecteront la tâche « *Configurer brouilleur* » à l'espace « *Configuration brouilleur* », en enregistrant les boutons « *Configurer brouilleur* » *Valider* et *Annuler* comme modalités d'activation, de validation et d'annulation (Figure 74). Les liens LL associeront à la précondition le widget « *Etat brouilleur* » (Figure 74). Ils sélectionneront, de plus, parmi les présentations associées au concept *brouilleur*, celles pertinentes pour l'état initial.

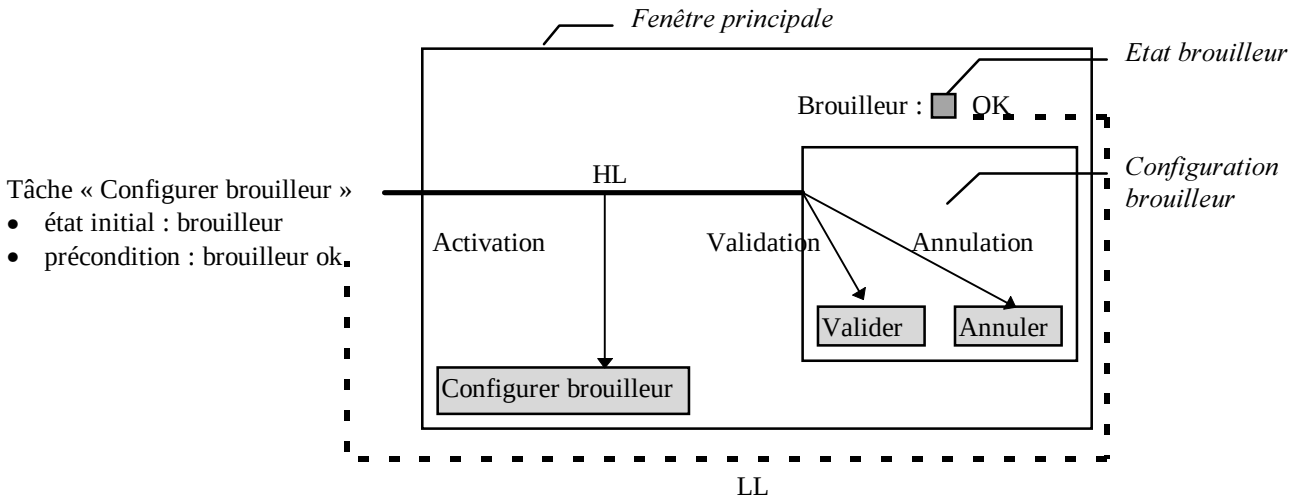

Figure 74 : Illustration des liens PaTcH HL et LL respectivement représentés en traits pleins et pointillés.

Si maintenant le modèle métier spécifie le corps de *Configurer brouilleur* en sous-tâches *configurer fréquence*, etc., alors les liens PaTcH permettront d'associer des présentations à ces sous-tâches. Ces présentations seront, par expérience, des champs texte agencés dans l'espace « *Configuration brouilleur* ». Le modèle normatif fixe donc bien le niveau d'abstraction des informations à contrôler.

Liens PatCH

Les liens PatCH HL associent des présentations aux concepts normatifs. Ces présentations sont des éléments de présentation ou des espaces de travail. La notion de modalité ne fait, en revanche, pas sens ici. Les liens LL affinent cette description en identifiant, au sein de ces présentations, les aspects relevant des différents attributs. Supposons, par exemple, qu'un espace de travail « *Etat détaillé brouilleur* » visualise un état détaillé du brouilleur et que, parmi ces présentations, figure un widget « émetteur » dédié à l'état de l'émetteur. Alors, parmi les liens HL, figurera l'association *brouilleur- Etat détaillé brouilleur* ; parmi les liens LL, l'association « *émetteur- Etat détaillé brouilleur, émetteur* ».

Le principe de cette double granularité étant décrit, nous le généralisons comme suit :

- un lien PATCH HL est modélisé par :
  - deux entités, l'une provenant du modèle normatif, l'autre de l'application ;
  - une éventuelle condition de mise en correspondance des deux entités ainsi identifiées ;
  - et un ensemble de liens pAtCH LL établis entre les constituants de ces entités ;

- un lien PATCH LL est modélisé par :
  - deux composants, l'un relatif à l'entité du modèle normatif, l'autre à celle de l'application ;
  - une éventuelle condition de mise en correspondance des deux composants ainsi définis ;
  - et une règle, dite de transformation, exprimant la relation concrète unissant ces deux protagonistes.

Pour illustrer l'intérêt des conditions de mise en correspondance et règles de transformation, supposons que :

- dans l'application, les pistes soient banalisées en la classe *ObjetTactiqueA*, alors que le modèle normatif offre une classe dédiée *PisteN*. La condition de mise en correspondance exprime le fait que la connexion entre les classes *ObjetTactiqueA* et *PisteN* n'a lieu d'être que pour des objets de type *piste* ;
- les attributs *vitesse* des classes *PorteurA* et *ObjetTactiqueN* déjà mentionnées soient d'unités discordantes : respectivement m/s et Noeuds. La règle de transformation exprime ce changement d'unité. Deux alternatives sont alors envisageables selon le point d'ancrage adopté : soit exprimer le modèle normatif en fonction de l'application, soit, au contraire, exprimer l'application en fonction de cette référence. Seule l'exploitation attendue de ces liens peut ici permettre de se positionner. Aussi, rappelons-nous leur ontologie : s'ils enrichissent l'application de sa sémantique opérationnelle, leur ultime objectif consiste à entretenir, côté modèle normatif, un contexte d'interaction représentatif des situations courante et passées.

A ces fins, tout événement applicatif est propagé vers le modèle normatif : le contexte y est alors mis à jour, calculé à partir des stimuli ainsi propagés. Il s'agit donc, autrement dit, de recalculer la vitesse normative à partir de la nouvelle vitesse applicative, ceci pour garantir la représentativité de l'état de l'application, côté modèle normatif. La première alternative s'impose donc : les règles de correspondance exprimeront le normatif en fonction de l'application. Sur notre exemple, c'est concrètement la formulation suivante qui est retenue : *vitesse normative = vitesse applicative** 3600/1852.

Avant de clore cette description, évoquons la gestion des liens PATCH ainsi établis. Ils sont maintenus, de façon externe, aux modèles applicatif et normatif, au sein d'un composant dédié : l'adaptateur (Figure 80). L'intérêt d'isoler les liens au sein d'un tel composant s'exprime en termes d'autonomie et de pureté pour les modèles connectés. Ce composant fera l'objet d'une description dans le chapitre suivant dédié à la mise en œuvre de l'outil. Nous nous consacrons, au préalable, à l'exploitation de ces liens.

3.2. Espace CriMeRéact

CriMeRéact (Critères et Métriques en Réactivité) propose un référentiel de critères et métriques pour une évaluation objective et quantitative de l'interaction obtenue. Aujourd'hui deux types de critères sont envisagés selon qu'ils traitent du domaine ou des propriétés d'utilisabilité.

3.2.1. Domaine

Il s'agit ici d'évaluer la qualité de l'application en regard du modèle normatif. Cette qualité s'exprime en termes de couverture et correction de la modélisation applicative vis-à-vis de cette référence.

Le débat porte ici sur les liens de type abstraction, c'est-à-dire les liens pAtCH, les pATcH n'étant aujourd'hui pas traités. Il s'agit d'évaluer, en regard du modèle normatif, d'une part, le taux de couverture atteint côté application, d'autre part, la complexité du tissage concrètement obtenu. Nous introduisons, pour cela, deux niveaux d'analyse : tandis que le niveau macroscopique considère les liens HL, le niveau microscopique s'intéresse à une entité donnée et en examine les connexions de type LL. Afin de banaliser ces deux échelles de traitement, nous adoptons une perspective ensembliste :

- nous considérons l'application et le modèle normatif comme respectivement des ensembles de départ et d'arrivée, notés E et F ;
- les liens définissent alors une relation R entre ces deux ensembles.

Il s'agit de quantifier la qualité de la connexion, sur la base de la relation R ainsi établie. Nous introduisons, à ces fins, la notion de graphe : un **graphe** isole, dans les ensembles E et F , les sous-ensembles d'éléments Ei et Fi interconnectés. Autrement dit :

g(Ei, Fi), noté gi, regroupe l'ensemble des éléments eij et fij de Ei et Fi tels que :

- $\forall$ eij $\in$ Ei, $\exists$fik $\in$ Fi / R(eij, fik)
- $\forall$ fij $\in$ Fi, $\exists$ eik $\in$ Ei / R(eik, fij)
- $\forall$ eij $\in$ E– Ei, $\neg\exists$ fik $\in$ Fi / R(eij, fik)
- $\forall$ fij $\in$ F – Fi, $\neg\exists$ eik $\in$ Ei / R(eik, fij).

A titre d'exemple, la relation R, graphiquement définie comme suit, révèle deux graphes g1, g2 tels que (Figure 75) :

- g1 implique 2 éléments de E et F et met en œuvre 3 liens ;
- g2 implique respectivement 2 et 3 éléments de E et F et met en œuvre 5 liens.

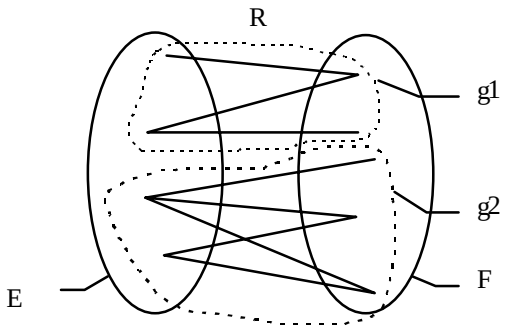

Figure 75 : Illustration de la notion de graphe : la relation R établie entre les deux ensembles E et F isole deux graphes g1 et g2 respectivement composés de 3 et 5 liens.

La notion de cardinalité apparait ainsi. Nous la définissons formellement comme suit :

- la **cardinalité** ou **cardinalité effective** d'un graphe gi consigne le nombre de liens concrètement tissés entre les sous-ensembles Ei et Fi interconnectés. Mathématiquement :

$$card(gi) = card(\{(eij, fik)\}), (eij, fik) \in Ei \times Fi / R(eij, fik))$$

Cette cardinalité évolue entre deux valeurs, dites cardinalités minimale et maximale, définies comme suit :

- la **cardinalité minimale** d'un graphe gi exprime le nombre de connexions strictement nécessaires pour en assurer l'existence. Mathématiquement :

$$cardMin(gi) = card(Ei) + card(Fi) - 1$$

- la **cardinalité maximale** d'un graphe gi se réfère, à l'inverse, au potentiel maximal de liens envisageables entre les sous-ensembles Ei et Fi . Mathématiquement :

$$cardMax(gi) = card(Ei) * card(Fi)$$

Nous complétons ces définitions par les notions de densité, poids et canonicité :

- la **densité** d'un graphe gi exprime la quantité de liens effective par rapport au nombre minimal requis. C'est-à-dire :

$$densité(gi) = card(gi) / cardMin(gi)$$

- le **poids** d'un graphe gi reflète la quantité d'entités impliquées et la densité des liens mis en œuvre :

$$poids(gi) = cardMax(gi)*densité(gi)$$

- la **canonicité** d'un graphe gi exprime, en relatif, le poids de ce graphe par rapport à un graphe basique n'impliquant que deux entités :

$$canonicité(gi) = 1/poids(gi)$$

Pour notre exemple :

	g1	**g2**
card(Ei)	2	2
card(Fi)	2	3
card(gi)	3	5
cardMin(gi)	3	4
cardMax(gi)	4	6

densité(gi)	1	5/4
poids(gi)	4	15/2
canonicité(gi)	1/4	2/15

Nous introduisons alors les notions de **couverture** et **correction** :

- la couverture sanctionne les éléments isolés. Elle se mesure par la formule :

$$\text{couverture}(R) = \text{couverture}(E) * \text{couverture}(F)$$

où couverture(E) = Σ_i card(Ei) / card(E)
et couverture(F) = Σ_i card(Fi) / card(F)

- la correction condamne les graphes fortement connectés. Elle se mesure par l'expression :

$$\text{correction}(R) = \Sigma_i \text{ canonicité (gi)} / \Sigma_i$$

Nous définissons alors la notion de **qualité**. Si elle mesure la bijectivité de la relation R, ce n'est nullement pour imposer une telle relation ou lui conférer un caractère optimal : il s'agit ici de mettre en lumière toute discordance de cette nature pour susciter, chez le concepteur, les « bonnes questions ».

$$\text{qualité}(R) = \text{correction}(R) * \text{couverture}(R)$$

Pour notre exemple :

	Valeurs
couverture(E)	1
couverture(F)	1
couverture(R)	1
correction(R)	23/120
qualité(R)	23/120 = 0.19

Mais sans référence, ces valeurs de qualité restent inexploitables. Aussi bornons-nous leur domaine de variation : elles évoluent entre 0 et 1. En effet :

. card(gi) ≥ cardMin(gi) donc densité(gi) ≥ 1
. or card(Ei)≥ 1 et card(Fi)≥ 1 donc poids(gi) ≥ 1
. d'où canonicité(gi) ≤ 1
. comme couverture(Ei) ≤ 1 et couverture(Fi) ≤ 1
. 0 ≤ qualité(R) ≤ 1

La valeur maximale est atteinte pour une relation bijective. Elle suppose en effet :

. ∀i canonicité(gi) = 1 c'est-à-dire card(Ei) = card(Fi) = 1 et card(gi)= cardMin(gi) = 1, c'est-à-dire des associations 1-1 ;
. et couverture(E) = couverture(F) = 1 c'est-à-dire l'absence d'éléments isolés de part et d'autre.

Ceci correspond bien à une relation bijective entre les ensembles E et F.

Dans ce contexte ensembliste, la notion de qualité est ainsi définie et assortie d'une métrique. Appliquons-la maintenant à la connexion entre modèles applicatif et normatif, pour en mesurer les couverture et correction. Nous proposons, à ces fins :

- d'appliquer cette formule aux niveaux macroscopique et microscopique de la connexion ;
- et de pondérer la valeur mesurée au niveau macroscopique, par celle obtenue, plus finement, au niveau microscopique.

La pondération s'effectue au niveau de la canonicité :

$$canonicité_{HL}(gi) = canonicité\ (gi)* qualité_{LL}(gi)$$

où $qualité_{LL}$(gi) représente la qualité, au niveau microscopique, du graphe gi. Elle se calcule en considérant les relations définies entre les attributs des classes deux à deux connectées : c'est la moyenne de leurs qualités respectives.

Avec ces notations, la qualité de la connexion se mesure par la formule :

$$qualité = qualité_{HL}(R) = couverture_{HL}(R) * correction_{HL}(R)$$

où $correction_{HL}(R) = \Sigma_i\ canonicité_{HL}(gi) / \Sigma_i$

Nous illustrons ces critère et métrique sur un exemple concret, issu du domaine de la guerre électronique. Nous considérons, à ces fins, les classes *PorteurA*, *PisteA*, *ObjetTactiqueN*, *CinematiqueA* et *CinematiqueN* définies et connectées comme suit (Figure 76) :

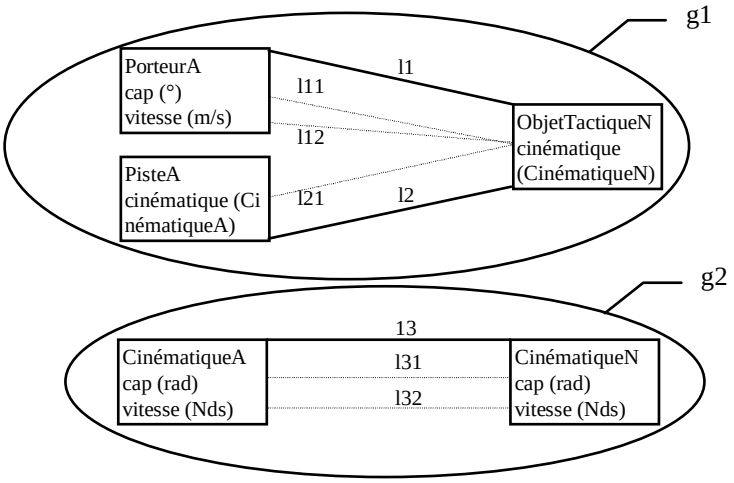

Figure 76 : Exemple support pour le calcul de la qualité de la connexion. Dans nos notations, les liens lij dénotent des liens microscopiques, affinant les liens macroscopiques li.

Cette configuration révèle :

- au niveau macroscopique : deux graphes, g1, g2, respectivement composés des classes : PorteurA, PisteA et ObjetTactiqueN, d'une part ; CinématiqueA et CinématiqueN, d'autre part ;
- au niveau microscopique : quatre graphes g11, g12, g21, g22 respectivement composés des liens {l11, l12}, {l21}, {l31} et {l32}.

	Niveau macroscopique E = *{PorteurA, PisteA, CinématiqueA}* F = *{ObetTactiqueN, CinématiqueN}* liens = {l1, l2 , l3}	
couverture(E)	1	
couverture(F)	1	
	g1 Ei = *{PorteurA, PisteA}* Fi = *{ObetTactiqueN}* liens = {l1, l2}	g2 Ei = *{CinématiqueA}* Fi = *{ CinématiqueN }* liens = {l3}
card(Ei)	2	1
card(Fi)	1	1
card(gi)	2	1
cardMin(gi)	2	1

cardMax(gi)	2		1	
densité(gi)	1		1	
poids(gi)	2		1	
canonicité(gi**)**	1/2		1	
	Niveau microscopique			
	g1		g2	
	E = *{cap, vitesse}* F = *{cinématique}* liens = {l11, l12}	E = *{cinématique}* F = *{cinématique}* liens = {l21}	E = *{cap}* F = *{cap}* liens = {l31*}*	E = *{vitesse}* F = *{vitesse}* liens = {l32}
	g11 Ei = *{cap, vitesse}* Fi = *{cinématique}* liens = {l11, l12}	g12 Ei = *{cinématique}* Fi = *{cinématique}* liens = {l21}	g21 Ei = *{cap}* Fi = *{cap}* liens = {l31}	g22 Ei = *{vitesse}* Fi = *{vitesse}* liens = {l32}
card(Ei)	2	1	1	1
card(Fi)	1	1	1	1
card(gi)	2	1	1	1
cardMin(gi)	2	1	1	1
densité(gi)	1	1	1	1
poids(gi)	2	1	1	1
canonicité(gi)	1/2	1	1	1
couverture(E)	1	1	1	1
couverture(F)	1	1	1	1
	En final ...			
	g1		g2	
qualitéμ(gi)	3/4		1	
	D'où ...			
qualité	11/16			

Cette qualité de 11/16 invite le concepteur à vérifier son application et notamment le bien fondé des écarts de modélisation avec la référence.

Ces critères et métriques « centrés domaine » étant ainsi définis, consacrons-nous maintenant au deuxième volet de CriMeRéact, à savoir : les propriétés d'utilisabilité.

3.2.2. Utilisabilité

Nous nous référons ici aux propriétés mentionnées dans le cahier des charges (chapitre III) :

- elles sont issues de [IFIP 96] pour les affinements de l'utilisabilité en souplesse et robustesse ;
- elles proviennent de notre espace 3Co pour la mesure de l'efficacité du comportement opérateur.

Aujourd'hui, les premières ne sont envisagées qu'en partie : seules les observabilité, représentation multiple et curabilité arrière sont traitées. Nous leur associons les métriques suivantes :

Observabilité

L'observabilité traite du caractère perceptible de l'état interne du système. Dans l'hypothèse où l'observabilité globale dépend de l'observabilité partielle des différents constituants du système, nous postulons que :

$$\text{observabilité} = \Sigma_i \text{ observabilité}(e_i) / \Sigma_i;$$

où :

- i parcourt les entités CoTaS du modèle normatif, à savoir ses concepts, tâches et stratégies ;
- et observabilité(e_i) mesure l'observabilité de l'entité e_i.

Si cette observabilité(e_i) transparaît sur les liens PATCH HL et LL , nous nous limitons aujourd'hui aux liens HL. Ainsi :

- dans le cas d'un concept :
  observabilité(ei) = 1 si au moins un lien PatCH HL est associé à ei ; 0 sinon ;
- dans le cas d'une tâche ou stratégie :
  observabilité(ei) = 1 si au moins un lien PaTcH HL est associé à ei ; 0 sinon.

Multiplicité de la représentation

Dans le même esprit que pour l'observabilité, nous travaillons ici sur les liens PATCH HL associés aux entités CoTaS du modèle normatif. La représentation est dite multiple pour une entité e_i de cet espace si le nombre de liens PatCH HL la concernant est supérieur ou égal à 2.

$$\text{multiplicité} = \Sigma_i \text{ multiplicité}(e_i) / \Sigma_i;$$

où :

- i parcourt les entités du modèle normatif, à savoir les concepts, tâches et stratégies;
- multiplicité(ei) = 1 si au moins deux liens respectivement PatCH et PaTcH HL sont associés à l'entité e_i, respectivement concept ou tâche ; 0 sinon.

Curabilité arrière

Si la curabilité se réfère, de façon générale, à la possibilité de corriger une situation non désirée, nous nous limitons ici à la curabilité arrière. Nous mesurons, pour ce faire et pour toute tâche, l'existence de moyens d'annulation. La correction du traitement d'annulation échappe, par contre, à la mesure. Nous proposons, sur cette base, la métrique prédictive suivante :

$$\text{curabilité} = \Sigma_i \text{ curabilité}(e_i) / \Sigma_i$$

où :

- i parcourt les tâches du modèle normatif ;
- curabilité(e_i) = 1 si e_i possède une fonction d'annulation ; 0 sinon.

Aujourd'hui, ces métriques ne sont appréhendées qu'en évaluation prédictive. CatchIt suggère, par contre, leur vérification manuelle en cas d'anomalie dans l'interaction. Ces suggestions relèvent des propriétés 3Co. Nous en précisons maintenant les métriques.

Complétude opératoire

Dans notre terminologie CoTaS, la complétude opératoire vérifie que toute tâche ou stratégie prévue est effective. La métrique proposée intègre les degrés de nécessité associés aux stratégies :

$$\text{complétudeOp} = \Sigma_i \text{ complétudeOp}(e_i) / \Sigma_i$$

où :

- i parcourt les tâches non facultatives, les stratégies conseillées et impératives du modèle normatif ;
- complétudeOp(e_i)
  - = 1 si e_i est effectivement mise en œuvre par l'opérateur ;
  - = 0.5 sinon, dans le cas d'une stratégie conseillée ;
  - = 0 sinon, dans le cas d'une tâche non facultative ou stratégie impérative.

Une valeur de 1 (c'est-à-dire 100%) signifie un respect fidèle du plan de résolution.

Correction opératoire

Dans cette même terminologie CoTaS, la correction opératoire vérifie que toute action physique opérateur s'inscrit dans une tâche ou stratégie légitime. Cette légitimité est vérifiée sur la base des préconditions des tâches et conditions des stratégies :

$$\text{correctionOp} = \Sigma_i \text{ correctionOp}(e_i) / \Sigma_i$$

où :

- i parcourt les tâches effectives ;
- correctionOp(e_i)
  - = 0 pour une tâche de précondition non satisfaite ou une stratégie interdite ;
  - = 0.5 pour une stratégie déconseillée ;
  - = 1 sinon.

Une valeur de 1 dénote une correction parfaite.

Concision opératoire

Dans cette même lignée, la concision encourage au strict nécessaire. Elle condamne l'exécution de tâches facultatives ou annulées ou non prévues et de stratégies autorisées mais ni conseillées, ni impératives.

$$concisionOp = \Sigma_i\ concisionOp(e_i) / \Sigma_i$$

où :

- i parcourt les tâches effectives ;
- $concisionOp(e_i)$
  - = 0 pour une tâche facultative ou annulée ou non prévue ou une stratégie autorisée ;
  - = 1 sinon.

Une valeur de 1 dénote une parfaite concision.

Le tableau suivant résume les modalités prédictive et expérimentale d'évaluation réactive de ces métriques dans CatchIt 98.

Propriétés en réactivité	Prédictif	Expérimental
Domaine		
Couverture et correction	●	-
Utilisabilité		
Multiplicité de la représentation	●	-
Observabilité	●	-
Curabilité	●	-
Complétude opératoire	-	●
Correction opératoire	-	●
Concision opératoire	-	●

Ainsi s'achève la description de CriMeRéact. Nous nous consacrons, dans la section suivante, aux mécanismes d'exploitation. Nous y ciblons résolument l'évaluation des 3Co.

3.3. Mécanismes

Nous occultons ici les mécanismes d'évaluation des métriques associées aux propriétés d'utilisabilité : ils feront l'objet d'une description dans le chapitre dédié à la mise en œuvre. Nous nous focalisons sur l'évaluation expérimentale : elle repose sur une observation de l'opérateur. Nous en décrivons ici les principes.

3.3.1. Observation

Par définition, la notion de contexte se réfère à des « circonstances ou situation globale où se situe un événement » [Larousse 94]. Selon la dualité habituelle état/événement, le contexte désigne donc un état situationnel, au sein duquel s'exercent des événements de natures diverses. C'est la définition que nous adoptons pour notre contexte d'interaction : il désigne la situation courante d'interaction.

Si cette situation se réfère typiquement à une configuration instantanée des concepts applicatifs, elle peut aussi porter sur des versions antérieures de ces mêmes concepts. Par exemple, en guerre électronique, l'échec d'une tactique sur une piste peut être déterminant dans les traitements futurs : sa prise en compte, dans le contexte d'interaction, est donc fondamentale. Nous postulons ici la modélisation de ces traits sémantiques par le biais des attributs des concepts de CoTaS. Ainsi, les concepts couvrent-ils la modélisation du contexte d'interaction (Figure 77).

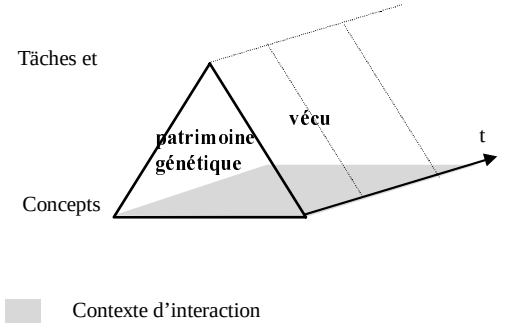

Figure 77 : Le contexte d'interaction se réfère à la situation courante. Cette situation est non seulement instantanée, mais aussi fonction des événements passés : sa modélisation est couverte par les concepts de l'espace CoTaS, grâce aux attributs, véhicules sémantiques génériques.

Si le contexte d'interaction est certes présent dans l'application, il y reste souvent diffus et implicite. Il s'agit ici de l'expliciter pour vérifier les complétude, correction et concision du comportement opérateur. Le modèle normatif étant le « représentant agréé » de la sémantique opérationnelle, il l'esquisse, a priori, au plus juste. D'où, l'idée de reproduire, sur le modèle normatif, tout événement applicatif et de maintenir ainsi, en parallèle, côté modèle normatif, un contexte d'interaction explicite. C'est donc concrètement d'une propagation d'événements dont il s'agit ici : les liens PATCH en sont le médiateur.

Les liens pAtCH connectent, à deux niveaux d'abstraction (HL et LL), les concepts de l'application et du modèle normatif. Il s'agit de les exploiter pour acheminer tout événement de l'application vers le modèle normatif. Ces événements, quels sont-ils ? Ceux qui modifient les concepts du modèle normatif. D'où l'idée :

- d'instrumenter, à l'insu du développeur, toute méthode applicative modifiant l'un ou l'autre de ces attributs ;
- de propager vers le modèle normatif, conformément aux liens pAtCH HL et LL, l'événement ainsi capturé, à savoir : le concept cible, l'attribut concerné et sa nouvelle valeur ;
- d'instancier et de maintenir, selon ces stimuli, le modèle normatif pour qu'il incarne, à tout instant, une représentation fidèle du contexte d'interaction.

C'est un composant dédié, l'analyseur, qui assume ces fonctions d'instrumentation et de propagation :

- l'instrumentation du code est un préliminaire à l'évaluation expérimentale. Elle consiste à insérer des instructions espion dans les méthodes applicatives susceptibles de modifier des attributs applicatifs ayant un correspondant dans le modèle normatif. L'analyseur considère, à ces fins, les concepts du modèle normatif à leur grain le plus fin, à savoir les attributs. Il parcourt, en sens inverse, les liens pAtCH LL établis entre l'application et le modèle normatif afin d'identifier, dans celle-ci, toute méthode accédant en écriture à des attributs applicatifs connectés à des homologues normatifs. Il instrumente alors ces méthodes, c'est-à-dire y insère une instruction espion pour rediriger, vers l'analyseur, cet événement ;
- à l'exécution, l'analyseur intercepte ainsi tout événement applicatif, concernant le contexte d'interaction. Il s'agit alors de formuler, dans les termes du modèle normatif, l'effet de ces stimuli. L'analyseur gère, à ces fins, un ensemble de proxys, représentant, côté modèle normatif, les concepts applicatifs : c'est concrètement une instanciation des concepts du modèle normatif et c'est cette instanciation qui constitue le contexte d'interaction. Tandis que les proxys sont créés conformément aux liens pAtCH HL, ils sont mis à jour selon les liens LL.

Considérons, par exemple, une application mettant en œuvre une situation tactique, composée d'un porteur. Supposons-les modélisés et connectés, comme suit (Figure 78).

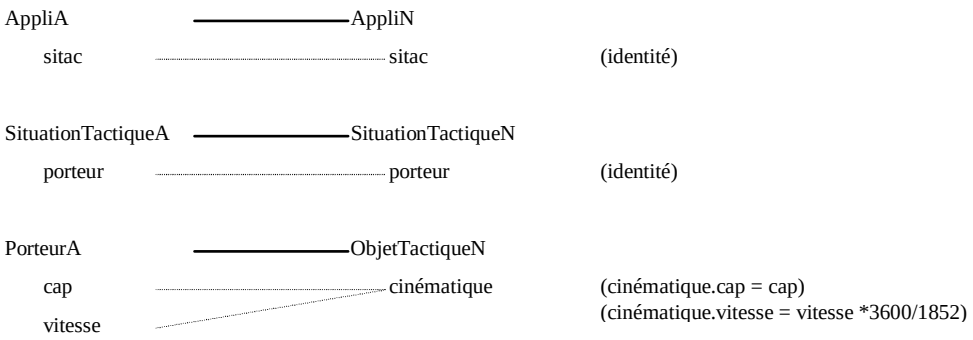

Figure 78: Exemple support à l'illustration du maintien du contexte d'interaction : les classes de l'application et du modèle normatif sont respectivement suffixées par A et N. Elles sont connectées par des liens pAtCH HL, matérialisés par des traits pleins. Leurs attributs sont reliés par des liens de même nature mais de granularité LL, représentés en pointillé. Les règles de correspondance sont mentionnées entre parenthèses.

Par hypothèse :

- la situation tactique est créée et initialisée au lancement de l'application dans la méthode *initSitac* de la classe *AppliA* ;
- l'initialisation de cette situation se limite à celle du porteur, via la méthode *initPorteur* de *SituationTactiqueA* ;
- les données du porteur sont modifiables via les méthodes *setCap(aFloat)* et *setVitesse(aFloat)*, définies dans *PorteurA.*

Ces quatre méthodes, à savoir *initSitac*, *initPorteur*, *setCap* et *setVitesse* sont instrumentées puisqu'elles accèdent en écriture à des attributs connectés à des classes normatives. Le lancement de l'application *appliA* déclenche, à l'exécution, via cette instrumentation, les traitements suivants (Figure 79) :

- dans *initSitac* : association d'un proxy *appliN* à *appliA* (une instance de la classe *AppliN*, conformément au lien HL tissé entre *AppliA* et *AppliN*). Puis, association d'un proxy *sitacN* (une instance de *SituationTactiqueN*) à la situation tactique *sitacA* et affectation de ce proxy à l'attribut *sitac* du proxy de l'application, à savoir *appliN* ;
- dans *initPorteur* : association d'un proxy *porteurN* (une instance de *PorteurN*) au porteur *porteurA* de *sitacA*; initialisation de ce porteur selon la règle de correspondance spécifiée et affectation de cette valeur à l'attribut *porteur* du proxy *sitacN* de la situation tactique *sitacA*.

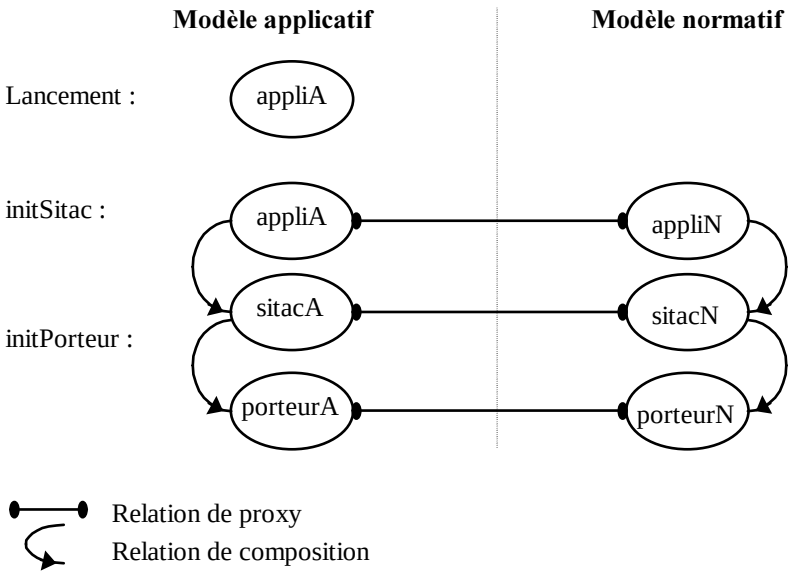

Figure 79 : Illustration des principes d'instanciation des proxys.

Si maintenant, en dynamique, le porteur évolue via ses méthodes d'accès *setCap* et *setVitesse*, alors son proxy *porteurN* évoluera selon cette même loi. Ce sont les règles de correspondance spécifiées qui permettent ces recalculs. Dans notre cas, les exécutions de *setCap(unCap)* et *setVitesse(uneVitesse)* engendreront respectivement les mises à jour suivantes :

- *porteurA->cap = unCap* et, en conséquence, *proxyN->cinematique.cap = unCap* ;
- *porteurA->vitesse = uneVitesse* et, en conséquence, *proxyN->cinematique.vitesse = uneVitesse** 3600/1852.

Ainsi le proxy de l'application (*appliN*) s'apparente-t-il à une grappe de proxys : il incarne le contexte d'interaction. Sa gestion est totalement assumée par CatchIt. C'est sur ce critère que l'instrumentation est dite automatique. Sa transparence provient, quant à elle, de son imperceptibilité : les instructions espion restent toujours masquées au développeur, même lorsque ce dernier parcourt ses méthodes. Les principes sous-jacents à cette transparence étant fortement dépendants de l'environnement de développement adopté, ils feront l'objet d'une description détaillée dans le chapitre suivant dédié à la mise en œuvre de CatchIt. Nous nous consacrons maintenant à la description de la trace d'interaction.

La trace d'interaction se réfère aux tâches opérateur effectives. Elle s'exprime en termes de tâches CoTaS, estampillées d'un instant d'occurrence. La seule manifestation de ces dernières étant les actions physiques déployées sur les dispositifs d'entrée, la capture s'effectue à ce niveau d'abstraction.

De façon plus détaillée, CatchIt identifie, dans l'application, l'espace de travail hôte et l'élément de présentation activé. Cette identification et le contexte d'interaction consignent l'observation médiatisée : ils alimentent l'analyse dont nous décrivons maintenant le principe.

3.3.2. Analyse

L'idée consiste ici à inférer, sur la base des liens PaTcH, les tâches CoTaS correspondantes. CatchIt considère, pour ce faire, le modèle normatif et infère des liens PaTcH HL les tâches opérateur compatibles de l'action physique capturée. Aujourd'hui, les algorithmes mis en œuvre supposent des espaces de travail dédiés, c'est-à-dire affectés à des tâches élémentaires données : tâches élémentaires et espaces sont bijectivement connectés et l'identification d'un espace suffit à identifier la tâche opérateur courante. Notons que cette hypothèse, certes réductrice, est loin d'être artificielle et simpliste : elle répond à une règle d'ergonomie visant à limiter tout risque de confusion opérateur. Elle est souvent respectée dans les applications développées à Thomson-CSF RCM.

La tâche opérateur concernée par l'action physique étant ainsi identifiée, l'analyse consulte les éventuelles modalités d'activation, validation et annulation associées. Elle transmet au module de critique la tâche ainsi valuée.

3.3.3. Critique

Les critiques supposent, non seulement, une caractérisation, mais aussi des mécanismes de détection, explication et publication :

- d'un point de vue caractérisation, nous ciblons ici les critiques négatives. Elles portent sur les 3Co ;
- nous distinguons deux mécanismes de détection :
  - le premier, déclenché par le module d'analyse, vérifie le bien fondé du comportement opérateur : il considère, à ces fins, les descriptions CoTaS qu'il confronte au contexte d'interaction ;
  - le second, tel un démon, vérifie, en permanence, le respect des stratégies CoTaS conseillées et impératives. Il pourrait, dans le principe, être explicitement déclenché par CatchIt sur mise à jour du contexte d'interaction : il suffirait, pour cela, d'étendre l'instrumentation aux méthodes des concepts normatifs ;
- en l'absence de modèle opérateur et notamment de modèle des écarts opérateur, les informations additionnelles s'apparentent ici à de la justification et non de l'explication : les informations portent non pas sur la raison d'être du comportement opérateur observé, mais sur la détection des critiques : elles justifient l'annonce en précisant, d'une part, le contexte d'interaction courant, d'autre part les tâches ou stratégies prévue et effective. C'est donc bien dans une démarche de traçabilité que s'inscrit cette notion de justification ;
- l'explication se limite aujourd'hui à la suggestion de points de contrôle, en termes d'observabilité. Il s'agit plus précisément de la non observabilité des préconditions et états initiaux des tâches et stratégies de CoTaS. La vérification n'est pas assumée par CatchIt : l'outil suggère ces axes d'explication ; au développeur d'en vérifier le bien fondé. Supposons, par exemple, qu'une nouvelle piste apparaîsse, que l'opérateur n'en examine pas les paramètres alors qu'une stratégie stipule cet examen. CatchIt invitera alors le développeur à vérifier l'observabilité de la condition « une nouvelle piste apparaît » ;
- la publication se réfère à l'acte de notification. Deux modes sont offerts : les publications à la volée et en différé. Elles transmettent à l'opérateur, justifications à l'appui, les critiques détectées, en précisant notamment leurs polarité (aujourd'hui négative) et nature (cf métriques 3Co). Pour notre exemple : anomalie de type complétude, sur la condition « une nouvelle piste apparaît », au regard de la stratégie « si une nouvelle piste apparaît alors surveiller piste impératif ».

La figure 80 résume les principes sous-jacents à l'évaluation expérimentale.

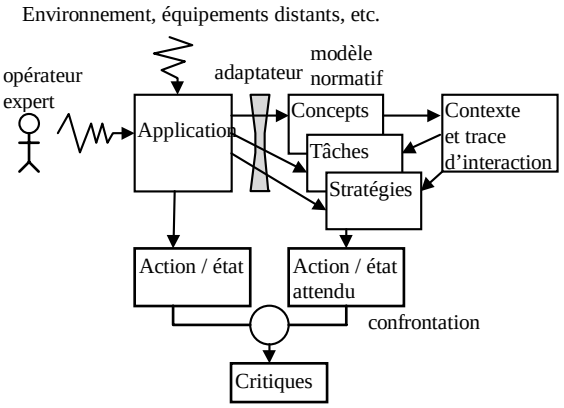

Figure 80 : Principe de l'évaluation expérimentale : l'application, telle un système ouvert, est soumise à des perturbations externes d'origine opérateur ou système. Ces stimuli se consignent, dans l'application, en termes d'action opérateur ou de changement d'état. Ils sont propagés, tels une onde, vers le modèle normatif : prescripteur du fonctionnement nominal, ce dernier maintient des contexte et trace d'interaction, desquels il infère le comportement idoine. Il reste alors à confronter l'effectif et l'attendu pour mesurer, non seulement l'adéquation homme-machine, mais aussi la conformité fonctionnelle de l'application, ceci en regard de cette référence. Les critiques sont le fruit de cette phase de confrontation.

En cas de critiques négatives, une rétroaction logicielle s'impose pour corriger l'anomalie. Cette intervention incombe au développeur. Les justifications et explications fournies par CatchIt le guident dans sa tâche.

Ainsi s'achève la description des principes de CatchIt. Avant d'en évoquer la mise en œuvre, nous en proposons maintenant une synthèse et discussion.

4. Synthèse et discussion

Cette section propose une prise de recul quant aux principes de CatchIt. Il rappelle, après une synthèse de ses fils directeurs, les hypothèses et limites de l'approche. Il cloture alors la description par des notes métaphoriques.

4.1. Synthèse

CatchIt procède, rappelons-le, par étalonnage : il suppose l'existence d'une norme, dite modèle normatif, consignant, à un instant donné, les connaissances métier de l'entreprise. Il évalue l'application selon cette référence et déploie, à ces fins, deux espaces :

- CoTaS pour fixer la granularité des entités modélisées : concepts, tâches et stratégies ;
- DEGRE pour gérer, en termes de généricité, affinement et évolutivité les informations ainsi capitalisées.

Si la réutilisation logicielle de ce noyau améliore logiquement la conformité de l'application, elle ne dispense néanmoins pas d'une évaluation de l'interaction. CatchIt complète alors sa proactivité par une facette réactive, fondée sur une connexion déclarative entre l'application et cette référence. Pour résumer les principes de la connexion, rappelons qu'ils s'articulent autour des éléments suivants :

- un tissage de liens déclaratifs entre l'application et le modèle normatif ;
- la proposition d'une taxonomie PATCH, dimensionnant l'espace problème par l'identification de quatre types de liens ;
- l'adoption d'une seule et unique modélisation fédératrice pour ces quatre types de liens ;
- l'opportunité d'un double niveau de détail, par les granularités HL et LL ;
- et l'introduction d'un composant dédié, l'adaptateur, gérant, de façon externe à l'application, les liens sémantiques ainsi tissés.

Bien entendu, si l'application dérive du modèle normatif, alors le tissage des liens, en termes d'abstraction, est immédiat : il s'apparente à une bijection entre les noyaux fonctionnels de ces deux ensembles. A l'inverse, une application non structurée nécessitera une rétroconception pour l'établissement de liens, peut-être non triviaux. Mais, c'est alors peut-être l'occasion d'une prise de recul quant au code applicatif.

Si cette connexion entre les modèles applicatif et normatif incombe au développeur, la prestation de CatchIt est ensuite totalement automatisée : CatchIt assume notamment l'instrumentation du code. La figure 81 schématise cet espace d'automatisation. Elle en rappelle les données d'entrée et de sortie.

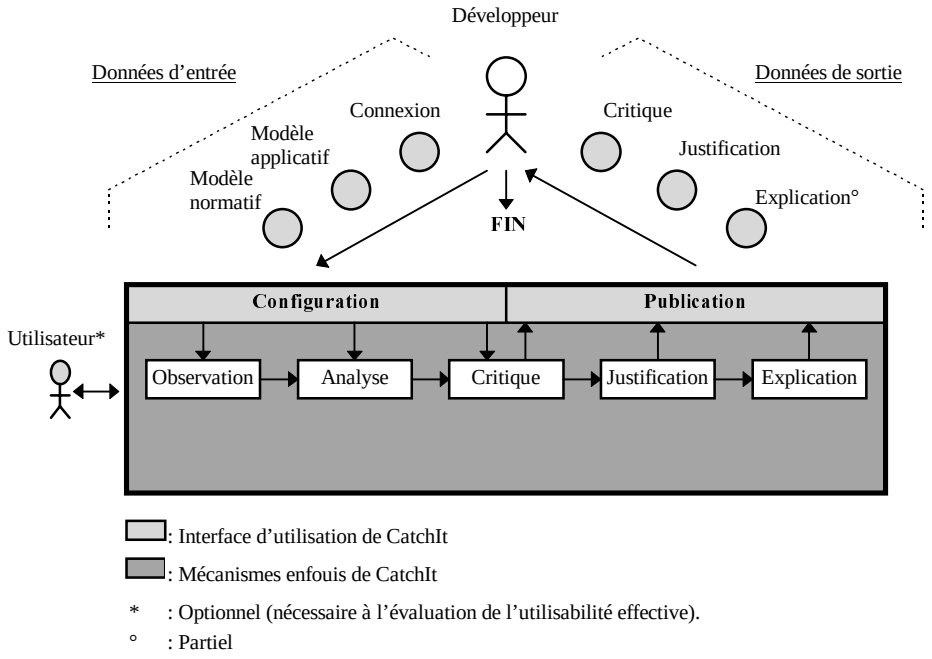

Figure 81 : Automatisation de la prestation réactive de CatchIt.

De façon générale, les critères et métriques sont centraux à l'approche. Ils mesurent :

- en proactivité, le taux de réutilisabilité : espace CriMePro (Critères et Métriques en Proactivité) ;
- en réactivité, la distance entre les modèles applicatif et normatif, en termes de couverture/correction de la modélisation et respect des propriétés d'utilisabilité : espace CriMeRéact (Critères et Métriques en Réactivité).

Avant d'illustrer CatchIt par des métaphores, rappelons-en les hypothèses et limites.

4.2. Hypothèses et limites

CatchIt s'adresse, en tout premier lieu, à des applications orientées-objet. Si cette consigne est aujourd'hui, dans l'ère du temps et rend ainsi candidates les nouvelles applications, elle exclut néanmoins tout système ancien, à moins d'une reconception. L'hypothèse est donc forte et se doit d'être soulignée. Notons aussi, pour l'instrumentation, la nécessité absolue de disposer du code source : c'est le cas pour nos applications.

Soulignons, par contre, l'investissement long-terme requis en termes de proactivité. Un point essentiel et déterminant dans CatchIt est, en effet, l'appréciation des niveaux de généricité des informations modélisées. Il s'agit bien pour chacune d'elles de préciser si elle est générale, propre au domaine ou, au contraire, spécifique à l'application traitée. Cette analyse incombe aux experts et peut, dans le court terme, être perçue comme un surcroît de travail. Mais elle force, sans aucun doute, à une maturité et une prise de recul par rapport aux données manipulées : elle aide à la consolidation des connaissances et confère, à leur modélisation, une unité sémantique.

Nous cloturons maintenant ce chapitre par un ensemble de métaphores.

4.3. Métaphores

Si le registre médical nous a permis d'illustrer l'auscultation externe de l'application par CatchIt et son rôle prescripteur, et non administrateur de remèdes, nous changeons ici de registre :

- mentionnons, en mathématiques, les vecteurs propres des matrices. Déjà appliqués en analyse d'images pour la reconnaissance de visages [Turk 90], ils peuvent ici être instanciés sous la forme d' « eigen domains » : incarnés par le modèle normatif, ils en dégagent les facettes saillantes. Ces vecteurs propres

doivent être préservés dans l'application, ce qui, notons-le, n'impose pas l'identité des matrices. Rappelons en effet que des opérations, telle la multiplication par un scalaire, préserve les vecteurs propres ;

- évoquons, dans la lignée de Wiener et Curry [Wiener 89] , la notion de « cocon informatique ». Ces derniers introduisent, en fait, le terme de « cocon électronique » pour désigner l'enveloppe de sécurité qu'intègrent aujourd'hui, en aéronautique, les systèmes intelligents : tant que l'avion reste dans ce cocon, l'équipage est libre de réaliser toutes les actions et manœuvres qu'il souhaite, sans justification apparente. Si l'avion franchit, par contre, le cocon, alors l'équipage est immédiatement alerté. Par analogie, le cocon informatique matérialise ici l'espace de liberté accordé aux équipes de développement : l'imagination est permise, mais dans le respect des entités CoTaS ;
- en métrologie, CatchIt offre fils électriques et instruments de mesure, mais l'apposition des broches incombe au développeur. Il est le seul à maîtriser l'installation et connaître les normes en la matière ;
- enfin, dans le domaine ludique, modèles applicatif et normatif représentent des portions d'une même image. Mais les niveaux de détails peuvent varier et les images ont été morcelées de façons potentiellement différentes : CatchIt invite le développeur à la reconstitution du puzzle applicatif sur la base du modèle normatif.

Le chapitre suivant décrit la mise en œuvre de CatchIt.

CHAPITRE VIII : MISE EN ŒUVRE

CatchIt est aujourd'hui implémenté en Smalltalk dont nous proposons une présentation synthétique en annexe B. L'ambition y reste pragmatique, l'objectif bien ciblé : il ne s'agit pas de présenter Smalltalk dans l'absolu, mais d'apporter au lecteur les éléments d'information nécessaires à la compréhension de la mise en œuvre ici décrite.

Cette présentation cible la logique de fonctionnement de CatchIt. Elle s'articule en deux parties : tandis que la première porte sur la structure d'accueil en termes de connaisssances, la seconde traite des mécanismes de CatchIt. La perspective utilisation de l'outil est illustrée au chapitre suivant.

1. Modélisation

Dans cette description, et en réponse à l'absence de frontière, dans Smalltalk, entre l'application et le système (cf Annexe B), nous adoptons la convention suivante : préfixer toute classe applicative par la chaîne distinctive *CI* (pour CatchIt). Si cette convention permet de circonscrire notre contribution logicielle, elle permet aussi, en conséquence, d'apprécier la forte réutilisation, dans Smalltalk, des classes système.

Nous étayons la description de modèles objet OMT. Pour une meilleure lisibilité, nous n'explicitons pas les relations de composition. Nous mentionnons, par contre, entre chevrons le type des différents attributs.

Avant d'envisager la mise en œuvre des connaissances et connexion, nous décrivons le point d'entrée de l'outil : la classe *CICatchIt.*

1.1. CatchIt

CatchIt est modélisé par la classe abstraite *CICatchIt* (Figure 82). *CICatchIt* hérite de *Object* et met en œuvre trois variables de classe :

- *étalonnage*, un *CIEtalonnage*, qui pointe sur l'étalonnage courant ;
- *étalonnages*, une liste de *CIEtalonnage*, capitalisant les étalonnages passés. Cette historisation permet, dans le long-terme, de recouvrer des configurations d'évaluation. Ce service est fondamental pour les tests de non régression ;
- *domaines*, une liste de *CIModele*, consignant les domaines capitalisés.

CIEtalonnage hérite de *Object*. Elle est définie pas trois attributs :

- *modeleApplicatif*, un *CIModele*, identifiant l'application traitée ;
- *modeleNormatif*, un *CIModeleNormatif*, désignant le modèle de référence ;
- *adaptateur*, un *CIAdaptateur*, mémorisant la connexion établie entre ces deux modèles.

CIModele, *CIModeleNormatif* et *CIAdaptateur* relèvent des connaissances et connexion. Ils feront l'objet de description dans ces sections.

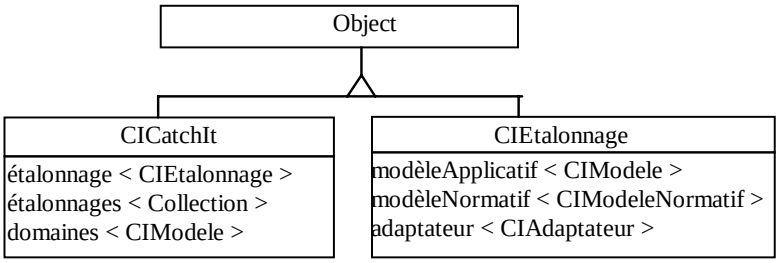

Figure 82 : Description OMT de la mise en œuvre de CatchIt. Le type des attributs y est spécifié entre chevrons.

1.2. Connaissances

Rappelons que deux espaces sous-tendent la modélisation des connaissances : CoTaS et DEGRE. Si dans DEGRE, généricité et évolutivité sont à l'appréciation du développeur, CatchIt 98 fixe à l'inverse la dimension

affinement : les entités CoTaS sont spécifiées en langage informatique orienté-objet : en l'occurrence Smalltalk. La mise en œuvre tient compte de cette restriction.

1.2.1. Espace CoTaS

L'espace CoTaS préconise une structuration tripartite des connaissances en concepts, tâches et stratégies. Si les concepts y restent une notion générique, définie à base d'attributs et de méthodes, tâches et stratégies justifient, en revanche, l'introduction de classes dédiées.

Dans un effort d'unification (Figure 83) :

- nous définissons une classe mère, *CIEntité*, fédératrice des entités de CoTaS. *CIEntité* hérite de *Object* et gère les proxys de l'entité considérée (aujourd'hui seuls les concepts sont assortis de proxys) ;
- nous associons les classes dédiées *CIConcept*, *CITache* et *CIStrategie* aux concepts, tâches et stratégies. Toutes trois héritent de *CIEntité*.

CITache

CITache implique quatre attributs :

- *identification* consigne l'identification de la tâche : c'est une instance de la classe dédiée *CITIdentification*. *CITIdentification* hérite de *Object*. Elle est définie par un *nom* et un *numero*. Aujourd'hui, seul le nom est exploité ;
- *elements* modélise les éléments de la tâche : c'est une instance de la classe dédiée *CITElements*. *CITElements* hérite de *Object*. Elle est définie par six attributs :
  - *but* est une formulation textuelle de l'objectif de la tâche ;
  - *etatInitial* est une liste d'objets décorés d'un poids. C'est, en pratique, une liste de points < objet, poids >, le poids étant un symbole ;
  - *precondition* est une instance de la classe dédiée *CICondition*, ci-après décrite ;
  - *corps* est un *CICorps*. *CICorps* hérite de *Object* et est définie par un attribut *structure*. C'est une liste ordonnée d'éléments hétérogènes : *CITache* pour les tâches constituantes et *Symbol* pour les opérateurs de composition ;
  - *etatFinal* adopte la modélisation de *etatInitial* ;
  - *postcondition* adopte la modélisation de *precondition* ;
- *attributs* regroupe les décorations de tâche : c'est une instance de la classe dédiée *CITAttributs*. Cette classe hérite de *Object*. Elle est définie par quatre attributs booléens : *optionnel*, *prioritaire*, *interruptible* et *itératif* ;
- *effecteur* consigne les acteurs prévus et potentiels : c'est une instance de la classe dédiée *CITEffecteur*. *CITEffecteur* hérite de *Object*. Elle est définie par deux attributs, *prevus* et *potentiels*. Aujourd'hui, seul le premier est traité. Il se limite à un acteur : c'est une liste restreinte à un symbole.

CIStrategie

CIStrategie implique trois attributs :

- *condition* est une *CICondition*, ci-après décrite ;
- *action* est un *CICorps* ;
- *valuation* est un *Symbol* de valeur #imperatif, #conseille, #autorise, #deconseille ou #interdit.

CICondition

CICondition modélise, de façon générale, les conditions. Elle hérite de *Object* et est définie par quatre attributs :

- *texte* est une description textuelle de la condition ;
- *nature* explicite la nature de la condition : état du monde, déclenchante, opérateur, résultat ou fin de tâche ;
- *type* est une prémisse à la formulation de la métrique d'évaluation de la condition : il précise si cette condition porte sur un état ou un événement. Cette information est exploitée lors de l'instrumentation ;
- *expression* est la métrique d'évaluation de la condition. C'est une instance de la classe *CIExpression* ;

CIExpression

CIExpression hérite de *Object*. Elle est définie par trois attributs, le premier étant exclusif des deux autres :

- *bloc* est un bloc d'instructions, au sens de Smalltalk, c'est-à-dire un *BlockClosure*. Il admet des paramètres d'entrée et évalue l'expression au regard du contexte d'interaction courant. Il est adapté à la formulation de conditions de type état ;
- *classe* et *méthode* identifient respectivement les classe et méthode normatives concernées par l'événement. Elles en permettent l'instrumentation. Ce sont, en pratique, des symboles.

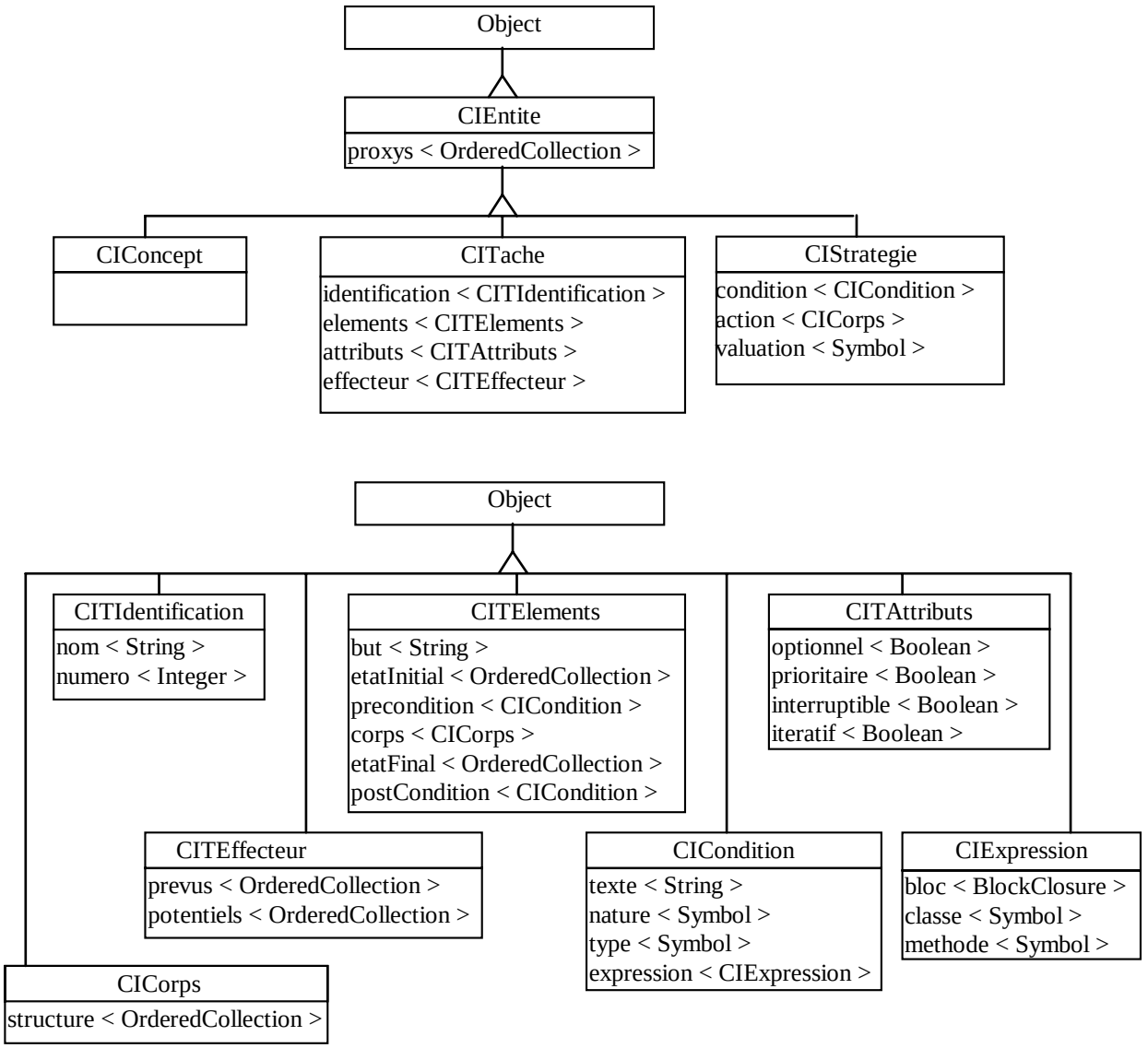

Figure 83 : Description OMT de la mise en œuvre de l'espace CoTaS. Le type des attributs y est spécifié entre chevrons.

Les connaissances étant ainsi modélisées, nous nous intéressons maintenant à leur organisation au sein de modèles : c'est l'objet de l'espace DEGRE.

1.2.2. Espace DEGRE

L'espace DEGRE s'articule autour de la notion de domaine. Un domaine est incarné par un modèle enrichi d'une localisation dans l'arborescence des domaines capitalisés.

Les modèles sont mis en œuvre par la classe dédiée *CIModele*. *CIModele* hérite de *Object*. Elle est définie par cinq attributs (Figure 84) :

- *nom* et *version*, des chaînes, identifiant le modèle ;
- *concepts*, *taches* et *strategies* des listes de *CIConcept*, *CITache* et *CIStrategie*, circonscrivant le modèle. Pour les modèles applicatifs, seuls les concepts sont renseignés.

La classe *CIModeleNormatif* modélise les modèles normatifs. Elle hérite de *CIModele* qu'elle enrichit d'un attribut *peres*. Cet attribut maintient la liste des *CIModeleNormatif* associés aux domaines que spécialise le modèle traité (Figure 84).

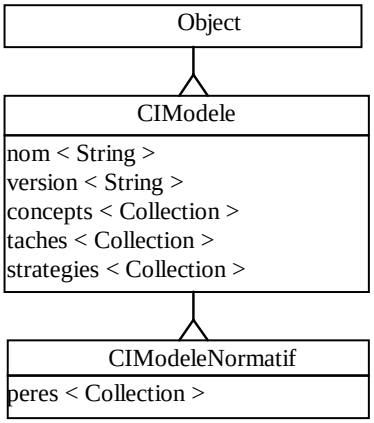

Figure 84 : Description OMT de la mise en œuvre de l'espace DEGRE. Le type des attributs y est spécifié entre chevrons.

Après CoTaS et DEGRE, étudions la mise en œuvre de la connexion déclarative entre modèles.

1.3. Connexion

L'espace PATCH préside à la connexion entre modèles. Après sa mise en œuvre, nous étudions l'organisation de ces liens au sein d'un composant dédié : l'adaptateur.

1.3.1. Espace PATCH

Conformément aux principes de CatchIt, nous introduisons une classe abstraite *CIPatch* fédérant les points communs entre les liens HL et LL (Figure 85). *CIPatch* hérite de *Object*. Elle est définie par trois attributs :

- *entiteA* et *entiteN* identifient respectivement les entités des modèles applicatif et normatif impliquées dans le lien ;
- *condition* exprime la condition de mise en correspondance. C'est une *CICondition* de type état.

Deux classes dédiées *CIPatchHL* et *CIPatchLL* sont alors introduites pour respectivement incarner les liens HL et LL. Elles héritent de *CIPatch*.

- *CIPatchHL* enrichit la modélisation d'un attribut *liensLL* : une liste de liens *CIPatchLL* ;
- *CIPatchLL* rajoute un attribut *règle* : une *CIExpression* de type bloc formulant l'éventuelle règle de mise en correspondance.

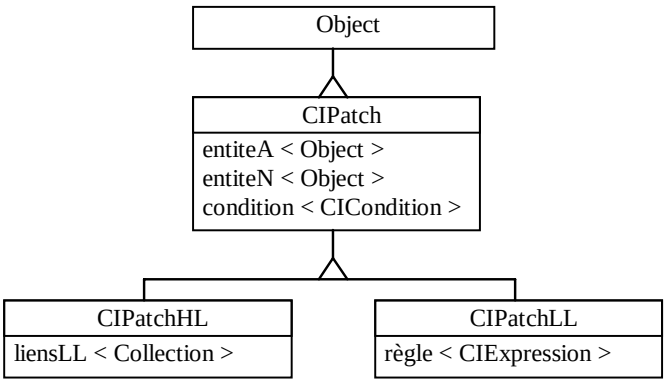

Figure 85 : Description OMT de la mise en œuvre de l'espace PATCH. Le type des attributs y est spécifié entre chevrons.

En pratique, trois types de liens sont aujourd'hui traités (Figure 86) :

- les pAtCH associent une abstraction aux concepts du modèle normatif . Les attributs *entiteA* et *entiteN* y sont des symboles identifiant :
  - au niveau HL, les classes applicative et normative connectées ;
  - au niveau LL, les attributs concernés ;
- les PaTcH associent une présentation aux tâches du modèle normatif :
  - au niveau HL, *entiteN* y est une *CITache* ; *entiteA* une *CITPresentation* ;
  - au niveau LL, *entiteN* y est un *Object* pointant indépendamment sur une pré/post condition ou un objet de l'état initial/final. *entiteA* y est une *CIPresentation* ;
- les PatCH associent une présentation aux concepts du modèle normatif : *entiteN* y est invariablement un symbole identifiant, au niveau HL, la classe, au niveau LL, l'attribut du concept concerné. *entiteA* y est une *CIPresentation.*

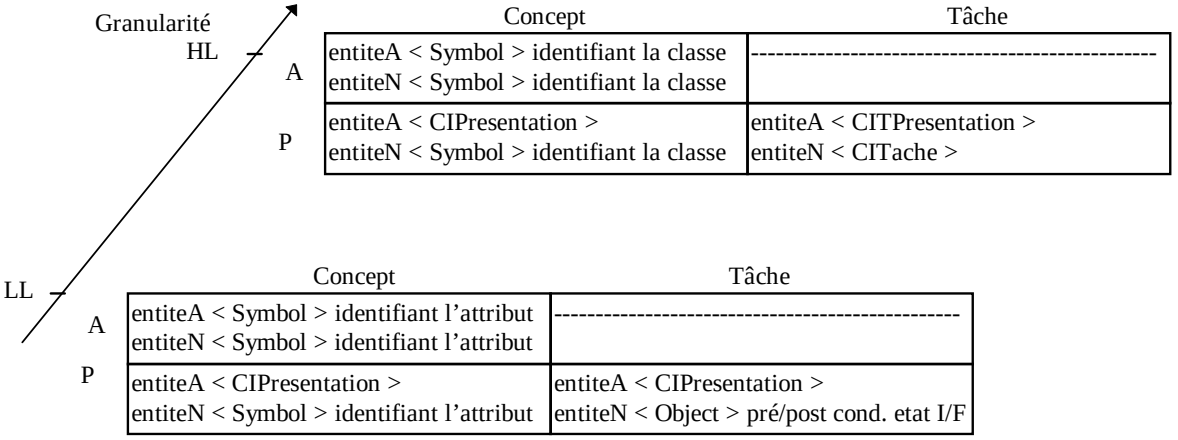

Figure 86 : Déclinaison des liens PATCH aux granularités HL et LL. Le type des attributs y est spécifié entre chevrons.

CIPresentation

CIPresentation est une classe abstraite fédérant les présentations. Elle hérite de *Object* et est définie par deux attributs (Figure 87) :

- *classeHôte* est un symbole. Il identifie la classe de l'espace de travail hébergeant l'élément considéré ;
- *hôte* est une expression retournant cet espace hôte. C'est une instance de la classe *classeHôte.*

CIPresentation se spécialise en deux sous-classes : *CIElementDePresentation* et *CIEspaceDeTravail.*

CIElementDePresentation

CIElementDePresentation traite des éléments de présentation. Elle est définie par deux attributs :

- *statut*, un symbole indiquant le statut de l'élément :
  - prédéfini si l'élément de présentation est un élément prêt à l'emploi issu, par exemple, d'une boîte à outils ;
  - défini si l'élément n'est pas prédéfini mais construit et mémorisé par l'application. C'est le cas typiquement des images ;
  - dynamique sinon. L'élément est alors construit à la volée. C'est le cas des dessins.
- *accesseur* un symbole identifiant l'élément prédéfini ou la méthode retournant l'élément défini. Ce champ est sans objet pour un objet dynamique.

CIEspaceDeTravail

CIEspaceDeTravail est définie par trois attributs :

- *classe* est un symbole identifiant la classe de l'espace de travail ;
- *constructeur* est un symbole identifiant la méthode de classe permettant l'allocation de cet espace ;
- *accesseur* est une expression retournant l'instance considérée.

Aujourd'hui seul *classe* est exploité.

CITPresentation

CITPresentation hérite de *Object*. Elle est définie par deux attributs :

- *présentation*, une *CIPresentation* ;
- *modalités*, une liste de *CIModalite.*

CIModalite

CIModalite incarne les modalités d'entrée en termes d'activation, validation et annulation. Elle hérite de *Object* et est définie par deux attributs :

- *type*, un symbole spécifiant la nature de la modalité : activation, validation ou annulation ;
- *élément*, un *CIElementDePresentation.* Les racourcis clavier/souris ne sont pas traités aujourd'hui.

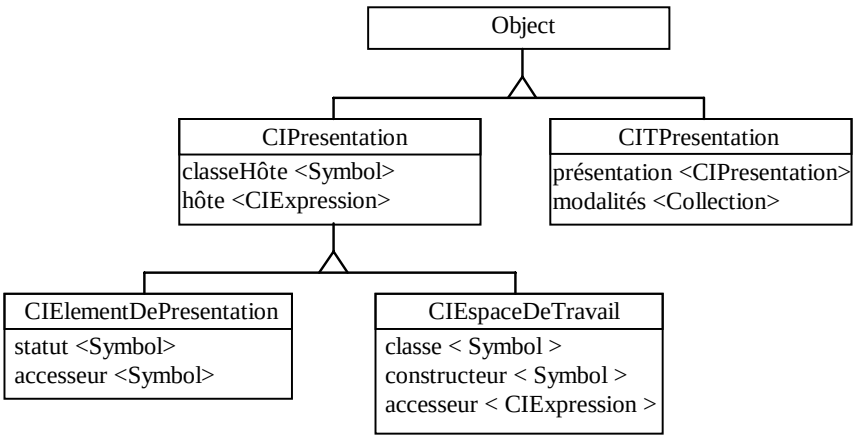

Figure 87 : Description OMT de la mise en œuvre des présentations. Le type des attributs y est spécifié entre chevrons.

Les liens étant ainsi modélisés, consacrons-nous à leur gestion : c'est l'objet de l'adaptateur.

1.3.2. Adaptateur

L'adaptateur mémorise la connexion déclarative entre les modèles applicatif et normatif. La classe *CIAdaptateur* lui est consacrée. Elle hérite de *Object* et met en œuvre cinq attributs :

- *modèleApplicatif* une référence sur le *CIModele* applicatif ;
- *modèleNormatif* une référence sur le *CIModeleNormatif* ;

- *lienspAtCH*, *liensPatCH* et *liensPaTcH* trois listes de *CIPatchHL*.

Connaissances et connexion étant ainsi modélisées, consacrons-nous aux mécanismes d'exploitation.

2. Mécanismes

Les mécanismes d'exploitation sont motivés, dans CatchIt, par les référentiels de critères et métriques : CriMePro et CriMeRéact. Nous les étudions dans cet ordre.

2.1. Espace CriMePro

CriMePro traite de la réutilisation. Il en mesure l'intensité sous deux aspects :

- d'une part, la profondeur dans l'arborescence du domaine réutilisé ;
- d'autre part, la proportion des entités CoTaS conservées.

Pour ce faire, CatchIt offre des mécanismes de navigation et sélection au sein des domaines capitalisés :

- les mécanismes de navigation adoptent la métaphore des répertoires et fichiers : le développeur navigue dans l'arborescence capitalisée (cf chapitre IX) ;
- la sélection s'effectue à la granularité du domaine. Les classes sont alors chargées dans l'image Smalltalk (cf Annexe B).

Pour ce qui est maintenant des classes réellement réutilisées, CatchIt adopte une politique binaire : dès lors que la classe est référencée, CatchIt la considère comme réutilisée. La proportion des informations ou compétences réellement exploitées échappe à l'outil. Seuls les concepts sont par ailleurs considérés.

D'un point de vue utilisation, précisons que la mesure s'effectue sur requête du développeur. Elle est groupée avec les mesures d'ordre réactif (cf chapitre IX). La section suivante est consacrée à ces dernières.

2.2. Espace CriMeRéact

Conformément aux principes de CatchIt, nous distinguons domaine, propriétés d'utilisabilité véhiculées par le système et efficacité du comportement opérateur (c'est-à-dire les 3Co). Tandis que les deux premiers sont mesurés de façon prédictive, les 3Co mobilisent une évaluation expérimentale. Nous nous focalisons ici sur ce dernier aspect. Précisons pour les autres qu'ils sont implémentés dans la classe *CIAdaptateur*. Ils portent :

- sur l'attribut *lienspAtCH* pour ce qui est du domaine ;
- sur les attributs *liensPaTcH* et *liensPatCH* pour les propriétés d'utilisabilité. Seules l'observabilité, la représentation multiple et la curabilité arrière sont ici considéres (cf chapitre VII). Tandis que les deux premières relèvent de la granularité HL, la curabilité arrière s'exprime au niveau LL.

L'évaluation des 3Co est mise en œuvre dans la classe dédiée *CI3Co*. *CI3Co* hérite de *Object* et est définie par les attributs suivants :

- *application*, une poignée sur l'application testée : c'est une instance de la classe applicative mère ;
- *contexte*, un pointeur sur le proxy de l'application : il représente le contexte d'interaction courant ;
- *actionsPhysiques*, un enregistrement factuel des actions physiques opérateur. Ces actions sont modélisées par la classe dédiée *CIActionPhysique*. Cette classe hérite de *Object* et est définie par cinq attributs :
  - *temps* consigne l'instant d'occurrence de l'action ;
  - *vue* et *contrôleur* identifient, en termes MVC, les vue et contrôleur hôtes ;
  - *objet* et *id* identifient l'élément de présentation respectivement défini ou prédéfini activé. Ces deux champs sont exclusifs ;
- *tacheCourante* représente la tâche opérateur courante, encore non achevée ;
- *tachesSuspendues* consigne les éventuelles tâches suspendues, c'est-à-dire les fils d'activité pendants ;
- *trace* est une liste de *CITacheEffective* consignant l'activité opérateur. *CITacheEffective* est dédiée à l'enregistrement de tâches effectives. Elle hérite de *Object* et est définie par quatre attributs :
  - *tDebut* et *tFin* deux entiers consignant les instants de début et de fin d'exécution de la tâche ;
  - *statut*, un symbole dénotant l'état validé ou annulé de la tâche ;

  - *actions*, la liste des *CIActionPhysique* ayant concouru à l'accomplissement de la tâche ;
- *etalonnage*, le *CIEtalonnage* au cœur de l'évaluation. Il embarque, rappelons-le, les modèles applicatif et normatif ainsi que la connexion les unissant ;
- *typageMN*, un dictionnaire mémorisant le type des attributs des concepts du modèle normatif . Ces types sont nécessaires à l'instanciation du modèle normatif. Smalltalk étant non typé, CatchIt les infère sur la base des commentaires textuels agréementant les modèles capitalisés ;
- *evolutionMA*, un dictionnaire calculant, une fois pour toutes, les méthodes d'accès en écriture aux attributs des concepts de l'application : la section suivante en rappelle la raison d'être et en expose le calcul.

Cette dernière information est centrale au mécanisme de propagation : elle sert de base à l'instrumentation automatique et transparente du code.

2.2.1. Instrumentation

L'instrumentation du code consiste en l'insertion d'instructions espions dans le code applicatif. Ces instructions redirigent, vers le modèle normatif, les événements applicatifs pertinents. Ces événements sont ceux qui modifient les concepts applicatifs, connectés au modèle normatif.

Par hypothèse, le contexte d'interaction est, en effet, ventilé au sein des concepts normatifs : il s'agit donc, en pratique, de capturer et propager vers le modèle normatif, tout événement affectant les concepts applicatifs, connectés à des homologues normatifs. Ces événements correspondent à des accès en écriture à l'un ou l'autre des attributs des concepts applicatifs, liés, par des liens pAtCH, à des homologues normatifs. Il s'agit donc :

- d'identifier les méthodes de l'application procédant à ces écritures ;
- puis de les instrumenter.

L'analyse inhérente au premier point est effectuée, une fois pour toutes, à l'initialisation. Elle travaille sur les liens pAtCH, tissés entre l'application et le modèle normatif. Ces liens sont mémorisés, rappelons-le, au sein de l'*adaptateur*. Les browsers de Smalltalk sont sollicités à ces fins. Le fruit de cette analyse est consigné dans l'attribut *evolutionMA* de la classe *CI3Co*. Cet attribut énumère les méthodes de l'application répondant au critère précité. C'est, en pratique, un dictionnaire :

- les clés répertorient les classes applicatives impliquées dans un accès en écriture : ce sont des symboles ;
- les valeurs sont des dictionnaires : les clés, des symboles, identifient les méthodes recherchées ; les valeurs indiquent les classes et attributs modifiés par l'accès en écriture. Ce sont des listes de listes à deux symboles : le premier identifie la classe ; le second la méthode accédant en écriture à l'attribut considéré.

Considérons, par exemple, la classe *CIPorteurA*, sous-classe de *CIObjetTactiqueA* (Figure 88). Supposons les respectivement définies par les attributs *type* et *position.* Considérons les méthodes *initialiser* telles que :

- dans *CIObjetTactiqueA, initialiser* affecte le point 0@0 à la position ;
- dans *CIPorteurA, initialiser* affecte la valeur *#fregate* au *type*.

Supposons, par ailleurs, que :

- dans *CIObjetTactiqueA*, la méthode *position: unPoint* affecte la valeur *unPoint* à la position ;
- dans *CIPorteurA, type: unSymbole* affecte la valeur *unSymbole* au type.

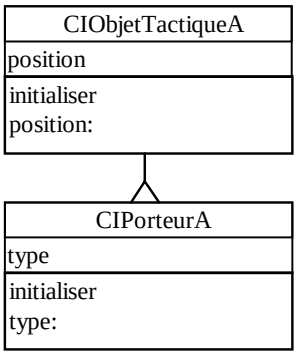

Figure 88 : Exemple support au calcul de *evolutionMA*.

Alors :

evolutionMA = [*#CIObjetTactiqueA* -> [*#initialiser* -> ((*#CIObjetTactiqueA #position*) (*#CIPorteurA #position*))
#position: -> ((*#CIObjetTactiqueA #position*) (*#CIPorteurA #position*))]
#CIPorteurA -> [*#initialiser* -> ((*#CIPorteurA #type*))
#type: -> (*#CIPorteurA #type*))]]

Il s'agit alors d'instrumenter les méthodes ainsi désignées. Pour notre exemple :

- *initialiser* et *position:* de *CIObjetTactiqueA* ;
- *initialiser* et *type:* de *CIPorteurA.*

L'instrumentation consiste en l'insertion d'une instruction espion avant tout *return*. Cette instruction redirige, vers *CI3Co*, l'événement ainsi capturé :

CI3Co propagerExecMethode: nomMethode impactant: self

où :

- *nomMethode* est un symbole identifiant la méthode exécutée. Par exemple : *#initialiser* ;
- *self* (cf Annexe B) désigne le récepteur du message, c'est-à-dire l'instance de l'application, modifiée par l'événement ainsi capturé.

Nous adoptons ce même principe d'instrumentation pour les métriques à événements : toute méthode désignée dans des *CIExpression* de type événement est instrumentée selon le même principe. Seule l'instruction espion diffère :

CI3Co propagerEvt: #nomMethode impactant: self.

Si l'instrumentation est ainsi clairement automatique, elle est aussi transparente : elle n'affecte que le champ compilé des méthodes et préserve leur version textuelle. C'est cette dernière qui est, en effet, visualisée lorsque l'opérateur parcourt ses méthodes ; ainsi l'instrumentation lui reste-t-elle imperceptible. En fin d'évaluation, CatchIt force la recompilation de ces versions textuelles : les instructions espion disparaîssent ainsi, et ceci toujours à l'insu du développeur. C'est le compilateur Smalltalk qui est sollicité à ces fins.

Supposons le code instrumenté : les événements sont alors propagés, à l'exécution, vers le modèle normatif. Nous en étudions les mécanismes d'observation dans la section suivante.

2.2.2. Observation

L'observation se consigne en un contexte et une liste d'actions physiques. Nous les évoquons dans cet ordre.

Contexte d'interaction

A l'exécution, les événements capturés par instrumentation sont redirigés vers *CI3Co*. Ce dernier cerne les aspects du concept affecté par l'événement et répercute l'évolution sur le modèle normatif. Cette propagation est relayée par un ensemble de proxys, sur lesquels pointe le contexte d'interaction. De façon détaillée :

- le concept applicatif concerné est mentionné dans l'argument *self* de l'instruction espion *propagerExecMethode:impactant:*. Ce concept héritant de la classe *CIEntite*, il possède potentiellement un ensemble de proxys, mémorisés au sein de son attribut *proxys* ;
- si cette liste est vide alors *CI3Co* crée les proxys idoines. Il consulte, à ces fins, les liens pAtCH de l'adaptateur et détermine les classes normatives connectées à la classe applicative concernée. Si les conditions de connexion sont satisfaites, alors des instances de ces classes normatives sont créées. Elle sont ajoutées à la liste *proxys* ;
- *CI3Co* répercute sur ces proxys les modifications survenues côté application. Pour cerner ces évolutions, il exploite le nom de la méthode passé en paramètre de l'instruction espion. L'information *evolutionMA* est alors efficace : elle répertorie les classes et attributs modifiés par chaque méthode. Il s'agit alors de calculer les valeurs correspondantes dans le repère normatif ;
- si ces valeurs sont encore non initialisées, il convient de les créer. Le typage du modèle normatif, *typageMN* ci-après décrit, est alors pertinent : il consigne, pour toute classe normative, le type des différents attributs. *CI3Co* instancie alors les classes ainsi spécifiées et affecte la valeur à l'attribut du proxy concerné ;
- la mise à jour proprement dite exploite les liens pAtCH LL : ces liens formulent une règle de correspondance exécutable entre les attributs applicatif et normatif connectés. *CI3Co* exécute ce bloc d'instructions et affecte la valeur à l'attribut étudié. Le proxy est ainsi mis à jour.

Ainsi, le contexte d'interaction est-il maintenu : c'est le proxy de l'application créée. *CI3Co* le référence par son attribut *contexte*. Consacrons-nous maintenant à l'enregistrement de la liste d'actions physiques.

Actions physiques

Dans le modèle MVC (*Model-View-Controller*) de Smalltalk, les interactions opérateur sont captées par les contrôleurs. C'est donc cette dernière classe qu'il s'agit d'instrumenter pour capturer les actions physiques opérateur. Afin de couvrir les interactions sur éléments prédéfinis et définis, nous déployons deux mécanismes de capture :

- le premier s'active sur changement de focus de la part de l'opérateur : il traque les déplacements du curseur d'un élément de présentation à un autre ;
- le second traite des sélections d'objets graphiques.

Nous les évoquons dans cet ordre.

Le premier mécanisme recourt à une classe espion *TimeProfiler* fournie par le système Smalltalk. Cette classe est dédiée à l'enregistrement de temps d'exécution sur des blocs d'instructions. L'astuce consiste ici à lui passer comme bloc d'instructions le changement de focus de l'opérateur. Ce changement est intercepté par la méthode *controlToNextLevel* des contrôleurs. Aussi, est-ce cette dernière que nous instrumentons et l'instruction sollicite l'espion.

En pratique, pour préserver le système, nous dérivons les classes *TimeProfiler* et *ApplicationModelStandardSystemController* en *CIEspion* et *CIApplicationModelStandardSystemController*. La méthode *controlToNextLevel* de cette dernière est instrumentée comme suit :

```
evaluer ifTrue: [actionOp := CIEspion view: view profile: [aView startUp].
                 view model actionOperateur: actionOp]
```

L'esprit de cette instruction est le suivant :

- elle teste, en tout premier lieu, le contexte d'interaction : le booléen *evaluer* indique si CatchIt est en phase d'évaluation. Il est positionné à *true* lorsque le développeur déclenche une évaluation. Il est rebasculé à *false* à l'issue de cette évaluation. Nous évoquerons ces aspects dans le chapitre suivant orienté logique d'utilisation de CatchIt ;
- le bloc d'instructions [*aView startUp]* est transmis à *CIEspion* : ce bloc traduit le changement de focus de la part de l'opérateur. Des informations inférées par *CIEspion*, via *TimeProfiler*, nous retenons l'instant d'occurrence de l'action, la vue d'intérêt ainsi que le contrôleur associé. Nous consignons ces informations en une instance de *CIActionPhysique*. Le champ *id* y est renseigné par l'appel de la méthode *actionOperateur:* de la classe *CIApplicationModel*, sous-classe de *ApplicationModel*. Cet appel est stipulé

dans l'instruction espion. Pour l'identification, la méthode parcourt les composants de la fenêtre cible pour identifier celui dont la vue correspond à l'attribut *vue* ;

- les mécanismes de dépendance de Smalltalk sont alors sollicités : à l'issue de cette identification, l'action physique est diffusée via un « changed » : *self changed: #actionPhysique with: uneCIActionPhysique* ;
- parmi les dépendances du *CIApplicationModel*, figure *CI3Co*. *CI3Co* réagit à l'événement selon sa méthode *update: unSymbole with: a Value*. Si *unSymbole* est *#actionPhysique*, alors *CI3Co* ajoute l'action physique à sa liste *actionsPhysiques*. Il active alors le module d'analyse par la méthode *analyserActionPhysique:*. Cette méthode fait l'objet d'une description dans la section suivante.

Avant d'évoquer l'analyse, présentons le mécanisme de capture sur des éléments de présentation « définis ». Ce mécanisme incombe aux contrôleurs : leur *controlActivity* est instrumenté pour traiter les actions souris. Ce traitement suppose que la vue maintienne une liste des objets visualisés et que tout objet connaisse ses contours. Le contrôleur sollicite alors cette liste pour identifier l'objet d'intérêt. Il le consigne dans une *CIActionPhysique* estampillée d'un instant d'occurrence. Il transmet l'information à *CI3Co*. La faiblesse réside aujourd'hui dans la gestion de cette liste d'objets. Elle impose une rigueur aujourd'hui manuelle.

Consacrons-nous maintenant à l'analyse.

2.2.3. Analyse

Le module d'analyse porte sur l'identification de l'intention opérateur. Il est hébergé par la classe *CI3Co*. Son point d'entrée est la méthode *analyserActionPhysique:* :

- cette méthode parcourt les liens PaTcH HL à la recherche de tâches opérateur prévues dans un espace de travail dont la vue coïncide avec celle identifiée dans la *CIActionPhysique* passée en paramètre ;
- grâce à l'identification de l'élément de présentation (*id* ou *objet*) de l'action physique, *CI3Co* identifie la tâche opérateur concernée :
  - en cas d'impossibilité, il traite l'action en créant une *CITacheEffective* « non identifiée » qu'il transmet au module de critique ;
  - dans tous les autres cas, il enrichit sa liste d'actions de cette nouvelle action physique et évalue l'intention opérateur sur les éventuelles modalités associées à la tâche :
    . en cas d'activation, il crée une *CITacheEffective* ; bascule l'éventuelle tâche en cours dans ses tâches suspendues et transmet ces informations au module de critique ;
    . dans tous les autres cas, *CI3Co* identifie, sur la base de l'espace de travail d'accueil, la tâche courante ou suspendue concernée par l'action. Si cette identification s'avère impossible, il applique le traitement dédié aux activations. Il enrichit sinon la liste d'actions associée à la tâche de l'action physique courante. En cas de validation ou d'annulation, il cloture la tâche ainsi identifiée. Il l'ajoute alors à la trace d'interaction ce qui déclenche un mécanisme de résolution au niveau tâche dans le module de critique.

Nous étudions ce module dans la section suivante.

2.2.4. Critique

Seules les critiques de polarité négative sont aujourd'hui traitées. Si les mécanismes permettent une extension aux homologues positives, elles sont occultées de cette description, puisque non implémentées à ce jour. Les mécanismes sont implantés dans la classe *CI3Co.*

Ces mécanismes travaillent à deux niveaux de granularité :

- au niveau macroscopique, ils considèrent la légitimité de la tâche en regard des contexte et trace d'interaction . Y interviennent la décomposition structurelle et temporelle de la tâche mère, c'est-à-dire son *CICorps*, ainsi que les préconditions des tâches et stratégies ;
- au niveau microscopique, ils vérifient la correction du corps de la tâche en termes d'actions physiques .

Si ces mécanismes sont explicitement sollicités par le module d'analyse, un mécanisme complémentaire, dédié à la complétude, travaille, à l'inverse, en continu. Il est assumé par un processus dédié, de plus faible priorité. Ce processus :

- parcourt, en permanence, les stratégies opérateur conseillées et impératives ;
- identifie les elligibles en fonction du contexte d'interaction courant ;
- et vérifie que les tâches associées sont, en pratique, déployées. Il adopte, pour ce faire, le critère suivant : la prochaine activation de tâche doit correspondre à celle attendue. Ce critère laisse, à l'opérateur, l'opportunité d'achever la tâche en cours.

Toute anomalie est consignée au sein d'une critique. La classe *CICritique* leur est consacrée. Elle dérive de *Object*. Elle est définie par les attributs suivants :

- *polarite* est un symbole indiquant la polarité de la critique : aujourd'hui négative ;
- *nature* reflète la nature de la critique : correction, concision ou complétude opératoires ;
- *tacheOuAction* est la *CITacheEffective* ou *CIActionPhysique* contestée (ou approuvée) ;
- *contexte*, *trace* et *tacheCourante* reflètent les informations contextuelles ;
- *justification* est une *CITache* ou *CIStrategie*, stipulant le comportement attendu.

Aujourd'hui, les explications se limitent à la suggestion d'une vérification manuelle de l'observabilité des préconditions et état initial des tâches transgressées : c'est un complément textuel aux justifications apportées. Les mécanismes de publication divulguent ces critiques et informations. Les politiques de diffusion sont présentées au chapitre suivant : elles relèvent de la logique d'utilisation.

3. Prise de recul

Si CatchIt est aujourd'hui implémenté en Smalltalk, rien ne s'oppose, dans les principes, à son implémentation dans un tout autre langage. Mais, plus que le langage, c'est bien l'environnement et le système qui ont ici, à l'évidence, motivé notre choix : CatchIt exploite, en effet, massivement l'absence de frontière entre l'application et le système. Il réutilise notamment :

- le parseur ;
- le compilateur ;
- les navigateurs et leurs méthodes ;
- les classes d'espionnage ;
- le modèle MVC ;
- les mécanismes de dépendances ;
- et l'éditeur de présentation.

CatchIt exploite, par ailleurs, l'absence de typage : les listes hétérogènes en témoignent.

Les principes de CatchIt restent généraux et indépendants de tout langage de programmation : la décorrélation entre principes et mise en œuvre visait à souligner cet aspect. Nous nous consacrons, dans le chapitre suivant, à une illustration de l'outil.

CHAPITRE IX : ILLUSTRATION

En support à l'illustration de CatchIt, nous nous plaçons dans un domaine d'activité phare du centre de Brest : celui de la Guerre Electronique Marine. Nous incarnons le rôle d'un développeur censé réaliser une application émanant de ce domaine. Pour une meilleure perception de CatchIt, nous adoptons une progression chronologique : nous décrivons le modèle normatif, l'application à évaluer puis l'étalonnage. Mais, notons, dès à présent, que l'objectif de cette illustration consiste à souligner l'apport de l'approche : en aucun cas, il ne s'agit de présenter l'interface de CatchIt. Cette présentation reste aujourd'hui une première maquette, dont le seul objectif consiste à éprouver les mécanismes de l'outil et en valider le principe.

CatchIt propose aujourd'hui un écran d'accueil offrant quatre services (Figure 89) :

- l'option *Normaliser* concerne les modèles normatifs. Elle permet au développeur de définir, organiser et consulter ces modèles : elle relève de la proactivité ;
- l'option *Développer* porte, à l'inverse, sur les applications : elle permet la définition de classes et méthodes applicatives ;
- l'option *Evaluer* relève typiquement de la réactivité : elle permet une connexion de l'application à un modèle normatif puis l'évaluation prédictive et expérimentale de l'interaction proposée ;
- l'option *Quitter* permet enfin de sortir de l'environnement CatchIt : elle propose, au préalable, une sauvegarde de l'image Smalltalk.

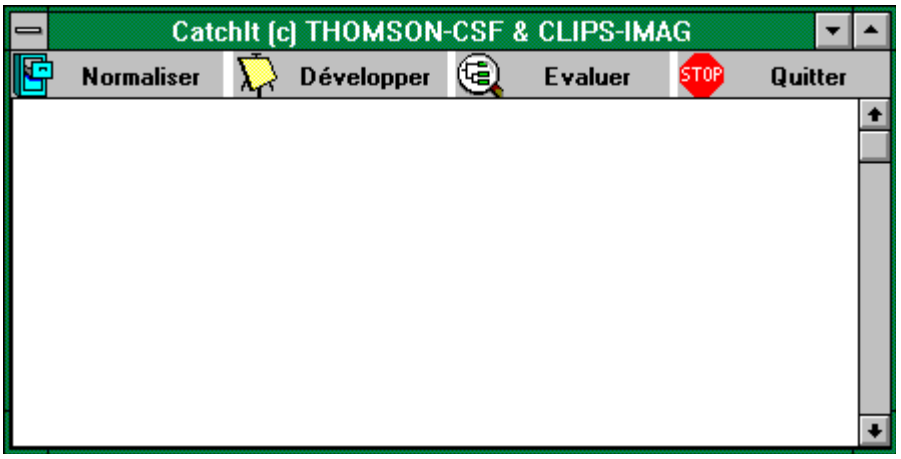

Figure 89 : Ecran d'accueil de CatchIt.

Nous nous focalisons ici sur les options Normaliser, Développer et Evaluer. Pour information, l'option Quitter ouvre une fenêtre de confirmation avec proposition de sauvegarde de l'image Smalltalk (Figure 90).

Figure 90 : Fenêtre de dialogue permettant de quitter l'application.

1. Modèle normatif

En termes de proactivité, CatchIt permet au développeur d'organiser, définir et consulter ses connaissances au sein de modèles. Le point d'entrée en est l'option *Normaliser* de l'écran d'accueil. Elle déclenche l'affichage de la fenêtre dédiée *Normaliser* (Figure 91). Cette fenêtre distingue sélection et spécification des domaines :

- la sélection porte sur la localisation du domaine et l'identification de sa version dans l'arborescence capitalisée ;
- la spécification relève de la constitution de ce domaine dans les termes de l'espace CoTaS.

Sélection et spécification sont respectivement hébergées en parties haute et basse de la fenêtre *Normaliser*. Les sections suivantes leur sont consacrées.

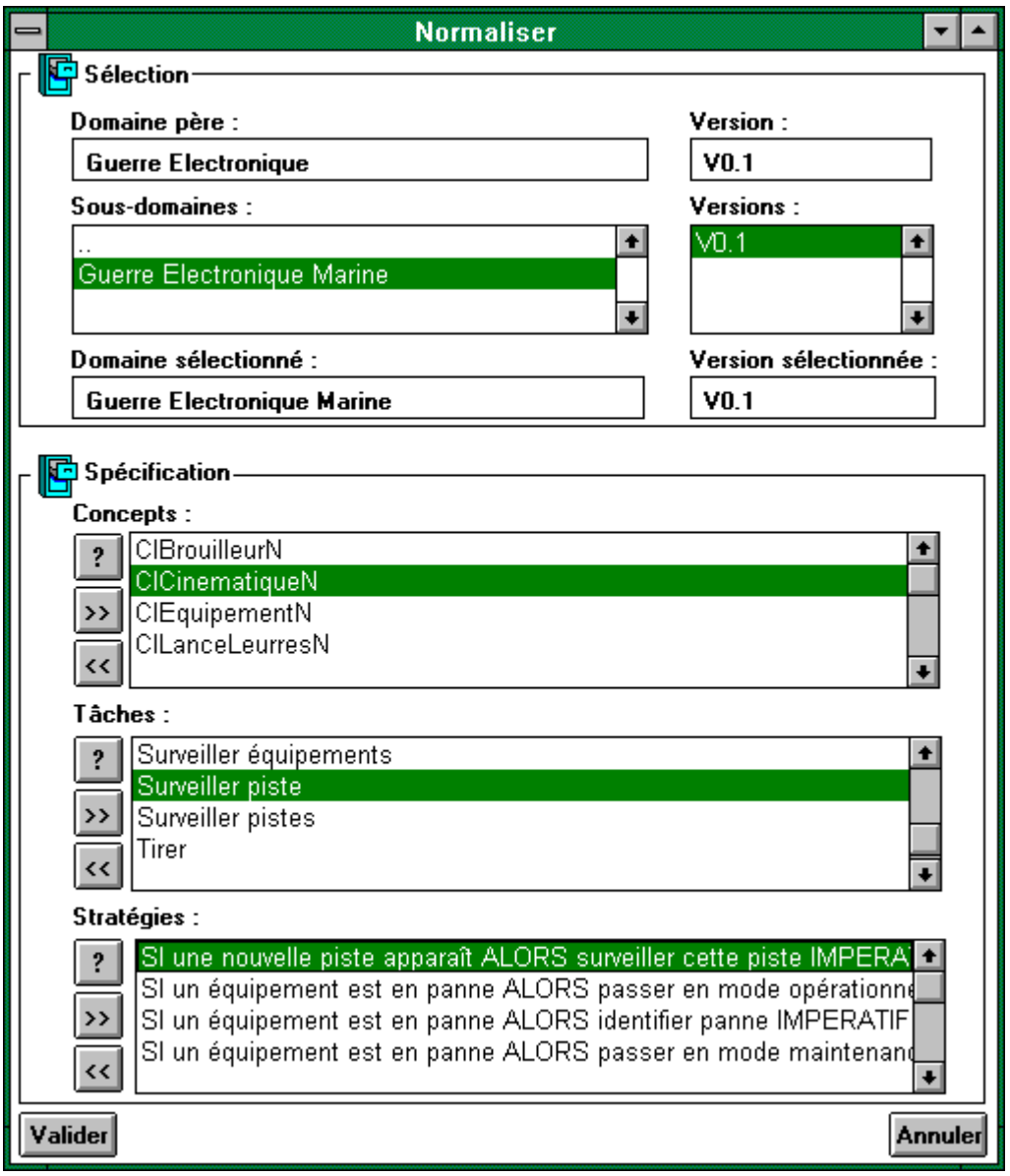

Figure 91 : Ecran dédié à la spécification des modèles normatifs pour une capitalisation des connaissances métier.

1.1. Sélection d'un domaine

Conformément à l'espace DEGRE, le choix d'un domaine s'exprime en termes de domaine et version. Il s'agit de localiser, dans l'arborescence, le domaine et la version requis. Si une création s'impose, il suffit de préciser, à la profondeur voulue, les nom et version à créer : ces spécifications s'effectuent dans les champs de sélection.

Une fois la sélection faite, les constituants CoTaS de ce choix sont répertoriés en partie basse. La section suivante est consacrée à leur spécification.

1.2. Spécification

Conformément à l'espace CoTaS, les constituants d'un domaine sont organisés en concepts, tâches et stratégies. Les éléments sont répertoriés au sein de listes. L'utilisateur peut enrichir ces listes, les inspecter ou les alléger via les boutons dédiés « >> », « ? » et « << ». Ces deux dernières fonctions supposent une sélection préalable.

Pour des raisons de confidentialité et d'expressivité, le modèle normatif restera volontairement succinct. Il portera sur le poste de l'opérateur tactique dont la mission consiste à surveiller l'évolution de la situation tactique. Cette situation s'exprime en termes de pistes. Une piste est un obstacle détecté et confirmé, au fil du temps, par corrélations temporelles et géographiques. C'est plus précisément la criticité de ces pistes qui est ici pertinente. En cas de danger, l'opérateur doit déployer des réactions justes et optimisées. Ces réactions, ou tactiques, sollicitent des équipements distants. L'état de ces matériels est, en conséquence, une information essentielle à laquelle l'opérateur doit aussi rester vigilant.

Etant donné cette description, nous illustrons, dans les trois sections suivantes, les concepts, tâches et stratégies inhérents à l'exemple.

1.2.1. Concepts

Nous décrivons ici, selon la syntaxe suivante, les classes modélisant les concepts impliqués dans l'exemple (Figure 92) :

Classe : *super-classe*
Commentaire
- *attribut*$_1$ < *type* > *commentaire*
- ...
- *attribut*$_n$ < *type* > *commentaire*

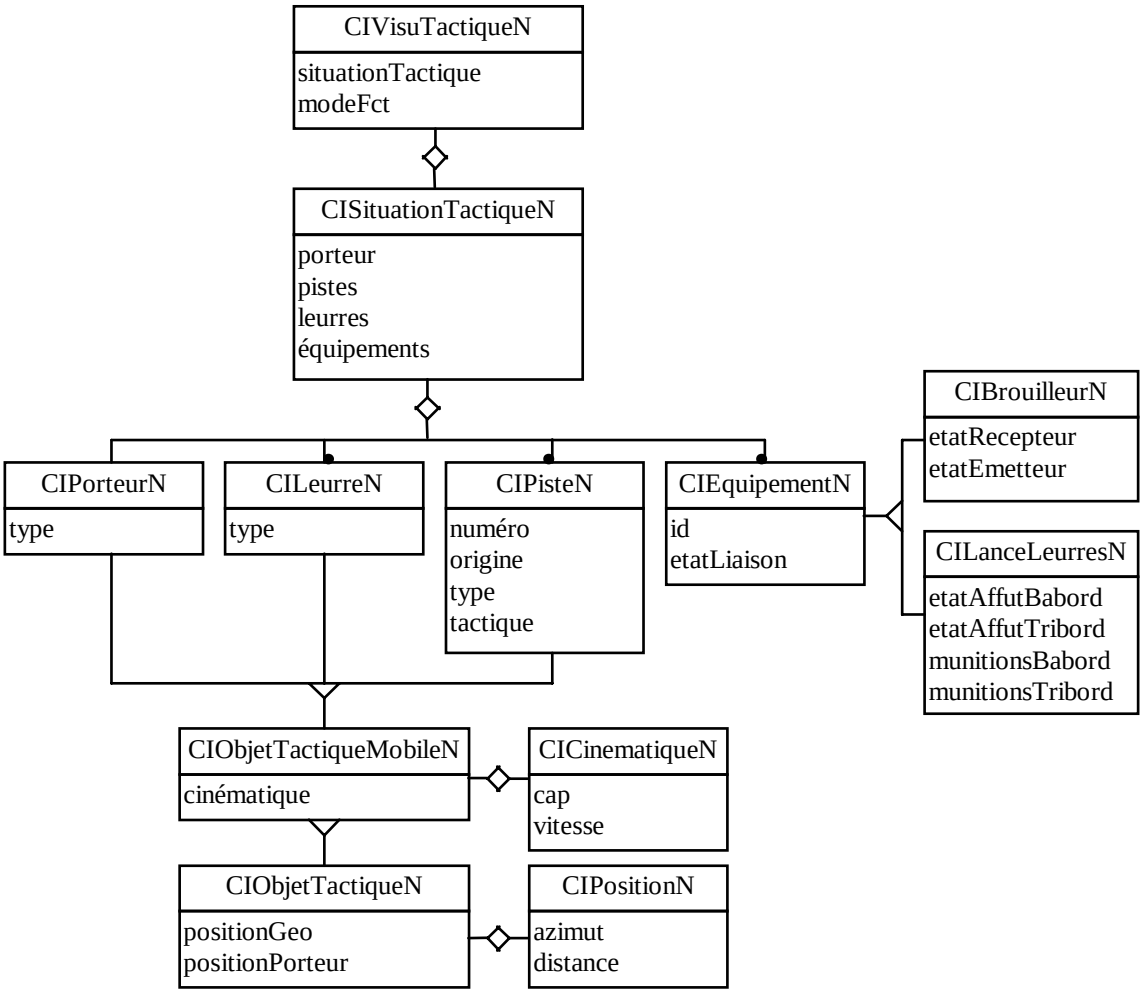

Figure 92 : Description OMT des concepts du modèle normatif, support à l'illustration.

- *CIVisuTactiqueN : Object*

Point d'entrée. Gère la situation tactique.

- *situationTactique* < *CISituationTactiqueN* >
- *modeFct* < *Symbol* > #operationnel #maintenance

- *CISituationTactiqueN* : *Object*
  - *porteur* < *CIPorteurN* >
  - *pistes* < *OrderedCollection* de *CIPisteN* >
  - *leurres* < *OrderedCollection* de *CILeurreN* >
  - *équipements* < *Collection* de *CIEquipementN* >

- *CIObjetTactiqueN* : *Object*
  - *positionGeo* < *CIPositionN* > position géographique absolue ;
  - *positionPorteur* < *CIPositionN* > position relative / au porteur ;

- *CIObjetTactiqueMobileN* : *CIObjetTactiqueN*
  - *cinematique* < *CICinematiqueN* > ;

- *CIPorteurN* : *CIObjetTactiqueMobileN*
  - *type* < *Symbol* > nature du porteur bateau, avion, etc. ;

- *CILeurreN* : *CIObjetTactiqueMobileN*
  - *type* < *Symbol* > électro-magnétique et /ou infra-rouge ;

- *CIPisteN* : *CIObjetTactiqueMobileN*
  - *numero* < *Integer* > identification de l'objet ;
  - *origine* < *String* > équipements ayant détecté la piste (B Brouilleur ; L lance leurres) ;
  - *type* < *Symbol* > nature hostile, amie ou inconnue ;
  - *tactique* <*Symbol* > éventuelle tactique en cours de déploiement sur la piste ;

- *CIPositionN* : *Object*
  - *azimut* < *Float* > exprimé en degrés ;
  - *distance* < *Float* > exprimé en Miles Nautiques (MN) ;

- *CICinematiqueN* : *Object*
  - *cap* < *Float* > exprimé en degrés;
  - *vitesse* < *Float* > exprimé en Noeuds (Nd) ;

- *CIEquipementN* : *Object*
  - *id* < *Symbol* > identificateur
  - *etatLiaison* < *Boolean* >

- *CIBrouilleurN* : *CIEquipementN*
  - *etatRecepteur* < *Boolean* >
  - *etatEmetteur* < *Boolean* >

- *CILanceLeurresN* : *CIEquipementN*
  - *etatAffutBabord* < *Boolean* >
  - *etatAffutTribord* < *Boolean* >
  - *munitionsBabord* < *Integer* >
  - *munitionsTribord* < *Integer* >.

Considérons, par exemple, la classe *CICinematiqueN*. Son inspection déclenche l'ouverture d'un browser de classe Smalltalk : ce browser donne l'accès aux attributs et méthodes de la classe (Figure 93).

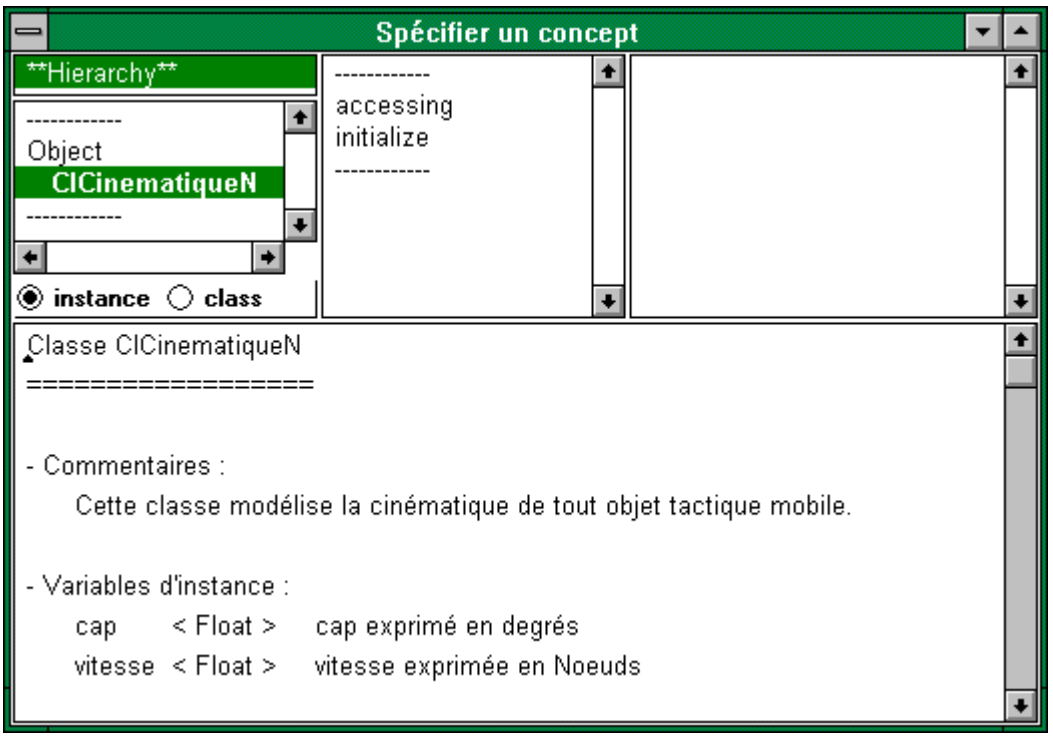

Figure 93 : Fenêtre de spécification des concepts du modèle normatif : la classe *CICinematiqueN* y est présentée.

Ces concepts étant introduits, nous nous intéressons, dans la section suivante, aux tâches du modèle normatif.

1.2.2. Tâches

Pour des raisons de concision, nous restreignons les tâches décrites au chapitre VII. Nous occultons notamment la consultation du stock de munitions et la corrélation d'informations piste. Nous numérotons, par ailleurs, les tâches afin de faciliter le suivi dans l'évaluation expérimentale.

1. *Surveiller* :
   - objectif : « Surveiller la situation tactique et les équipements connectés »
   - précondition : -
   - etat initial : *situationTactique*
   - corps : *Surveiller pistes* PAR *Surveiller équipements*
   - etatFinal : -
   - postcondition : -
   - attributs : *boucle*
   - effecteur : *opérateur*

2. *Surveiller pistes* :
   - objectif : « Surveiller les pistes »
   - précondition : -
   - etat initial : *pistes*
   - corps : *Surveiller piste*
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

3. *Surveiller piste* :
   - objectif : « Surveiller une piste »
   - précondition : -

- etat initial : *unePiste*
- corps : *Evaluer criticité* PUIS *Choisir tactique* PUIS *Déployer tactique*
- etatFinal : *unePiste*
- postcondition : *unePiste estNonHostile*
- attributs : *prioritaire, itératif*
- effecteur : *opérateur*

4. *Evaluer criticité* :
   - objectif : « Evaluer la criticité d'une piste »
   - précondition : -
   - etat initial : *unePiste*
   - corps : -
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

5. *Choisir tactique* :
   - objectif : « Choisir une tactique adaptée »
   - précondition : -
   - etat initial : -
   - corps : *Imaginer tactique* PUIS *Vérifier faisabilité*
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

6. *Imaginer tactique* :
   - objectif : « Réfléchir à une parade »
   - précondition : -
   - etat initial : -
   - corps : -
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

7. *Vérifier faisabilité* :
   - objectif : « Vérifier la faisabilité matérielle d'une tactique »
   - précondition : -
   - etat initial : -
   - corps : *Consulter état équipement*
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

8. *Consulter état équipement* :
   - objectif : « Consulter l'état de l'équipement d'intérêt »
   - précondition : -
   - etat initial : -
   - corps : -
   - etatFinal : -
   - postcondition : -
   - attributs : -

- effecteur : *opérateur*

9. *Déployer tactique* :
   - objectif : « Déployer une tactique »
   - précondition : *uneTactique estFaisable*
   - etat initial : *uneTactique*
   - corps : *Paramétrer tactique* PUIS *Tirer*
   - etatFinal : -
   - postcondition : -
   - attributs : -
   - effecteur : *opérateur*

10. *Paramétrer tactique* :
    - objectif : « Paramétrer la tactique décidée »
    - précondition : -
    - etat initial : *uneTactique*
    - corps : -
    - etatFinal : -
    - postcondition : -
    - attributs : -
    - effecteur : *opérateur*

11. *Configurer équipement* :
    - objectif : « Configurer l'équipement conformément à la tactique décidée »
    - précondition : -
    - etat initial : *uneTactique*
    - corps : -
    - etatFinal : -
    - postcondition : -
    - attributs : -
    - effecteur : *opérateur*

12. *Tirer* :
    - objectif : « Tirer la tactique paramétrée »
    - précondition : -
    - etat initial : *uneTactique*
    - corps : -
    - etatFinal : -
    - postcondition : -
    - attributs : -
    - effecteur : *opérateur*

13. *Surveiller équipements* :
    - objectif : « Surveiller les équipements connectés »
    - précondition : -
    - etat initial : *equipements*
    - corps : *Surveiller équipement*
    - etatFinal : -
    - postcondition : -
    - attributs : -
    - effecteur : *opérateur*

14. *Surveiller équipement* :
    - objectif : « Surveiller l'équipement d'intérêt »
    - précondition : -

- etat initial : *unEquipement*
- corps : -
- etatFinal : -
- postcondition : -
- attributs : -
- effecteur : *opérateur.*

Nous complétons cette énumération par des tâches, non explicites dans l'arbre des tâches, mais mentionnées dans des stratégies :

15. *Passer en mode maintenance* :
    - objectif : « Passer en mode maintenance des équipements connectés »
    - précondition : *modeFct = #operationnel*
    - etat initial : *self*
    - corps : -
    - etatFinal : *self*
    - postcondition : *modeFct = #maintenance*
    - attributs : -
    - effecteur : *opérateur*

16. *Passer en mode opérationnel* :
    - objectif : « Passer en mode opérationnel »
    - précondition : *modeFct = #maintenance*
    - etat initial : *self*
    - corps : -
    - etatFinal : *self*
    - postcondition : *modeFct = #operationnel*
    - attributs : -
    - effecteur : *opérateur*

17. *Identifier panne* :
    - objectif : « Identifier la panne de l'équipement déclaré défectueux »
    - précondition : *unEquipementEstEnPanne*
    - etat initial : *situationTactique*
    - corps : *Consulter état équipement détaillé*
    - etatFinal : -
    - postcondition : -
    - attributs : -
    - effecteur : *opérateur.*

Une fenêtre est dédiée à la spécification des tâches. Cette fenêtre s'ouvre sur activation, dans la fenêtre *Normaliser*, des boutons « ? » ou « >> » dédiés aux tâches (Figure 94). Tandis que le premier visualise la tâche sélectionnée, le second permet d'en créer une nouvelle : la fenêtre est alors vierge.

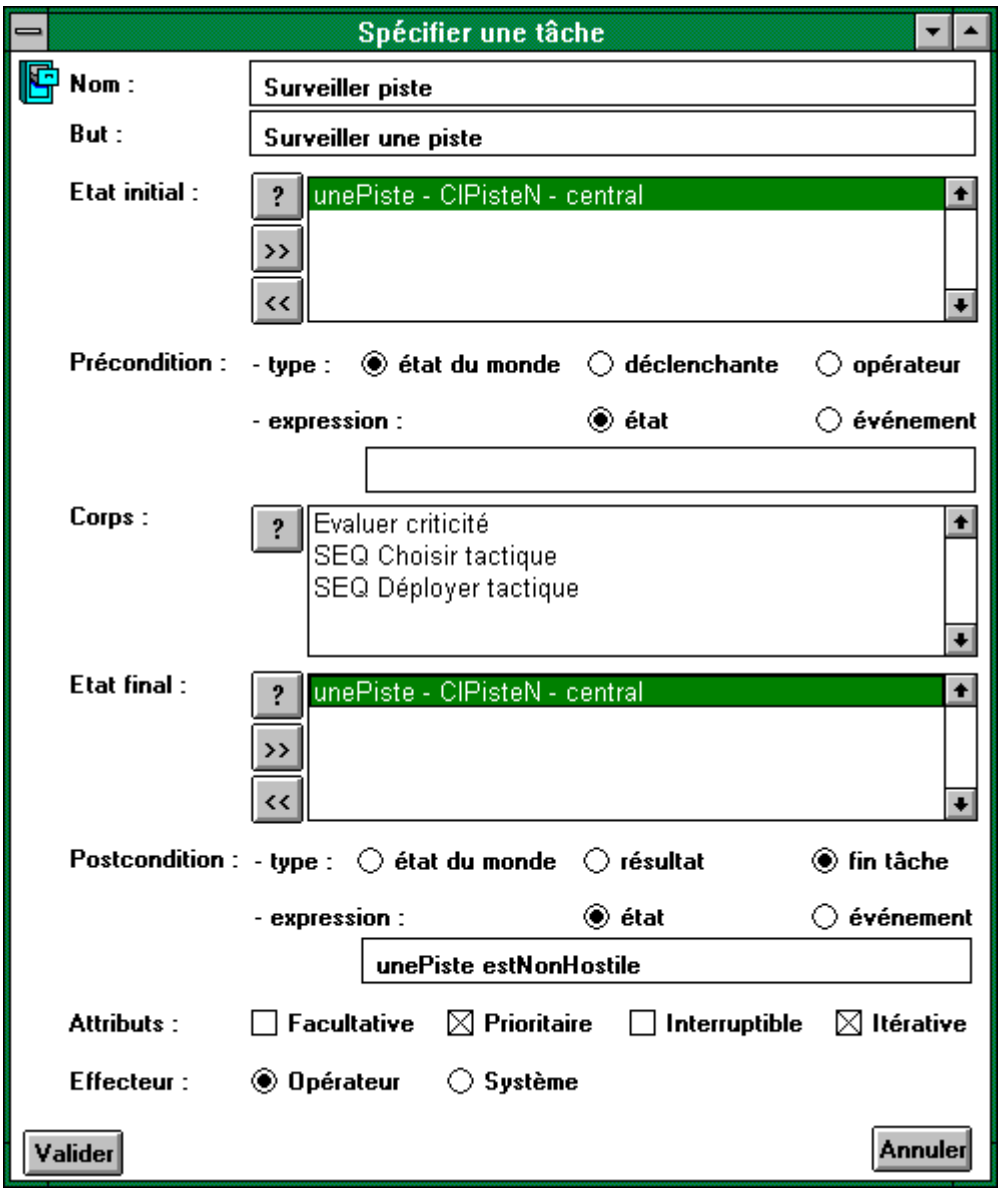

Figure 94 : Fenêtre de spécification des tâches du modèle normatif : la tâche « surveiller piste » (numéro 3 dans l'énumération) y est présentée.

Notons, dans cette fenêtre, la structuration de l'état initial en niveaux : la piste ici surveillée est un objet central, son observabilité est donc impérative. Ces poids sont définis dans une fenêtre dédiée à la spécification des états initiaux et finaux (Figure 95). Ils sont mis en œuvre par un type énuméré de valeurs : central, périphérique ou masqué (cf chapitre VII).

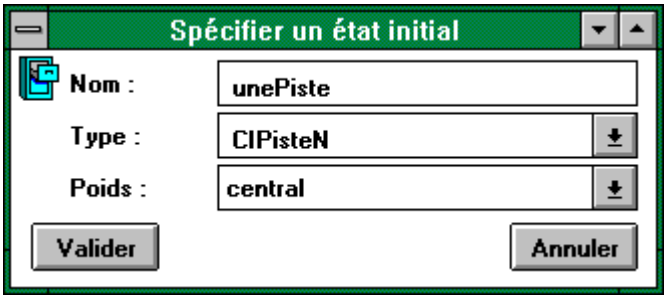

Figure 95 : Fenêtre de spécification des états initiaux et finaux : ici, une CIPisteN, centrale à la tâche.

De même, une fenêtre est consacrée à la spécification du corps de la tâche (Figure 96) : ce corps y est saisi sous forme textuelle, grammaire à l'appui. CatchIt procède à une analyse lexicale et syntaxique pour vérifier la correction de l'expression. Il la représente alors, sous la forme d'une liste, dans la fenêtre de spécification de la tâche.

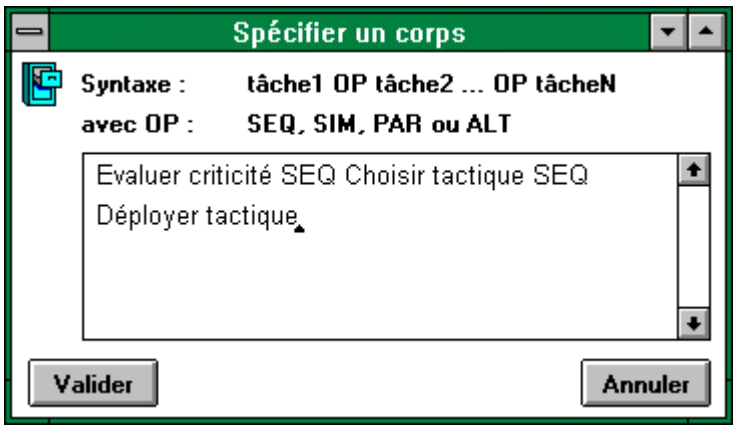

Figure 96 : Fenêtre de spécification du corps des tâches. Ici, celui de la tâche « surveiller piste » (numéro 3 dans l'énumération).

Les tâches étant ainsi spécifiées, nous nous consacrons, dans la section suivante, et suivant ce même modèle, à la définition des stratégies.

1.2.3. Stratégies

Pour une plus grande clarté, nous nous limitons à cinq stratégies :

1. SI *une nouvelle piste apparaît* ALORS *surveiller cette piste* IMPERATIF :
   - condition : *new* sur *CIPisteN*
   - action : *Surveiller piste*
   - attributs : *impératif*

2. SI *une piste est hostile* ALORS *passer en mode maintenance* INTERDIT :
   - condition : *situationTactique estMenacante*
   - action : *Passer en mode maintenance*
   - attributs : *interdit*

3. SI *un équipement est en panne* ALORS *passer en mode opérationnel* DECONSEILLE :
   - condition : *situationTactique unEquipementEstEnPanne*
   - action : *Passer en mode opérationnel*
   - attributs : *déconseillé*

4. SI *un équipement est en panne* ALORS *identifier panne* IMPERATIF :
   - condition : *situationTactique unEquipementEstEnPanne*
   - action : *Identifier panne*
   - attributs : *impératif*

5. SI *un équipement est en panne* ALORS *passer en mode maintenance* CONSEILLE :
   - condition : *situationTactique unEquipementEstEnPanne*
   - action : *Passer en mode maintenance*
   - attributs : *conseillé*

Nous présentons ci-dessous la fenêtre de spécification des stratégies (Figures 97). Elle s'ouvre sur activation, dans la fenêtre *Normaliser*, des boutons « ? » et « >> ». Dans le premier cas, la stratégie sélectionnée est

visualisée ; dans le second, la fenêtre est vierge. Nous l'illustrons sur la première ici étudiée : elle se caractérise par une précondition de type événement.

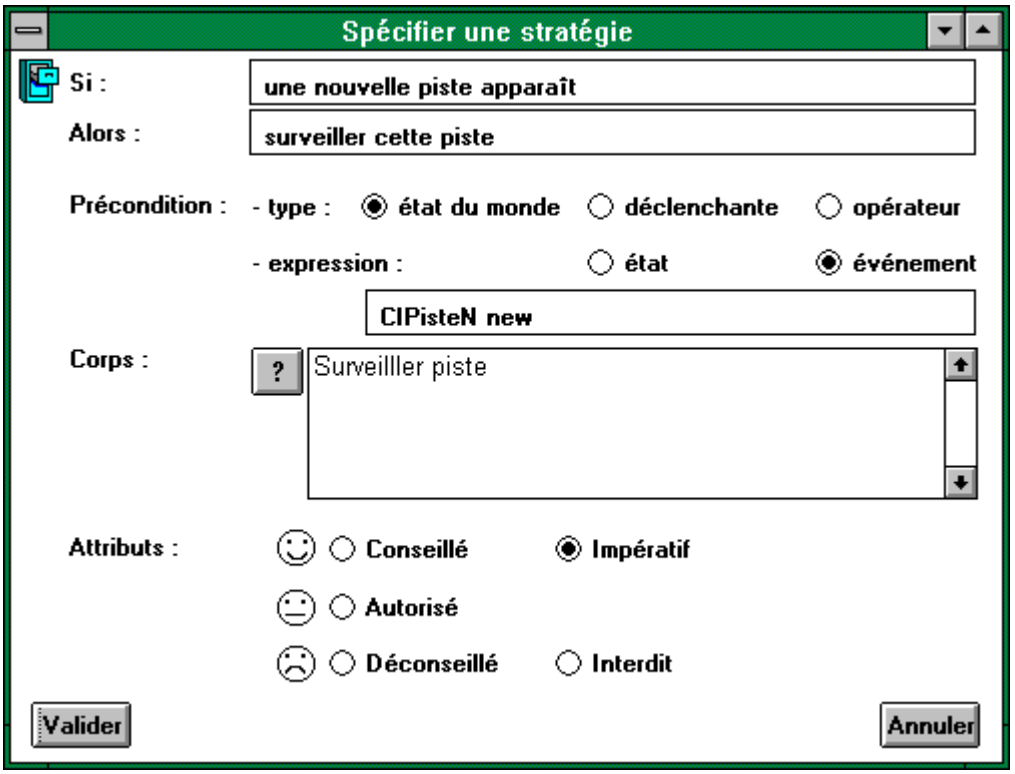

Figure 97 : Fenêtre de spécification des stratégies. Illustration sur une condition de type événement.

Le modèle normatif étant ainsi décrit, considérons une application « cas d'école » émanant de ce domaine. Elle fait l'objet d'une description dans la section suivante.

2. Modèle applicatif

Nous présentons, dans cette section, une application simpliste émanant de la guerre électronique marine. Les objets ici décrits sont spécifiés dans un browser Smalltalk ouvert grâce à l'option *Développer* de l'écran d'accueil (Figure 98).

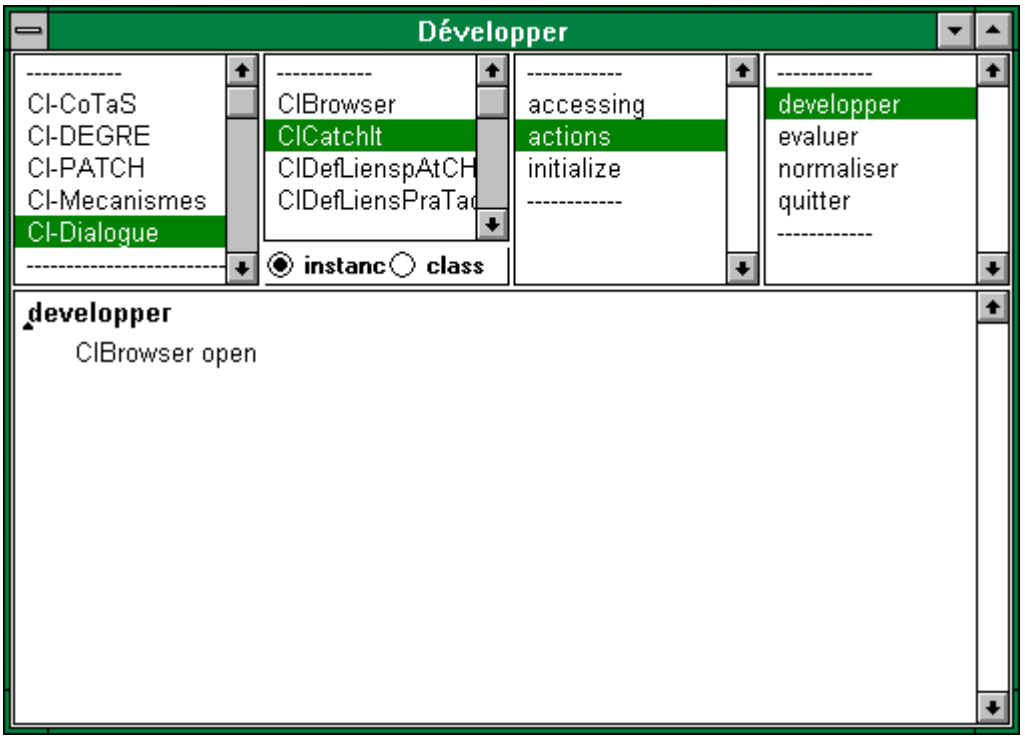

Figure 98 : Browser offert au développeur pour implémenter ses concepts.

Nous contournons ici volontairement les enseignements de la capitalisation métier pour mettre en évidence la contribution réactive de CatchIt. Ainsi l'application se veut-elle résolument non bijective par rapport à ce modèle normatif de référence. Nous présentons ici les classes concourant à la réalisation de l'application. Nous les décrivons selon la syntaxe adoptée pour la présentation des concepts du modèle normatif. Nous y distinguons les attributs selon qu'ils émanent du noyau fonctionnel ou de la présentation. La figure 99 schématise cette modélisation : elle se limite aux attributs de type abstraction.

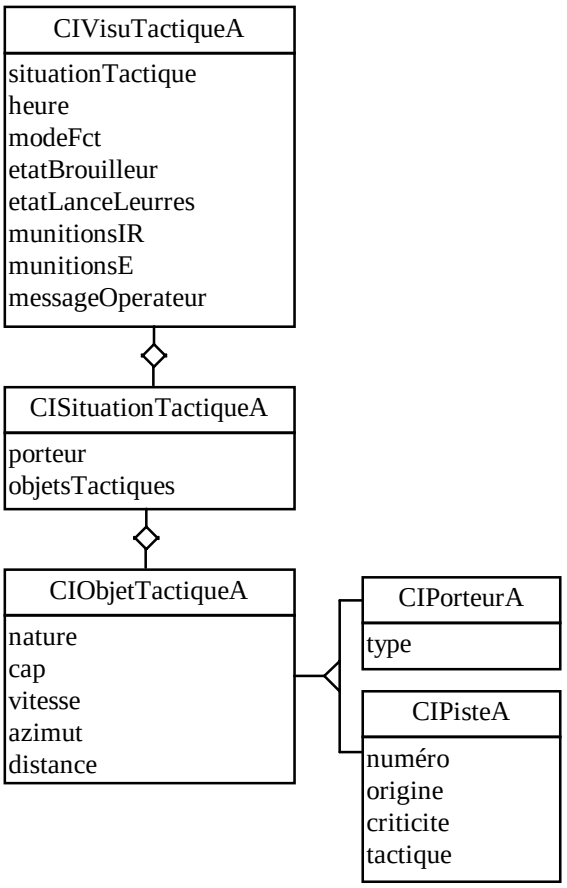

Figure 99 : Description OMT des concepts du modèle applicatif, support à l'illustration.

- *CIVisuTactiqueA : CIApplicationModel*

Point d'entrée de l'application.

→ Abstraction
– *situationTactique* < *CISituationTactiqueA* >
– *heure* < *String* >
– *modeFct* < *Symbol* > #operationnel ou #maintenance
– *etatBrouilleur* < *Integer* > code panne
– *etatLanceLeurres* < *Integer* > code panne
– *munitionsIR* < *Integer* > code munitions IR (tribord et babord respectivement en MSB et LSB)
– *munitionsE* < *Integer* > code munitions E (tribord et babord respectivement en MSB et LSB)
– *messageOperateur* < *String* >
→ Présentation
– *vueTactique* < *CIVueTactiqueA* > présentation panoramique
– *aros* < Dictionary of : *CIAro* > aires de renseignements opérateur
– *message* < *ValueHolder with : String* >
– *clavier* < *CIClavierA* >

- *CISituationTactiqueA* : *Object*

→ Abstraction
– *porteur* < *CIPorteurA*>
– *objetsTactiques* < *OrderedCollection* of : *CIObjetTactiqueA* >

- *CIObjetTactiqueA : Object*

→ Abstraction

- *nature* < Symbol > #- #E ou #IR (se justifie par la banalisation des leurres en objets tactiques)
- *cap* < *Float* > en degrés
- *vitesse* < *Float* > en KT
- *azimut* < *Float* > en degrés
- *distance* < *Float* > en NM

- *CIPorteurA : CIObjetTactiqueA*
  → Abstraction
  - *type* < *Symbol* > type de la plate-forme

- *CIPisteA* : *CIObjetTactiqueA*
  → Abstraction
  - *numero* < *Integer* > identification de l'objet ;
  - *origine* < *String* > codage des équipements ayant détecté la piste (B et L pour respectivement brouilleur et lanceleurres) ;
  - *criticite* < *Symbol* > #hostile, #nonHostile, #amie ou #inconnue ;
  - *tactique* <*Symbol* > tactique en cours de déploiement sur la piste (#aucune par défaut).

Des classes à connotation purement « présentation » participent à l'implémentation (Figure 100) :

- *CIVueTactiqueA* : *CIView*

Vue panoramique.
  - *echelle* < *MenuButton* >
  - *heure* < *ValueHolder with : String* >

- *CIAroA* : *CIApplicationModel*

Classe générique des Aires de Renseignements Opérateur (AROs).

- *CIAroBrouillageA* : *CIAroA*

Aro dédié au brouillage de pistes.

- *CIClavierA* : *CIApplicationModel*

Clavier opérateur donnant notamment l'accès aux équipements et aux pistes.

- *CIAroConfigBrouilleurA* : *CIAroA*

ARO dédiée à la configuration par défaut du brouilleur (loi de brouillage, etc.).

- *CIAroConfigLanceLeurresA* : *CIAroA*

ARO dédiée à la configuration par défaut du lance leurres (direction, etc.).

- *CIAroEtatBrouilleurA* : *CIAroA*

ARO dédiée à l'affichage d'un état détaillé du brouilleur.

- *CIAroEtatEquipementsA* : *CIAroA*

Affichage d'un état synthétique des différents équipements.

- *CIAroEtatLanceLeurresA* : *CIAroA*

ARO dédiée à l'affichage d'un état détaillé du lance leurres.

- *CIAroLeurrageA* : *CIAroA*

ARO dédiée à l'affichage d'un état détaillé du lance leurres.

- *CIAroModeFctA* : *CIApplicationModel*

ARO dédiée à l'affichage d'un état détaillé du brouilleur.

- *CIAroNavigationA* : *CIAroA*

ARO de navigation (cap et vitesse porteur).

- *CIAroPistesA* : *CIAroA*

Table synthétique des pistes.

- *CIAroPistesA* : *CIAroA*

ARO dédiée aux informations et traitements inhérents à une piste (affichage, suppression, fin tactique).

Nous illustrons cette réalisation par la capture d'écran suivante (Figure 100). Elle met en évidence :

- la zone panoramique (*CIVueTactiqueA*) reflétant notamment :
  - l'échelle ;
  - l'heure ;
  - et les marqueurs distance (cercles concentriques). Ils matérialisent les distances pistes - porteur ; la cartographie étant centrée sur ce dernier ;
- la zone dédiée au mode de fonctionnement et notamment à sa commutation opérationnel / maintenance (*CIAroModeFctA*) ;
- les informations synthétiques quant à l'état des différents équipements (*CIAroEtatEquipementsA*). Notons l'accès possible à des données plus détaillées via les boutons « ? ». Les AROs *CIAroEtatBrouilleurA* et *CIAroEtatLanceLeurresA* s'affichent alors en lieu et place de l'ARO courant. Ici *CIAroEtatLanceLeurresA ;*
- le canvas d'accueil des différents AROs. Ici les informations détaillées sur l'état du lance leurres ;
- le clavier (*CIClavierA*) permettant
  - le déploiement de tactiques (*CIAroBrouillageA* et *CIAroLeurrageA*) ;
  - l'accès à des informations détaillées sur la piste en contrôle direct (CD) (*CIAroInfosPisteA)* ;
  - la suppression de la piste (*CIAroInfosPisteA*) ;
  - la configuration par défaut des équipements connectés (*CIAroConfigBrouilleurA* et *CIAroConfigLanceLeurresA*) ;
  - et une touche de réserve.

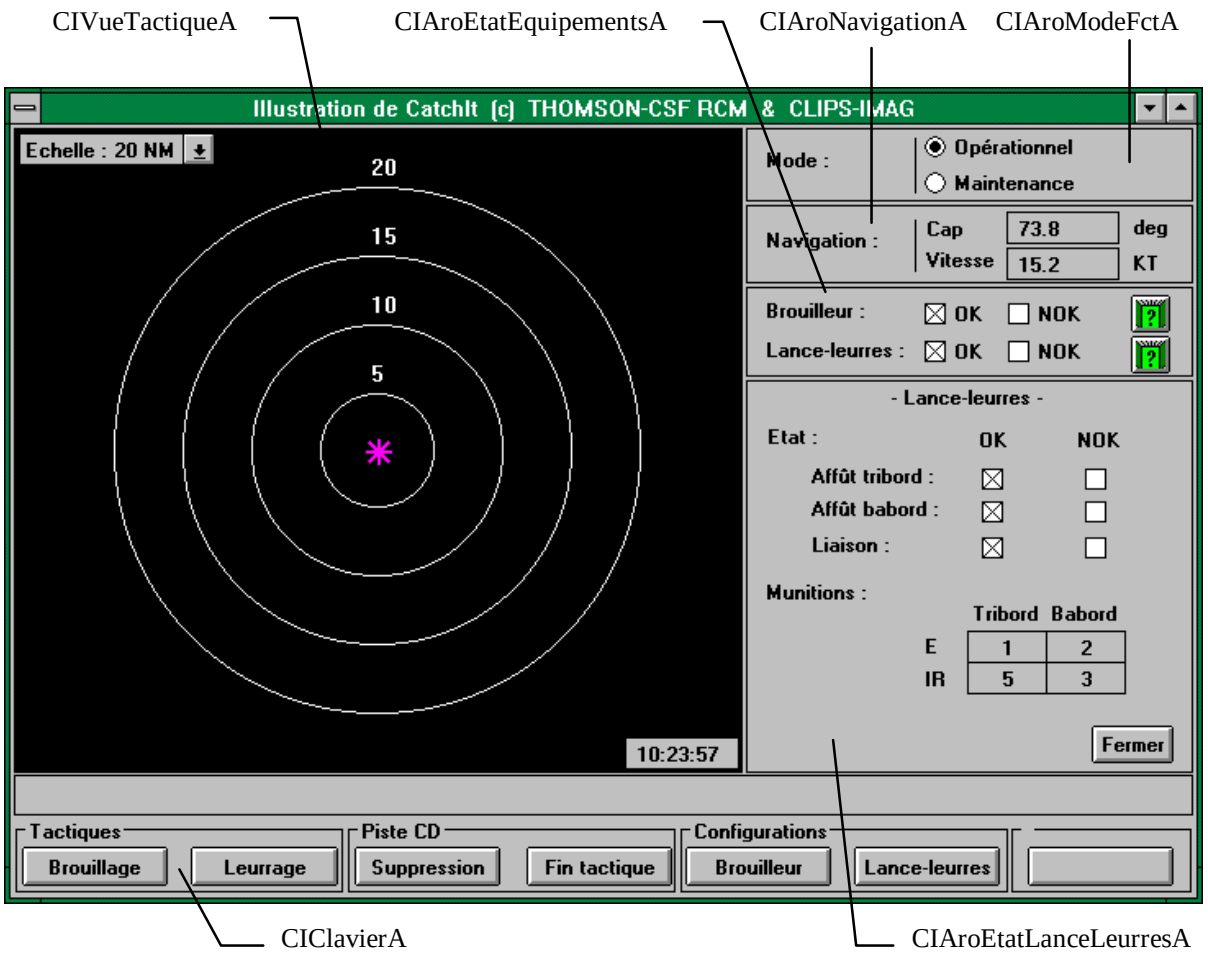

Figure 100 : Application « Cas d'école » en support à l'illustration de CatchIt.

L'application étant ainsi décrite, consacrons-nous à la présentation de la contribution réactive de CatchIt : c'est l'objet de la section suivante.

3. Etalonnage

La facette réactive de CatchIt est accessible via le bouton « Evaluer » de l'écran d'accueil. Son activation déclenche l'affichage de la fenêtre 101 :

- les modèles applicatif et normatif y sont identifiés. Les boutons « ? » permettent de les spécifier ;
- un espace est consacré à la connexion déclarative entre ces modèles : il s'agit de projeter les concepts, tâches et stratégies du modèle normatif en abstractions et présentations au sein de l'application ;
- enfin, une configuration de l'évaluation permet de préciser, selon la modalité prédictive ou expérimentale, les centres d'intérêt de l'évaluateur.

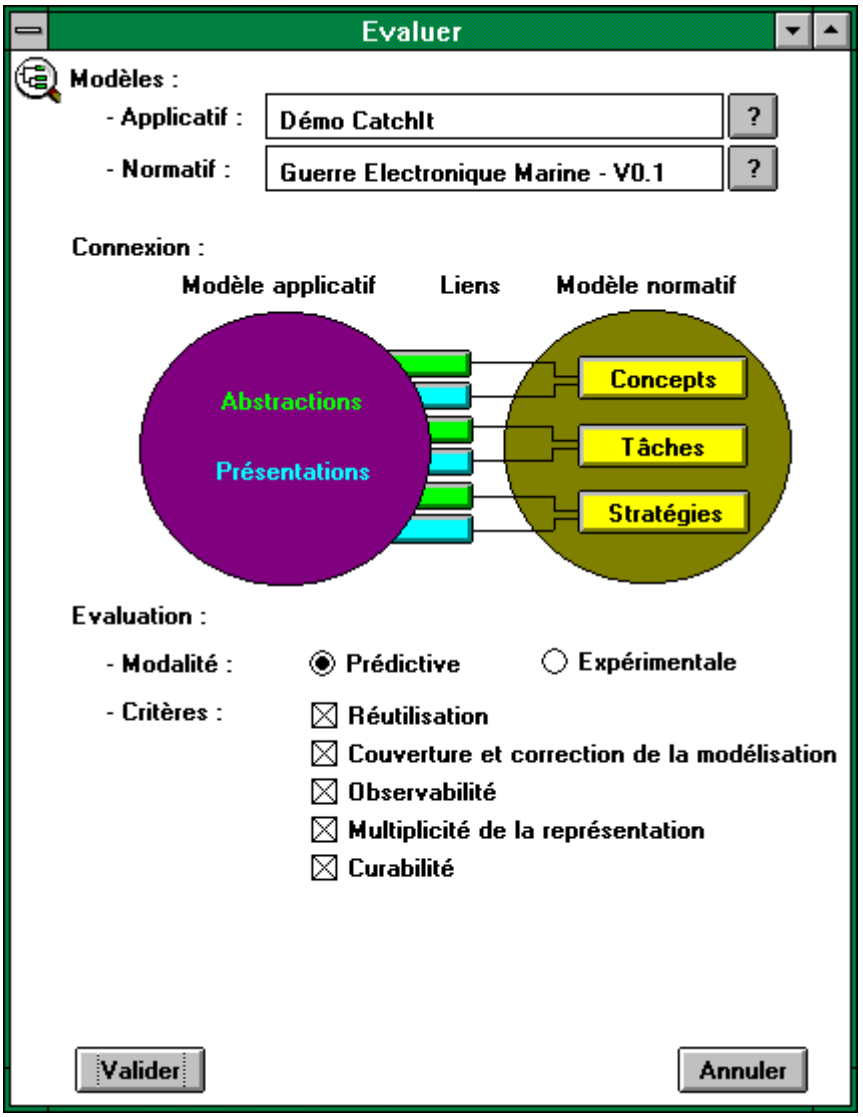

Figure 101 : Fenêtre dédiée à l'évaluation de l'application. Les critères offerts sont ceux disponibles en version prédictive.

La figure 102 présente les critères d'évaluation disponibles en version expérimentale.

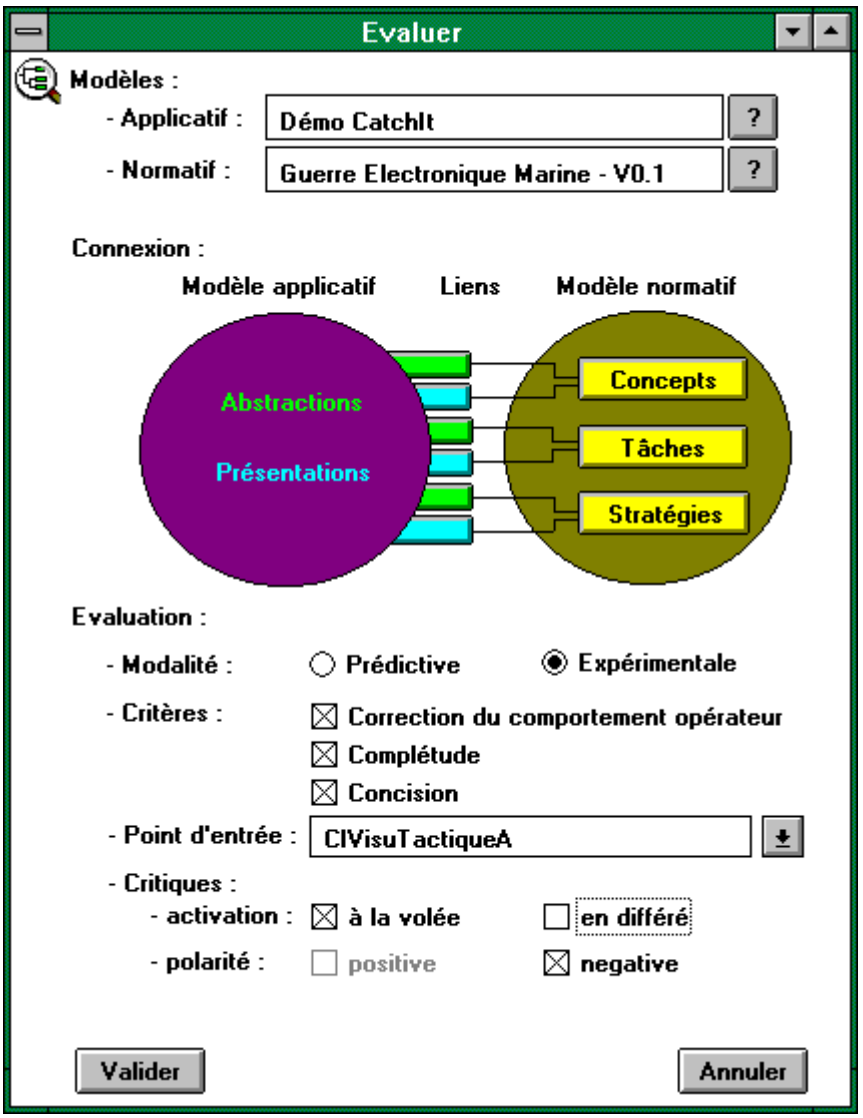

Figure 102 : Fenêtre dédiée à l'évaluation de l'application. Les critères offerts sont ceux disponibles en version expérimentale.

Mais, pour exploiter ces prestations réactives, il faut, au préalable, avoir circonscrit l'application et établi la connexion entre modèles : nous en étudions la mise en œuvre dans les sections suivantes.

3.1. Périmètre applicatif

Circonscrire l'application signifie délimiter les classes contribuant à sa modélisation. Rappelons que cette information est nécessaire à la mesure de la couverture de la modélisation.

Une fenêtre est dédiée à cet usage. Elle est accessible via le bouton « ? » associé à l'identification du modèle applicatif, situé dans la fenêtre Evaluer (Figure 103).

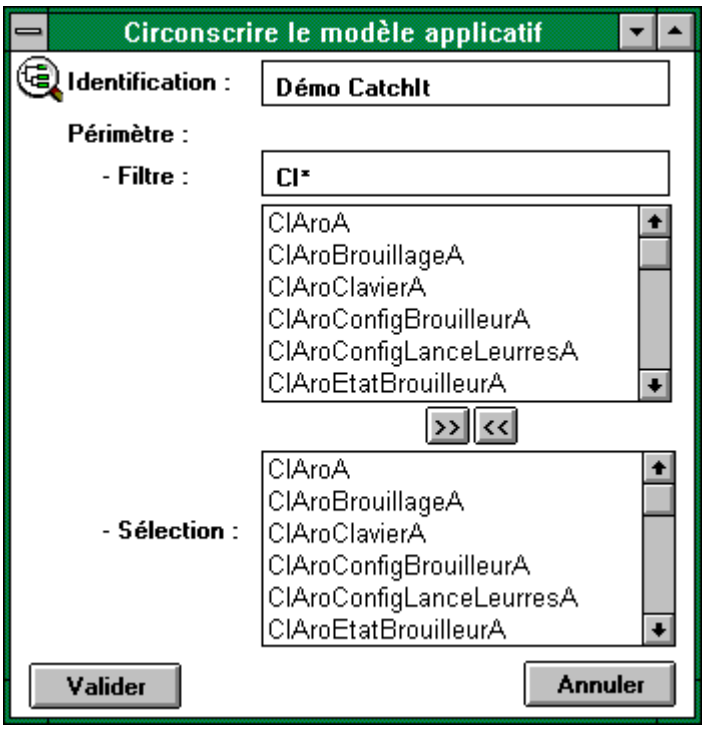

Figure 103 : Fenêtre dédiée à la circonscription de l'application.

Ce périmètre étant délimité, la connexion entre modèles applicatif et normatif peut être établie.

3.2. Connexion

Il s'agit ici d'associer abstractions et présentations aux concepts, tâches et stratégies du modèle normatif. Nous les illustrons dans cet ordre.

3.2.1. Liens pAtCH

Les liens sont définis aux niveaux HL puis LL. Pour notre exemple :

Entité A	**Entité N**	**Condition**	**Règle**
CIVisuTactiqueA	**CIVisuTactiqueN**	-	
situationTactique	situationTactique	-	
etatBrouilleur	situationTactique	-	situationTactique brouilleur etatRecepteur: ((etatBrouilleur bitAt: 1)=0). situationTactique brouilleur etatEmetteur: ((etatBrouilleur bitAt: 2)=0).
etatLanceLeurres	situationTactique	-	situationTactique lanceLeurres etatAffutBabord: ((etatLanceLeurres bitAt: 1)=0). situationTactique lanceLeurres etatAffutTribord: ((etatLanceLeurres bitAt: 2)=0).
munitionsIR	situationTactique	-	situationTactique lanceLeurres munitionsBabord: ((munitionsIR bitAnd: 16r0F) + (munitionsE bitAnd: 16r0F)). situationTactique lanceLeurres

			munitionsTribord: ((munitionsIR bitAnd: 16rF0) bitShift: -4) + (munitionsE bitAnd: 16rF0) bitShift: -4))
munitionsE	situationTactique	-	situationTactique lanceLeurres munitionsBabord: ((munitionsIR bitAnd: 16r0F) + (munitionsE bitAnd: 16r0F)). situationTactique lanceLeurres munitionsTribord: ((munitionsIR bitAnd: 16rF0) bitShift: -4) + (munitionsE bitAnd: 16rF0) bitShift: -4))
modeFct	modeFct	-	-
heure	-		
messageOperateur	-		
vueTactique	-		
aros	-		
message	-		
clavier	-		
CISituationTactiqueA	**CISituationTactiqueN**	-	
porteur	porteur		
objetsTactiques	pistes	nature = #-	-
objetsTactiques	leurres	nature=#E or : [nature=#IR]	-
-	equipements		
CIObjetTactiqueA	**CIObjetTactiqueMobileN**		
nature	-		
cap	cinematique	-	cinematique cap: cap
vitesse	cinematique	-	cinematique vitesse: vitesse
azimut	positionGeo	-	positionGeo azimut: azimut
distance	positionGeo	-	positionGeo distance: distance
-	positionPorteur		
CIObjetTactiqueA	**CILeurreN**	nature=#E or : [nature=#IR]	
nature	type	-	-
CIPorteurA	**CIPorteurN**	-	
type	type	-	-
CIPisteA	**CIPisteN**		
numero	numero	-	-
origine	origine	-	-
criticite	type	-	-
tactique	tactique	-	-

Figure 104 : Exemples de liens pAtCH HL et LL.

Notons la présence de classes isolées :

- dans l'application : CIAroA, CIAroBrouillageA, CIClavierA, CIAroConfigBrouilleurA, CIAroConfigLanceLeurresA, CIAroEtatBrouilleurA, CIAroEtatEquipementsA, CIAroEtatLanceLeurresA, CIAroLeurrageA, CIAroModeFctA, CIAroNavigationA et CIAroPistesA ;
- dans le modèle normatif : CIObjetTactiqueN, CIPositionN, CICinematiqueN, CIEquipementN, CIBrouilleurN et CILanceLeurresN.

Nous présentons Figure 105 la fenêtre dédiée à la spécification des liens pAtCH. Cette fenêtre s'ouvre sur activation, dans la fenêtre *Evaluer*, du bouton connectant les abstractions applicatives aux concepts normatifs. Elle répertorie, en partie haute, la liste des liens HL établis. La partie basse en offre l'accès. L'exemple porte sur la connexion entre les classes *CIObjetTactiqueA* et *CIObjetTactiqueMobileN*.

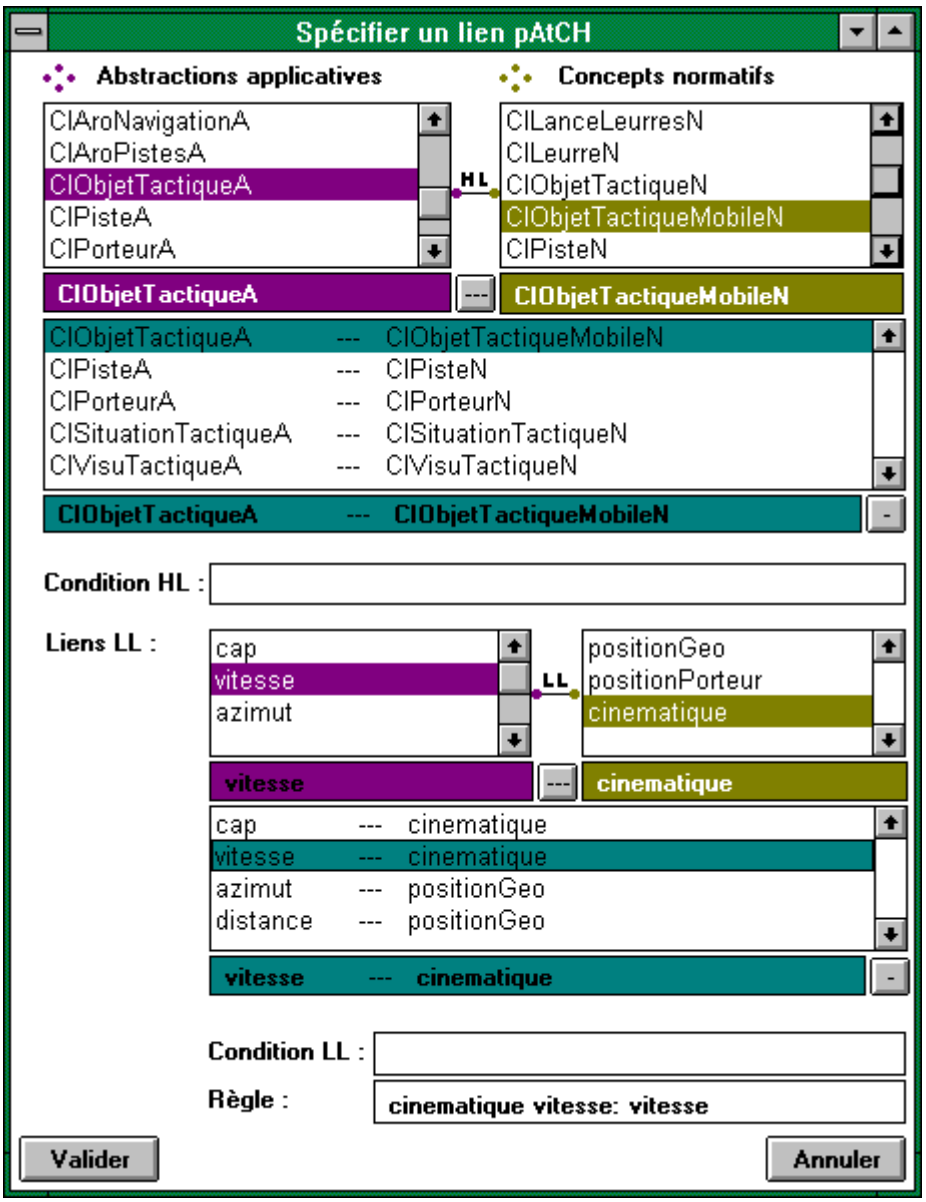

Figure 105 : Fenêtre de spécification des liens pAtCH.

Nous nous consacrons, dans la section suivante, à la spécification des liens PatCH.

3.2.2. Liens PatCH

Les liens mis en œuvre dans l'exemple sont les suivants (Figure 106) :

Entité A	**Entité N**	**Condition**
CIAroNavigationA	**CIPorteurN**	-
widget cap	cinematique	
widget vitesse	cinematique	
CIVueTactiqueA	**CIPorteurN**	-
objetTactique (non traité)		
CIVueTactiqueA	**CIPisteN**	-
objetTactique (non traité)		
CIAroPisteA	**CIPisteN**	-
widget azimut	positionGeo	

widget distance	positionGeo	
widget cap	cinematique	
widget vitesse	cinematique	
widget numero	numero	
widget origine	origine	
widget criticite	criticite	
widget tactiqueEnCours	tactique	
CIAroPistesA	**CIPisteN**	-
(non détaillé)		
CIVueTactiqueA	**CILeurreN**	-
objetTactique (non traité)		
CIAroEtatEquipementsA	**CIBrouilleurN**	-
brouilleurOK	etatRecepteur	
brouilleurOK	etatEmetteur	
brouilleurOK	etatLiaison	
brouilleurNOK	etatRecepteur	
brouilleurNOK	etatEmetteur	
brouilleurNOK	etatLiaison	
CIAroEtatBrouilleurA	**CIBrouilleurN**	-
liaisonOK	etatLiaison	
liaisonNOK	etatLiaison	
recepteurOK	etatRecepteur	
recepteurNOK	etatRecepteur	
émetteurOK	etatEmetteur	
émetteurNOK	etatEmetteur	
CIAroEtatEquipementsA	**CILanceLeurresN**	-
lanceLeurresOK	etatAffutBabord	
lanceLeurresOK	etatAffutTribord	
lanceLeurresOK	etatLiaison	
lanceLeurresNOK	etatAffutBabord	
lanceLeurresNOK	etatAffutTribord	
lanceLeurresNOK	etatLiaison	
CIAroEtatLanceLeurresA	**CILanceLeurresN**	-
liaisonOK	etatLiaison	
liaisonNOK	etatLiaison	
affutBabordOK	etatAffutBabord	
affutBabordNOK	etatAffutBabord	
affutTribordOK	etatAffutTribord	
affutTribordNOK	etatAffutTribord	
leurresIRBabord	munitionsBabord	
leurresEBabord	munitionsBabord	
leurresIRTribord	munitionsTribord	
leurresETribord	munitionsTribord	

Figure 106 : Exemples de liens PatCH.

Une fenêtre est dédiée à la spécification de ces liens PatCH. Cette fenêtre s'ouvre sur activation, dans la fenêtre *Evaluer*, du bouton connectant les présentations applicatives aux concepts normatifs. Nous la présentons Figure 107 sur l'exemple de la connexion entre l'espace de travail *CIAroEtatLanceLeurresA* et le concept *CILanceLeurresN*.

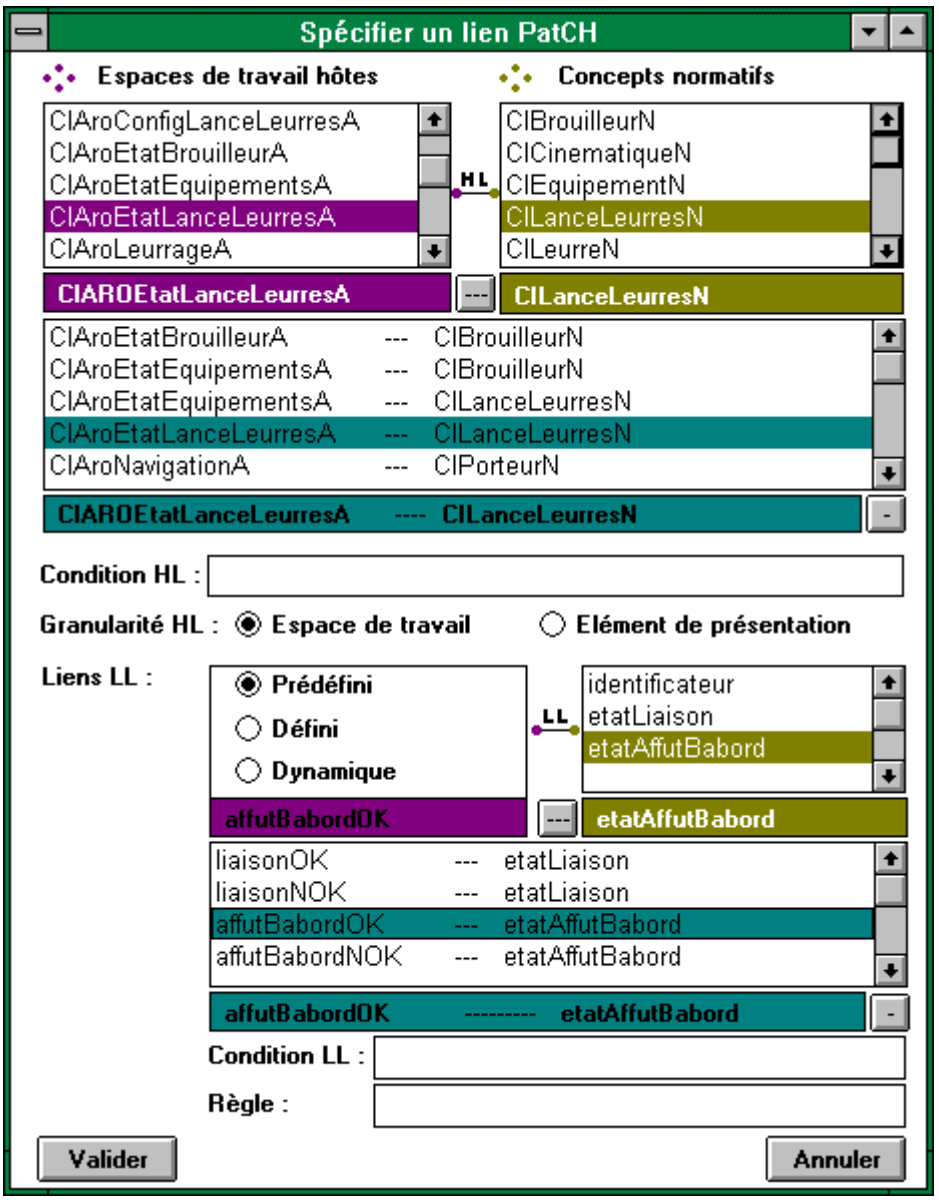

Figure 107 : Fenêtre de spécification des liens PatCH.

Nous nous consacrons, dans la section suivante, à l'illustration du dernier type de liens : les liens PaTcH.

3.2.3. Liens PaTcH

Les liens sont définis aux niveaux HL et LL. Pour notre exemple :

Présentations	**Tâches**	**Condition**
CIVisuTactiqueA	Surveiller	-
CIVisuTactiqueA	Surveiller pistes	-
CIAroPistesA	Surveiller pistes	-
CIVisuTactiqueA	Surveiller piste	
CIAroPistesA	Surveiller piste	
CIAroPisteA – activation : raccourci souris	Surveiller piste	numero value = unePiste numero
CIAroPisteA – activation : raccourci souris	Evaluer criticité	
-	Traiter piste	
-	Choisir tactique	

-	Imaginer tactique	
-	Vérifier faisabilité	
CIAroEtatEquipementsA	Consulter état équipements	
CIAroConfigBrouilleurA – activation : espace CIClavierA ; widget #configBrouilleur – validation : widget #valider – annulation : widget #fermer	Configurer équipement	unEquipement estUnBrouilleur
CIAroConfigLanceLeurresA – activation : espace CIClavierA ; widget #configLanceLeurres – validation : widget #valider – annulation : widget #fermer	Configurer équipement	unEquipement estUnLanceLeurres
CIAroBrouillageA – activation : espace CIClavierA ; widget #brouillage – validation : widget #valider – annulation : widget #fermer	Déployer tactique	unEquipement estUnBrouilleur
CIAroLeurrageA – activation : espace CIClavierA ; widget #leurrage – validation : widget #valider – annulation : widget #fermer	Déployer tactique	unEquipement estUnLanceLeurres
CIAroBrouillageA – widgets : *confidentiel*	Paramétrer tactique	uneTactique estUnBrouillage
CIAroLeurrageA – widgets : *confidentiel*	Paramétrer tactique	uneTactique estUnLeurrage
CIAroLeurrageA - widget #tir	Tirer	uneTactique estUnBrouillage
CIAroLeurrageA - widget : #tir	Tirer	uneTactique estUnLeurrage
CIAroEtatEquipementsA	Surveiller équipements	
CIAroEtatBrouilleurA – activation : espace CIAroEtatEquipementsA; widget #infosBrouilleur – validation : widget #valider – annulation : widget #fermer	Surveiller équipement	unEquipement estUnBrouilleur
CIAroEtatLanceLeurresA – activation : espace CIAroEtatEquipementsA; widget #infosLanceLeurres – validation : widget #valider – annulation : widget #fermer	Surveiller équipement	unEquipement estUnLanceLeurres
CIAroEtatBrouilleurA – activation : espace CIAroEtatEquipementsA; widget #infosBrouilleur – validation : widget #valider – annulation : widget #fermer	Identifier panne	unEquipement estUnBrouilleur
CIAroEtatLanceLeurresA – activation : espace CIAroEtatEquipementsA; widget #infosLanceLeurres	Identifier panne	unEquipement estUnLanceLeurres

– validation : widget #valider – annulation : widget #fermer		
CIAroModeFctA – widget #modeFct	Passer en mode maintenance	mode value = #maintenance
CIAroModeFctA – widget #modeFct	Passer en mode opérationnel	mode value = #operationnel

Figure 108 : Exemples de liens PaTcH tissés entre les modèles applicatif et normatif.

Nous présentons Figures 109 et 110 la fenêtre dédiée à la spécification des liens PaTcH. Cette fenêtre s'ouvre sur activation, dans la fenêtre *Evaluer*, du bouton connectant les présentations applicatives aux tâches normatives. Dans la figure 109, la tâche « passer en mode maintenance » est une tâche élémentaire dont la procédure se limite à une action physique. Dans la figure 110, la tâche est assignée à un espace de travail (*CIAroEtatLanceLeurresA*), modalités d'activation, validation et annulation à l'appui. La fenêtre 111 illustre la fenêtre dédiée à la spécification de ces modalités. Elle s'ouvre sur activation, dans la fenêtre 110, du bouton « ? » associé à la liste des modalités. Remarquons qu'aujourd'hui la connexion PaTcH ne s'effectue qu'au niveau HL.

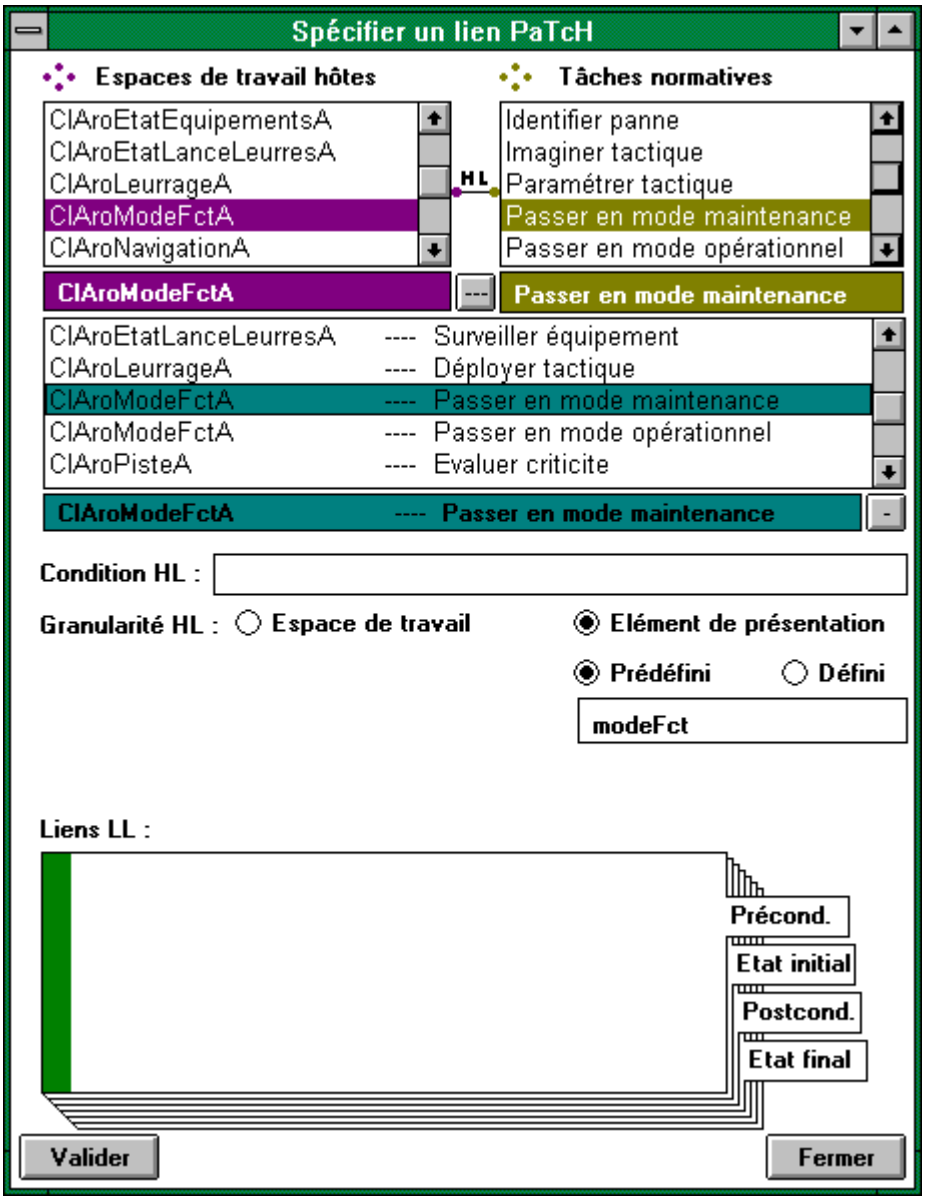

Figure 109 : Fenêtre de spécification des liens PaTcH.

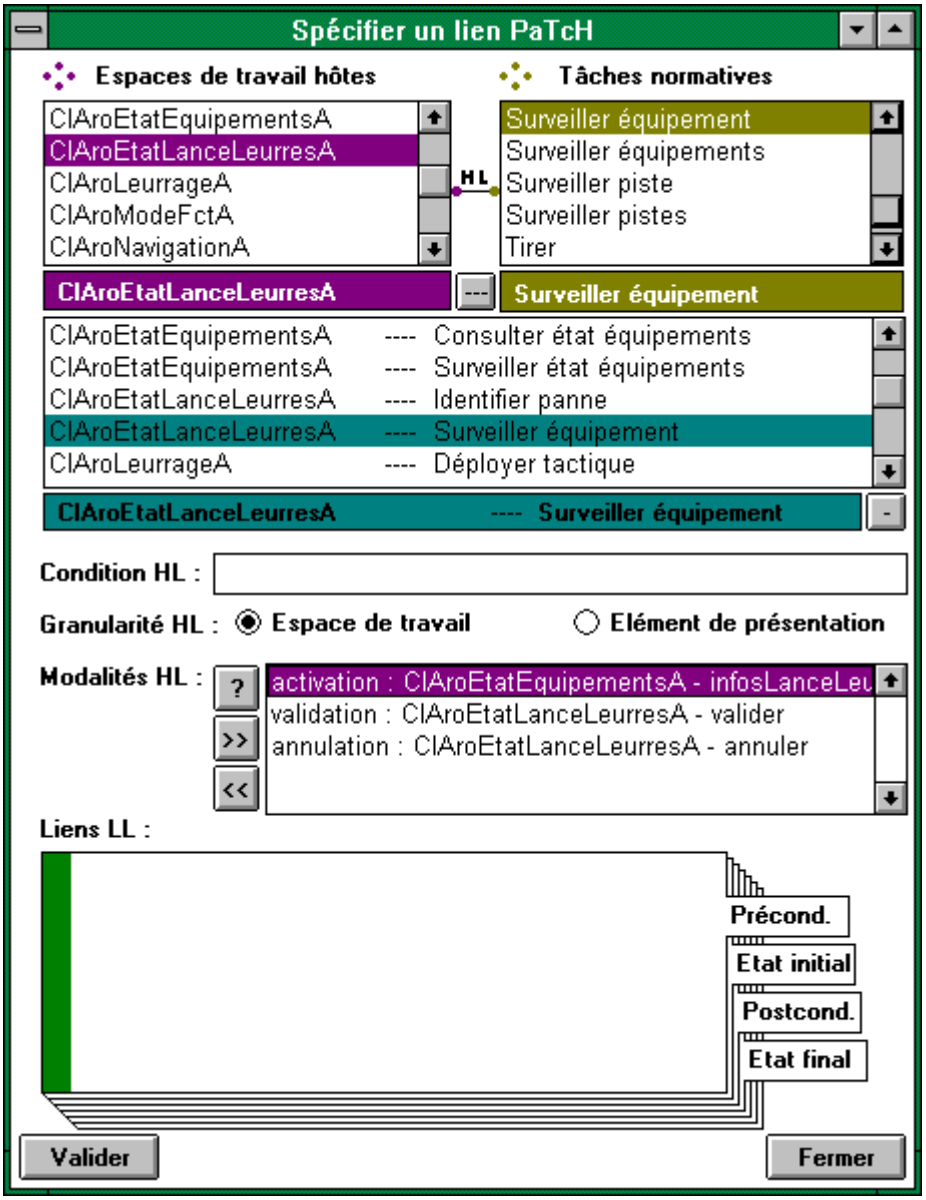

Figure 110 : Fenêtre de spécification des liens PaTcH.

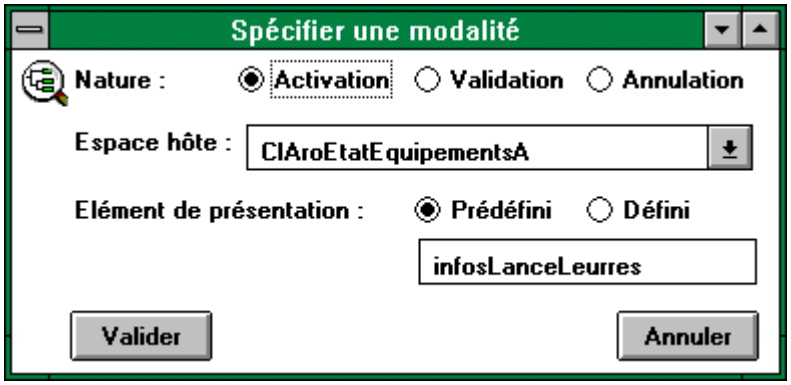

Figure 111 : Fenêtre de spécification des modalités.

Ainsi s'achèvent la description de la connexion et son illustration, via des écrans niveau maquettes. Les liens étant ainsi tissés, nous nous consacrons à l'évaluation de l'interaction. Nous commençons par l'évaluation prédictive.

3.3. Evaluation prédictive

Nous présentons Figure 112 la fenêtre dédiée à la publication des critiques prédictives établies. Cette fenêtre n'est aujourd'hui qu'un maquettage, non encore connecté aux mécanismes de CatchIt. Elle s'ouvre sur activation du bouton *Valider* dans la fenêtre *Evaluer*.

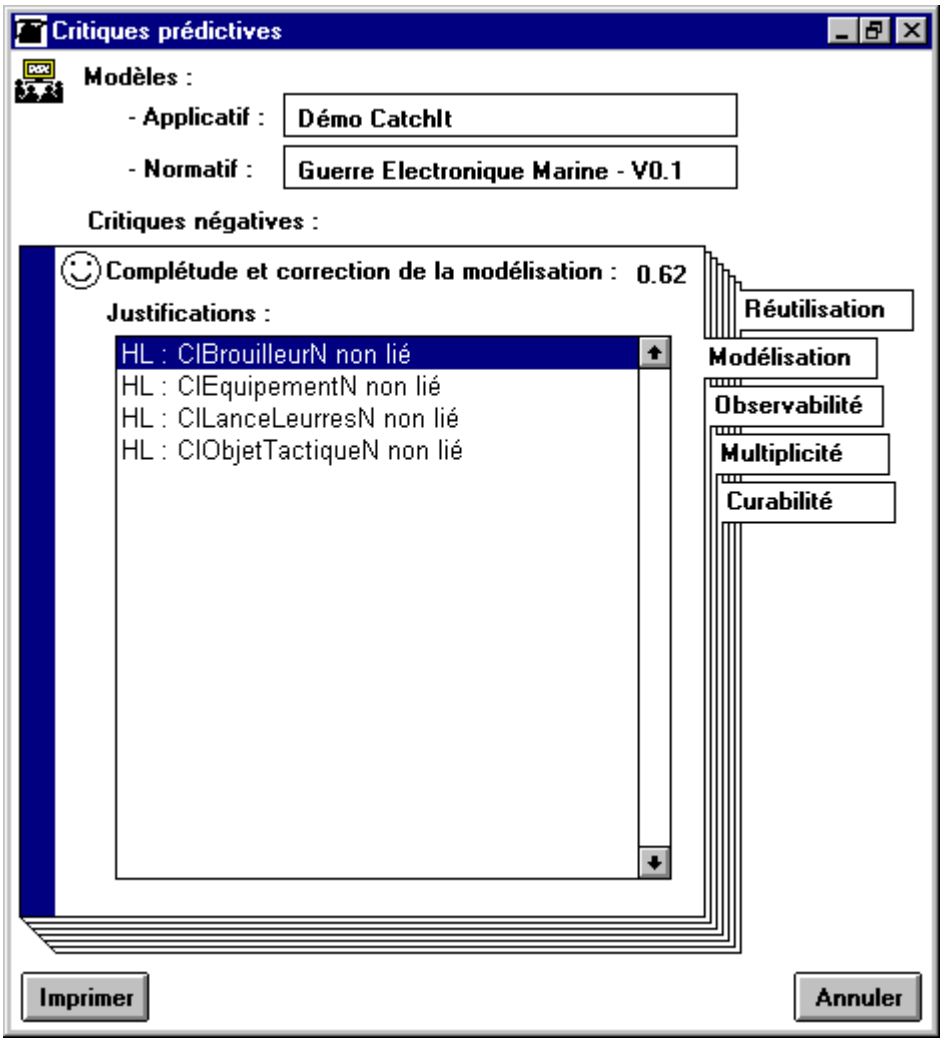

Figure 112 : Fenêtre de publication, en différé, des résultats de l'évaluation prédictive.

Nous nous consacrons, dans la section suivante, à l'évaluation expérimentale.

3.4. Evaluation expérimentale

En version expérimentale, l'activation du bouton *Valider* de la fenêtre *Evaluer* déclenche :

- l'instrumentation du code (le booléen *evaluer* mentionné dans la mise en œuvre est alors basculé à *true*) ;
- et le lancement de l'application « Démo CatchIt ».

Nous nous plaçons, conformément à la configuration de l'évaluation, dans le cadre d'une polarité négative et d'une publication « à la volée ». Nous proposons, sur cette base, dans les sections suivantes, la détection de critiques de type « correction », « complétude » et « concision » opératoires.

3.4.1. Correction opératoire

Supposons le contexte d'interaction suivant :

- l'application est en mode opérationnel ;
- la situation tactique comporte une piste hostile.

Supposons que, malgré ce contexte, l'opérateur active le widget de changement de mode de fonctionnement dans l'ARO *CIAroModeFctA*. Alors :

- le module d'observation de CatchIt capture cette action physique ;
- au regard des liens PaTcH, il identifie deux tâches impliquant cette action : *Passer en mode opérationnel* et *Passer en mode maintenance* ;
- la précondition de la première tâche (numéro 16 dans l'énumération) n'est pas satisfaite (ici modeFct = #operationnel). Donc, si telle est l'intention opérateur, il convient de s'assurer de l'observabilité de cette précondition ;
- la précondition de la seconde tâche (numéro 15 dans l'énumération) est, en revanche, satisfaite. Mais, elle transgresse une stratégie (la 2 dans l'énumération) relative à cette même tâche (ici une piste est hostile). Il convient donc de s'assurer de l'observabilité de l'hostilité de cette piste.

La fenêtre dédiée à la publication, à la volée, des critiques expérimentales consigne ces inférences (Figure 113). De façon générale, le point d'interrogation dans cette fenêtre donne accès aux informations contextuelles relatives à la critique. C'est aujourd'hui un inspecteur Smalltalk.

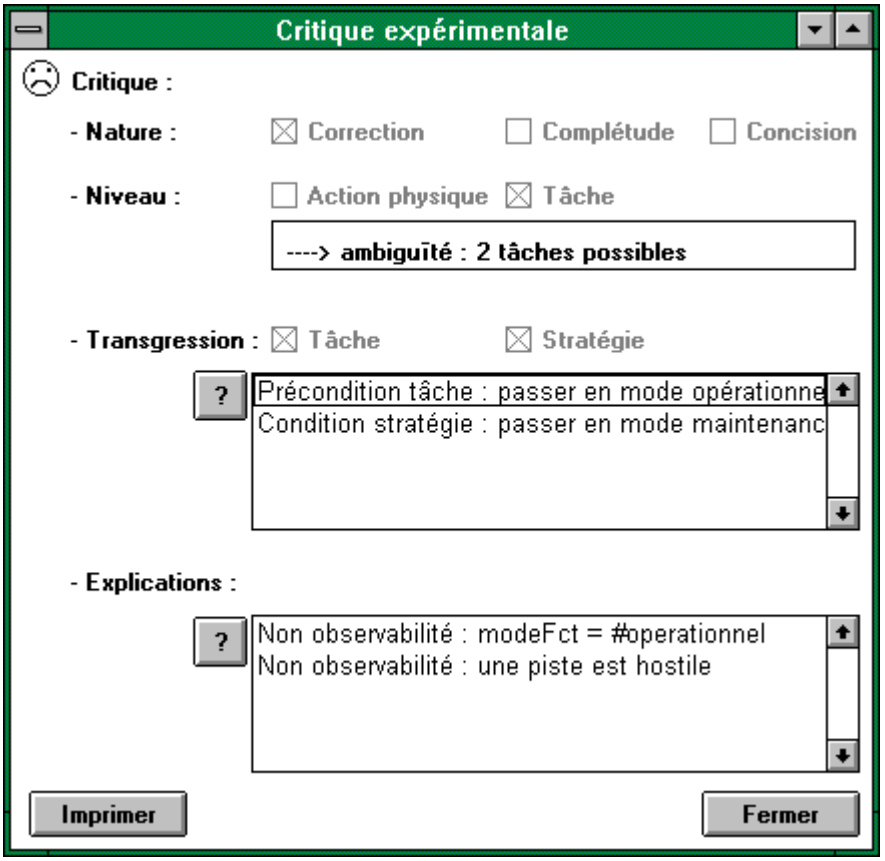

Figure 113 : Fenêtre de publication, à la volée, des critiques expérimentales. Elle est ici illustrée sur une incorrection opératoire.

La section suivante traite d'anomalies de type complétude.

3.4.2. Complétude opératoire

Supposons le contexte d'interaction suivant :

- l'application est en mode opérationnel ;
- la liaison brouilleur est défectueuse.

Supposons que l'opérateur, malgré ce contexte, tente une configuration de l'équipement. Alors CatchIt intervient. Il rappelle les deux stratégies suivantes et s'interroge quant à l'observabilité de leurs préconditions :

- SI *un équipement est en panne* ALORS *identifier panne* IMPERATIF
- SI *un équipement est en panne* ALORS *passer en mode maintenance* CONSEILLE.

La figure 114 présente cette publication.

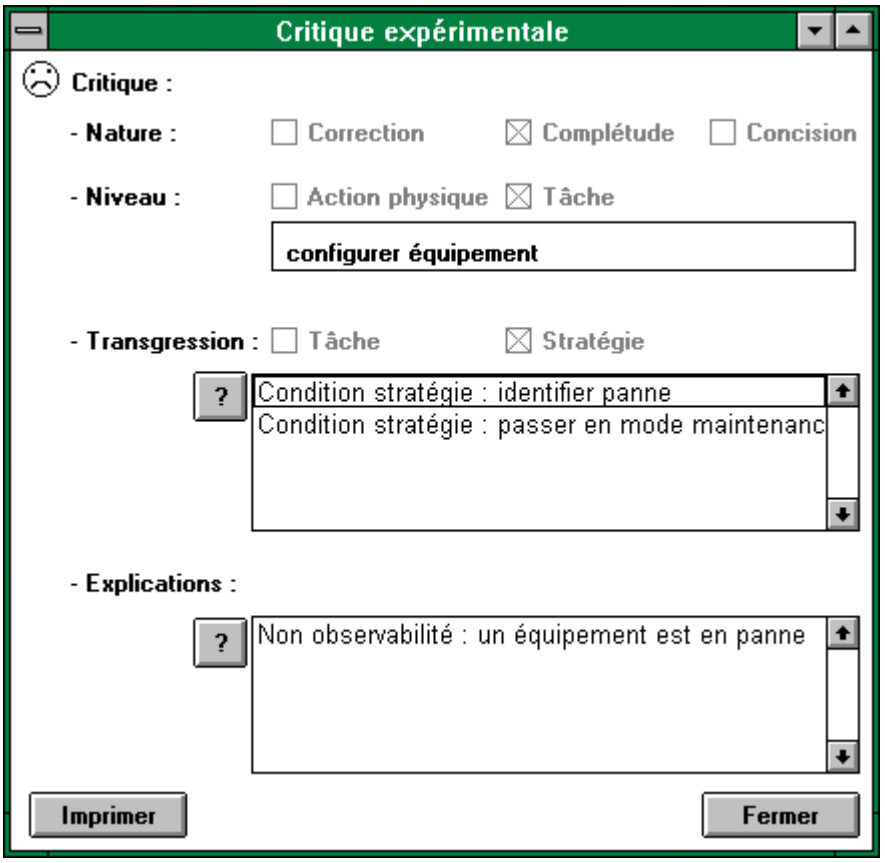

Figure 114 : Fenêtre de publication, à la volée, des critiques expérimentales. Elle est ici illustrée sur une incomplétude opératoire.

La section suivante traite de la concision opératoire.

3.4.3. Concision opératoire

Supposons que le lance-leurres soit opérationnel mais que l'opérateur en demande un état détaillé. CatchIt infère alors l'intention opérateur en termes de tâches, à savoir : surveiller équipement. La précondition de cette tâche est certes satisfaite mais aucune stratégie n'en impose la mise en œuvre dans ce contexte. CatchIt consigne ces réflexions en une critique (Figure 115).

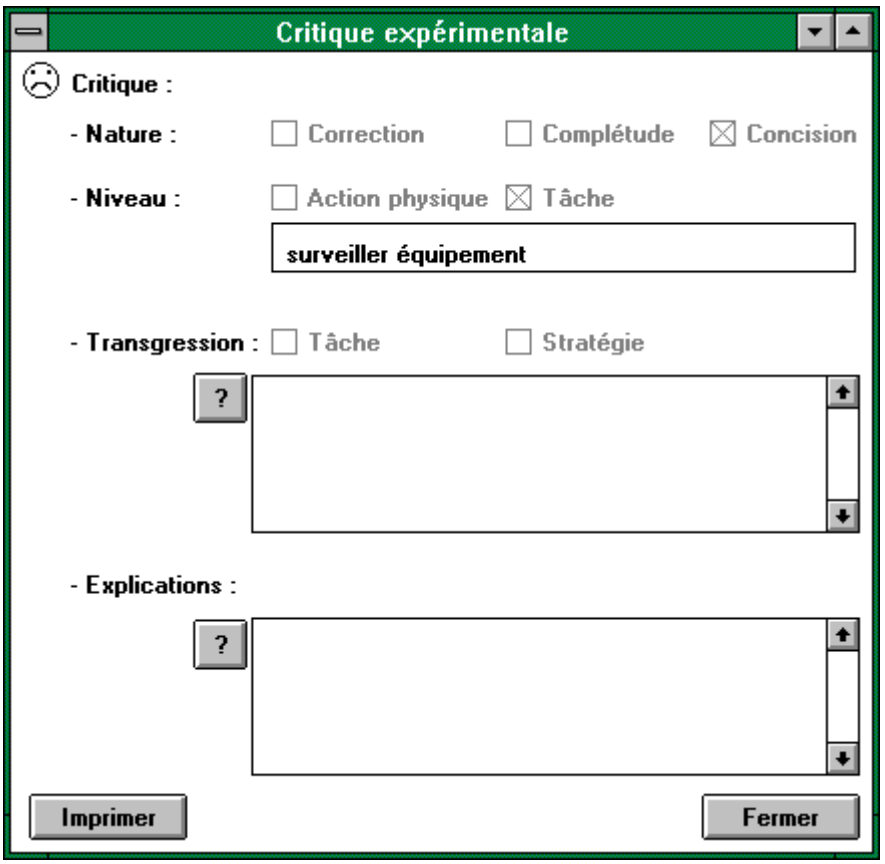

Figure 115 : Fenêtre de publication, à la volée, des critiques expérimentales. Elle est ici illustrée sur une absence de concision opératoire.

Ainsi s'achève l'illustration de CatchIt, et plus généralement le présentation de l'outil. Nous en proposons, en conclusion, une analyse critique.

CONCLUSION

Motivés par une concurrence de plus en plus pressante, des marchés de moins en moins protégés, nous avons étudié, dans cette thèse, les méthodes et outils désormais nécessaires pour affronter une conjoncture économique défavorable. Après un état de l'art, dans la partie I, de l'ingénierie des IHM en industrie, nous nous sommes focalisés, dans la partie II, sur l'offre académique. Nous avons alors dressé, de cette confrontation, des axes de progrès et fils directeurs exploités en partie III. Nous proposons ici, en conclusion, un résumé de la contribution puis une analyse critique de l'approche. Cette prise de recul met en exergue les forces et faiblesses de l'outil.

1. Résumé de la contribution

Cette thèse est résolument ancrée dans le contexte industriel. Elle procède, en tout premier lieu, à une analyse critique de la pratique industrielle en matière de conception et d'évaluation d'IHM. Elle souligne l'insuffisance des outils commercialisés et, en conséquence, une pratique où le savoir-faire prédomine.

En matière d'outils, l'offre s'apparente à une mosaïque décousue et incomplète d'outils dédiés. Ce paysage « accidenté » renforce encore le caractère intrinsèquement discontinu du processus de développement. Déjà favorisée par une ampleur souvent multisite voire multi-industrielle des projets, cette discontinuité est antinomique de facteurs qualité, dont la traçabilité. Si les espoirs se tournent alors vers la sacro-sainte capitalisation métier, cette capitalisation reste aujourd'hui artisanale et épisodique. Pourtant, l'actuelle tendance à la spécialisation par domaine milite en sa faveur. Elle permettrait :

- de pérenniser un savoir-faire désormais stratégique ;
- de s'imposer, de façon crédible et systématique, en phase de spécification système ;
- de pallier l'imperfection des spécifications ;
- de rehausser les applications de leur sémantique originelle ;
- et de déployer des évaluations opérationnelles, trop souvent restreintes aux seules considérations fonctionnelles.

Malgré ces atouts, la capitalisation métier reste aujourd'hui soumise au bon vouloir et à la sensibilité de chacun : aucune structure d'accueil ne l'intègre de façon centrale. Nous postulons ici en son potentiel « proactif » et « réactif», deux termes que nous introduisons pour respectivement refléter :

- une saine intention de bien faire ;
- et un contrôle, a posteriori, de l'atteinte de l'objectif commun.

Si, dans le principe, proactivité et réactivité agissent en complémentarité, un tour d'horizon des approches académiques révèle, en réalité, une assignation systématique :

- les outils de nature proactive ne couvrent pas l'évaluation de l'interaction ;
- inversement, les outils réactifs s'apparentent à des greffons totalement étrangers à l'application.

C'est l'espace de classification ESPRIT (Examination Space for Proactive and Reactive Innovative Tools) qui a dirigé cette étude comparative. La caractérisation des outils académiques au sein de cet espace a clairement révélé l'assignation des approches envisagées. CatchIt rompt avec ce paysage dichotomique en adoptant une structure d'accueil résolument proactive et réactive. Centrée sur les connaissances métier, cette structure favorise la complémentarité en hébergeant :

- les capitalisation et réutilisation logicielles ;
- les évaluations prédictive et expérimentale de l'interaction proposée.

CatchIt déploie, pour ce faire, plusieurs espaces de réflexion :

- CoTaS (Concepts, Tasks, Strategies) propose une structuration tripartite des connaissances métier en concepts, tâches et stratégies. Ces dernières, résolument anthropocentrées, formalisent le savoir-faire opérateur quasiment immuable et si contextuel ;
- DEGRE (Domain Encapsulation by Genericity Refinement and Evolutivity) traite de la pérennisation des connaissances. Il identifie trois axes discriminants : la généricité des informations, leur évolution en termes de version et enfin leur niveau d'affinement au regard du processus de développement ;

- PATCH (Presentation and Abstraction for Task and Concept Hooks) incarne toute la philosophie de l'approche. Il sous-tend la connexion déclarative entre l'application à tester et sa référence métier. C'est, en effet, sur une ausculation externe de l'application que CatchIt fonde sa polyvalence proactive et réactive. Il préserve le code applicatif, la liberté du développeur mais propose un étalonnage de l'application sur une référence. Cette référence consigne, en termes CoTaS, les invariants métier. PATCH propose une projection à deux niveaux d'abstraction HL (High Level) et LL (Low Level) de ce modèle normatif sur l'application. Cette projection s'exprime, côté applicatif, en termes d'abstractions et de présentations.

Si cette connexion incite le développeur à une prise de recul quant à son application, elle sert aussi de base à l'évaluation de l'interaction. CatchIt se dote, pour ce faire, d'un référentiel de critères et métriques :

- CriMePro (Critères et Métriques en Proactivité) est résolument proactif : il mesure le degré effectif de réutilisation métier ;
- CriMeRéact (Critères et Métriques en Réactivité) traite de l'évaluation. La mesure est double : elle porte sur le domaine et l'utilisabilité. Pour ce dernier volet, nous introduisons un espace dédié : 3Co (Correction, Complétude et Concision opératoires). Il mesure l'efficacité du comportement opérateur en regard des connaissances métier capitalisées. A l'instar de GOMS, il suppose des opérateurs experts du domaine considéré. Mais, à sa différence, il intègre l'erreur humaine : il traque ce symptôme, potentiellement révélateur d'inadéquations homme-machine. CatchIt déploie, à ces fins, un mécanisme automatisé d'instrumentation pour filtrer, et propager vers la référence, les événements applicatifs pertinents. Les critiques établies sont proférées à fort niveau d'abstraction, de façon automatique, avec justifications à l'appui. L'instrumentation du code est assumée par l'outil.

Après ce rappel factuel de la contribution, nous nous prêtons à une analyse critique de l'approche. Nous en soulignons, en premier lieu, les originalités et points forts, puis mentionnons les limites et perspectives.

2. Originalités et points forts

Ce n'est pas tant le statut central et permanent accordé aux connaissances métier, que leur considération normative qui distingue CatchIt de ses homologues UIDE, Humanoïd ou Mastermind. Si CatchIt prône, tels ces précurseurs, la modélisation des connaissances métier, la finalité est ici d'une tout autre nature :

- il ne s'agit pas de modéliser les connaissances manipulées dans l'application, mais celles inhérentes au domaine dont émane l'application ;
- il ne s'agit pas non plus de générer le code applicatif à partir de ces descriptions, mais d'étalonner l'application sur cette référence.

Autrement dit, l'idée est de capitaliser un cadre de réflexion métier, non pour l'imposer mais toujours garantir des choix de conception réfléchis. CatchIt force à cette prise de recul, à cette maturité applicative tout en préservant la liberté du développeur. Rappelons, en effet, que « si la technologie propose, c'est la sociologie qui dispose, ce qui est logique pour des systèmes construits par des hommes pour d'autres hommes » [Amalberti 89]. La possibilité de contextualisations reste donc essentielle.

Si CatchIt innove par cette capitalisation métier organisée, il se distingue aussi par l'exploitation qui en est faite. Les connaissances alimentent une polyvalence proactive et réactive par le panachage de techniques éprouvées : modèles et liens entre modèles, d'une part ; critères et métriques, d'autre part. CatchIt est, en ceci, une proposition éclectique qui prolonge l'actuelle tendance à la normalisation : UML en notations, CORBA en interopérabilité et, plus modestement, CatchIt en métier.

De façon synthétique, mentionnons, outre la synergie atteinte entre proactivité et réactivité :

- la mise à disposition d'un cadre de réflexion pour l'analyse et la formalisation des connaissances métier ;
- l'intégration de la dualité tâche/objet et son affinement au sein de l'espace CoTaS ;
- l'intégration explicite du contexte opérationnel dans les stratégies opérateur ;
- l'incitation à l'appréciation des généricité et évolutivité des connaissances ;
- l'opportunité d'une réutilisation, et symétriquement, capitalisation logicielles maximales, sous-tendues par un format d'expression exécutable de ces connaissances ;
- la préservation des liberté et responsabilité des développeurs ;

- l'induction de formes de conception et de modèles d'architecture, notamment une décorrélation franche entre abstraction et présentation ;
- l'intégration explicite des propriétés d'utilisabilité dans l'environnement de développement ;
- l'opportunité d'une sensibilisation permanente des développeurs à ces exigences ;
- l'aspect outillé de la démarche et sa mise en œuvre au sein d'une structure d'accueil unique et fédératrice ;
- l'encouragement à des évaluations formatives de par cette automatisation ;
- le fort niveau d'abstraction des critiques, prédictives et expérimentales, proférées à l'égard de l'interaction proposée.

Soulignons enfin la généricité de CatchIt et donc sa large applicabilité : il s'adresse à des applications, existantes ou non, émanant de domaines révélant l'existence de connaissances métier.

« Le moment est venu de rechercher en toutes choses les limites de la machine et de poser en principe que, si l'on écrit à Dieu, il faut écrire à la main » Duhamel. La section suivante y est consacrée.

3. Limites et perspectives

CatchIt cible les applications émanant de domaines où l'expertise opérateur prédomine. S'il met à profit la capitalisation métier, il se heurte aux difficultés inhérentes à cette pérennisation, à savoir :

- les difficultés de modélisation liées à la complexité des domaines traités ;
- le surcoût, à court terme, des modélisations et gestions des modèles normatifs ;
- la nécessité d'une adhésion collective et de contributions équilibrées : la capitalisation doit devenir une culture d'entreprise et chacun doit s'y investir pour que l'approche fructifie.

La forte durée de vie de nos applications et leur criticité justifient cet investissement initial puis continu.

D'un point de vue technique, deux points faibles peuvent être perçus dans l'approche :

- premièrement, CatchIt ne s'adresse qu'à des applications orientées objet ;
- deuxièmement, il est nécessaire de disposer de leur code source.

Les restrictions sont, a priori, faibles :

- le premier point exclut essentiellement les applications existantes développées en langages C, ASM, etc. La tendance actuelle consistant à s'orienter vers les approches orientées objet, la limite est faible et, de surcroît, dans l'ère du temps ;
- le second interdit les comparaisons, sur critères d'utilisabilité, d'outils commercialiés. Cette utilisation ne cadre pas avec notre ambition bien ciblée, à savoir : garantir, à coûts et délais maîtrisés, une interaction homme-machine adaptée.

Si CatchIt offre, en l'état, des fondations saines et stables pour les construction et capitalisation d'applications métier, il présente des perspectives techniques et applicatives prometteuses. Mentionnons, en tout premier lieu, la finalisation de CatchIt, à savoir :

- l'extension de la couverture du processus de développement aux phases d'analyse et de modélisation orientées objet grâce à une interopérabilité entre CatchIt et, par exemple, des outils tels StP (cf chapitre II) ;
- l'enrichissement de CatchIt pour une conservation des choix de conception et notamment des relâches concédées ;
- l'extension de la couverture des propriétés d'utilisabilité ;
- bien entendu, la mise en œuvre des principes adoptés mais non encore implémentés : les liens pATcH et PaTcH HL, l'exploitation du poids des objets au sein des pré et post conditions ;
- et enfin la présentation de CatchIt, aujourd'hui au stade du maquettage.

Nous envisagerons alors :

- la mise en œuvre d'un environnement de simulation pour une couverture totale des contextes d'interaction pertinents : les préconditions des tâches et stratégies du modèle normatif déterminent ces configurations singulières. Il s'agirait d'en garantir l'occurrence par une simulation automatisée ;
- la connexion de CatchIt à des bases de connaissances cognitives et ergonomiques : dans quelles limites et conditions, par exemple, la longueur d'une trace d'interaction est-elle acceptable ? ;
- l'extension des décorations de tâches aux propriétés CARE [Coutaz 95a], aux caractères fréquent, critique, etc., ainsi que leur exploitation dans les métriques ;
- le maintien d'un modèle des écarts opérateur pour une capitalisation des contextes sensibles ;
- l'inférence d'explications et axes de correction ;
- l'intégration de la dimension collecticielle pour d'éventuelles migrations de tâches ;
- la qualification, dans ce cadre, des tâches de CoTaS selon leur nature de production, communication ou coordination [Salber 95]. Les liens PaTcH et pATcH incarneraient alors le modèle d'architecture PAC* [Calvary 96] [Calvary 97a].

Nous souhaiterions ensuite mettre à profit l'outil pour :

- la détection des conditions opérationnelles optimales d'utilisation et, à l'inverse, l'identification des contextes sensibles ;
- la maîtrise des exigences et débits d'interaction par restructurations ou migrations ;
- l'assistance à l'opérateur par aide contextuelle ou délégation grâce au contexte d'interaction ;
- l'adaptation, voire l'adaptativité, de l'interface selon le contexte d'interaction ;
- l'extraction d'expertise pour une consolidation des connaissances métier ;
- l'explication tant pour la pédagogie que l'entraînement pour toujours affermir l'expertise métier des opérateurs.

A l'évidence, « le futur ne manque pas d'avenir … » [Coutaz 98]

ANNEXE A : LA CONDITION HUMAINE

« Les hommes doivent savoir que c'est du cerveau, et du cerveau seul, que naissent nos plaisirs, nos joies, nos rires et nos plaisanteries, aussi bien que nos chagrins, nos douleurs, nos peines et nos larmes... Par le même organe, nous devenons fous et délirants, et peurs et terreurs nous assaillent... ainsi que les rêves et les divagations les plus folles... Voilà tout ce que nous fait endurer notre cerveau lorsqu'il est malade » Hippocrate.

Cette partie, dédiée à la condition humaine, vise à identifier, dans l'absolu, toute faiblesse susceptible de perturber l'homme au travail. C'est sur la notion d'astreinte que nous centrons notre approche. Prise dans le contexte du travail, elle désigne « les effets de la contrainte de travail sur l'homme en rapport avec les caractéristiques et aptitudes individuelles » [INRS]. La « contrainte de travail » se définit, quant à elle, par les « éléments de l'environnement, de l'organisation, du poste de travail et de la tâche qui peuvent avoir des répercussions physiologiques et psychologiques sur l'homme au travail » [INRS]. La littérature en propose une classification [Spérandio 92]. Elle distingue :

- les astreintes induites par la tâche elle-même, indépendamment des outils utilisés ;
- les astreintes découlant de l'informatisation de cette tâche ;
- les astreintes liées au travail sur écran ;
- et enfin les interactions entre ces astreintes, qui complexifient considérablement l'analyse.

C'est sur la base de cette taxonomie, que nous structurons l'exposé. Après un premier chapitre consacré à l'homme, nous étudions l'individu dans ses interactions avec écrans et machines. Nous y occultons, par contre, les perturbations liées à l'utilisation de périphériques spécifiques [52] et les astreintes posturales dues, par exemple, à une position assise prolongée.

[52] Clavier, souris, boule roulante , etc.

CHAPITRE I : L'HOMME

Si l’erreur est la rançon de l’intelligence, de la capacité de l’homme à innover et à faire face à des situations inattendues, il est intéressant de noter que c’est paradoxalement en en commettant qu’on progresse [Chambost 96]. Et cet apprentissage est double. Mené de façon individuelle et personnelle, il l’est aussi à l’échelle scientifique. Ainsi, du milieu des années soixante-dix à nos jours, des développements théoriques et méthodologiques ont-ils été consacrés en psychologie cognitive à l’étude des erreurs. Si une meilleure compréhension des processus mentaux fournit une conception adéquate des processus cognitifs de contrôle, elle doit aussi expliquer, non seulement la performance correcte, mais aussi les formes les plus prédictibles de défaillances humaines. Maillon central à l’analyse des erreurs, l’attention est souvent incriminée dans l’étude des ratés et lapsus quotidiens. Sa manifestation la plus connue consiste en l’inattention, c’est-à-dire une omission de vérification. Mais elle sévit aussi sous une autre forme, par un excès d’attention. Une vérification attentionnelle effectuée à une étape inappropriée de la séquence d'actions peut en effet engendrer une défaillance du mode de contrôle [Reason 93]. Le danger s’exprime donc résolument en termes de sollicitations inopportunes des ressources attentionnelles. Aussi, orientons-nous ce chapitre vers ces considérations : après une étude du potentiel et des limites humaines en la matière, nous en examinons les facteurs influents.

1. Ressources attentionnelles : potentiel et limites

Les termes d’attention et de vigilance sont souvent pris pour synonymes. Pourtant, une différence fondamentale les distingue : tandis que l’attention s’apparente à une ressource dédiée à la gestion des modalités sensorielles permettant à un individu d’être vigilant, la vigilance se réfère à un processus consommateur en la matière [Giry 95].

1.1. Attention

Dans sa forme initiale, la théorie des ressources assimile l'attention à une réserve de moyens disponibles pour le traitement de l'information. A partager entre toutes les opérations mentales, l’attention sert alors toutes tâches concurrentes. La littérature étayant cette théorie s’intéresse aux doubles tâches pour en mesurer les conflits. Si l’on en considère un continuum, borné d’une part par des tâches entièrement automatisées, ne requérant donc pas ou peu de ressources attentionnelles, et, d’autre part, des tâches exigeant un niveau attentionnel si élevé qu'elles peuvent difficilement être réalisées correctement, même isolément l’une de l'autre, seule la partie médiane a réellement fait l’objet d’études. Dans cette région, c'est la similarité entre tâches qui s’avère cruciale : plus leurs composantes sont comparables, plus les risques d’interférences mutuelles sont élevés. Elles sont en effet susceptibles de requérir, au même moment, des ressources identiques, et sont donc rivales. Les études montrent que cette interférence peut survenir à différentes étapes de la séquence de traitement de l'information. Il n’existe pas de section critique, elle intervient là où les tâches sont les plus compétitives vis-à-vis des mêmes fonctions. L’interférence se répercute alors de diverses manières : l’interruption de l'une des activités, la détérioration de la performance, le ralentissement de l'exécution ou des commutations entre activités [Reason 93].

Cette théorie rejoint le principe dit du «canal unique » qui stipule que, dès lors que l’attention est sollicitée, les informations transitent par une unité centrale à capacité limitée [Paty 95]. Pour des raisons de saturation, l'homme ne peut alors effectuer plus d’une opération simultanément. Bien entendu, un parallélisme reste possible, mais seulement pour des opérations programmées ou apprises. Mentionnons par exemple le fait d’entendre, de voir, d’imaginer ou de gribouiller. Bien entendu, l’éducation et l’apprentissage contribuent à améliorer la gestion des tirages sur cette banque centrale. Ils facilitent notamment le passage des stratégies contrôlées et sérielles à des stratégies parallèles et automatiques, clé de l’économie. Mais attention, cet apprentissage peut alors fausser la perception du monde environnant, comme en témoignent les travaux de Minsky : si l'on présente brièvement à quelqu'un un objet et qu’on lui en demande une description, sa représentation prototypique influencera largement la perception. Ainsi, par exemple, une horloge sans aiguille s’en verra tout de même agrémentée dans la description [Reason 93].

Si la gestion multitâches reste coûteuse, l’homme s’accommode, à l’inverse, très bien d’activités mono-tâche. Doté d’un pouvoir de concentration exceptionnel, il focalise remarquablement son attention sur le problème à traiter. Il fait notamment abstraction d’événements non pertinents de son environnement immédiat. Cette sélectivité fait l’objet de nombreuses études. Il s’avère que des percées peuvent néanmoins survenir, selon le type des informations issues du canal à négliger. La proximité spatiale entre sources de signaux et l’accord contextuel entre le contenu de l’intrus et l’activité courante en sont des facteurs favorisants. Mentionnons, par exemple, comme intrusion courante, l'apparition de nos nom ou prénom dans l’environnement. Si ces intrusions naturelles sont spontanées et incontrôlées, notons par contre le coût en temps d’une attention sélective contrôlée [Reason 93].

Si les sources concurrentes affectent la capacité attentionnelle, il en est de même pour l'habitude et le changement. Et il est intéressant de noter que ces deux opposés se rejoignent dans le traitement : la stratégie consiste en effet bien souvent à attendre qu'une configuration familière réapparaisse pour agir [Reason 93]. Ainsi, les problèmes multi-dynamiques ouverts constituent un terrain d'étude privilégié. Par leur exceptionnelle variabilité additionnelle, dont les traitements d'urgence, l'anticipation ne peut être systématique. Comment alors maintenir, à son plus fort potentiel, la concentration de l'opérateur pour qu'il puisse mobiliser, au moment opportun, son capital attentionnel ? C'est le problème de la vigilance.

1.2. Vigilance

La définition du concept de vigilance varie selon le point de vue adopté. Si, en physiologie, elle se réfère à un «état d'efficacité élevée du système nerveux central qui assure la promptitude des réponses adaptatives », elle désigne, en psychologie, « un état de préparation pour détecter et répondre à de petites modifications spécifiées de l'environnement survenant de façon aléatoire » [Giry 95]. C'est cette approche qui retient ici notre attention.

La compréhension et l'évaluation des états de vigilance revêtent une importance fondamentale dans de nombreux domaines, civils ou militaires [Lagarde 91]. Conduite automobile, pilotage de trains, d'avions, surveillance de centrales nucléaire, garde d'enfants, soins médicaux sont autant d'applications qui illustrent l'enjeu d'une vigilance infaillible. Motivées par cette foule de domaines candidats, les recherches sur le sujet sont nombreuses : c'est bien, à terme, la mise à disposition d'heuristiques pour un maintien efficace de la vigilance humaine qui est recherchée. Aussi, tant la compréhension du processus de variation de la vigilance que la découverte des facteurs influants font l'objet d'études.

Les différentes recherches menées dans le domaine confirment l'existence d'une succession de vigilances partielles finalisant une vigilance motivée. Schématiquement, trois systèmes de vigilance collaborent pour l'adaptation de l'homme à son environnement : la vigilance diffuse, affective et focalisée.

- La vigilance diffuse correspond à l'activité d'éveil des structures de traitement de l'information. Elle est modulée par les interrelations qu'elle entretient avec les messages sensoriels. Facilitée par tout stimulus inhabituel, elle est, au contraire, inhibée par tout message habituel dépourvu de signification.
- La vigilance affective tient aux facteurs de motivation du sujet. Elle joue un rôle primordial dans sa vulnérabilité et son adaptation. Elle contrôle son attention sélective.
- Enfin, la vigilance focalisée, en étroite relation avec l'attention sélective, implique un double processus de facilitation de certaines activités associatives et d'inhibition d'autres activités.

En première approximation, on peut admettre que l'organisme dispose d'un potentiel de vigilances à gérer. Alors qu'une vigilance soutenue (nécessitée, par exemple, par un débit élevé d'informations à traiter, par la complexité de ces données, etc.) l'épuise rapidement, une vigilance moins intense peut être maintenue plus longtemps. Autrement dit, une tâche simple consomme moins d'énergie qu'une tâche complexe.

De nombreuses expériences sont aujourd'hui menées pour identifier les facteurs facilitant ou, au contraire, déstabilisant la vigilance. Il semble aujourd'hui qu'ils s'organisent en deux catégories selon leur nature endogène ou exogène [Giry 95]. Un facteur est dit endogène s'il est « lié aux propriétés de réactivité cyclique du système nerveux central », tandis qu'un facteur exogène est relatif « aux perturbations apportées par l'environnement sur l'état des structures cérébrales » [Giry 95].

Facteurs endogènes

Parmi les facteurs endogènes, on recense :

- les rythmes circadiens, qui correspondent à l'alternance veille-sommeil ;
- le sommeil ;
- et les rythmes ultradiens, qui traduisent l'alternance de bonne et d'hypo vigilance. Notons que la durée totale d'un cycle de vigilance serait de 100 minutes, l'amplitude de la variation pouvant être de 50%. Les baisses de vigilance seraient dues à un état de fatigue, atteint après 30 à 45 minutes d'attention soutenue. Des phases d'hyper-vigilance peuvent compléter ces cycles traditionnels : elles se caractérisent pas une attention hyperfocalisée (diminution de l'attention diffuse) sur la tâche à accomplir, ceci au détriment de toute information périphérique.

Facteurs exogènes

Les facteurs exogènes se répartissent eux-mêmes en deux catégories selon qu'ils sont relatifs à la tâche ou à l'environnement.

- Les caractéristiques de la tâche interfèrent de façon conséquente avec le coût physiologique et psychique de celle-ci. Tandis que les tâches élémentaires très simples (détection d'un point lumineux ou d'un bip sonore) semblent relativement résistantes à la fatigue, les tâches complexes nécessitant une sélection ou une interprétation d'informations s'avèrent nettement plus sensibles à l'accumulation de la fatigue. La présentation de l'information joue alors un rôle essentiel, tant en intensité, durée, fréquence que répartition spatiale : pour chacune de ces caractéristiques, il s'agit de trouver le juste compromis pour optimiser l'effort de détection et de traitement et la surcharge cognitive. [Wisner 90] rapporte, à ce sujet, que si les signaux très rares, à savoir moins de cinq par heure, sont très difficiles à détecter, les signaux trop fréquents (plus de 200 par heure) polluent, à l'inverse, l'exécution de la tâche. La redondance d'information se soumet à cette même règle : y recourir systématiquement peut produire l'inverse des effets escomptés. Il faut par ailleurs savoir qu'un contexte multitâches, par la multiplication des sources de signaux, aurait un effet défavorable par un épuisement plus rapide du potentiel de vigilance disponible. L'organisation temporelle joue alors un rôle fondamental. Multiplier les pauses, même très brèves, permet de reconstituer le potentiel attentionnel. Alterner les tâches, opter pour des sessions de surveillance courtes (1 heure si possible) en sont d'autres consignes.
- Les facteurs d'environnement semblent influencer, de façon variable, la vigilance. Tandis que les ambiances stressantes dégraderaient la performance, des variations d'ambiance pourraient, au contraire, jouer un rôle éveillant. Ainsi, tandis que les bruits relativement peu élevés, continus, monotones, non significatifs agissent peu sur la vigilance, un signal bref, rare, isolé peut l'améliorer (à condition, bien entendu, qu'il ne constitue pas une tâche concurrente !) [Wisner 90]. Mais attention, là encore, la nature de la tâche intervient : alors que dans une activité à faible vigilance, l'environnement pourrait stimuler, dans une tâche à fortes contraintes, un environnement agressif semblerait néfaste. Interviendraient les caractéristiques des locaux, tant en termes d'éclairage, de régulation thermo-hygrométrique, de bruit, vibrations, mouvements de plate-forme, composition d'atmosphère, etc., mais aussi des facteurs d'environnement psychique comme la pharmacologie, l'entraînement ou la motivation. Notons, à ce sujet, la faible influence des primes d'efficience ! [Wisner 90]

Bien entendu, même si la littérature ne rapporte que très peu d'études sur la combinaison et l'intrication de ces facteurs, gardons à l'esprit qu'ils influencent, de façon complexe, l'état de vigilance des opérateurs [Giry 95]. N'oublions pas non plus que la situation opérationnelle y joue un rôle crucial. Ainsi, le problème de la vigilance est-il complexe et seul un savant dosage semble pouvoir y répondre. En l'absence de cette recette, étudions en les principaux ingrédients : la fatigue, la charge de travail et le stress.

2. Facteurs influents : physiques et psychologiques

C'est l'activité nerveuse qui nous intéresse ici et plus précisément ses facteurs influents. Rappelons qu'elle « consiste en un traitement d'information à un régime défini par la nature et la fréquence des signaux reçus, la complexité des choix et la nature des décisions à prendre » [Universalis 93]. Si la fatigue en est une manifestation tant physique que psychologique, son interaction avec l'activité est double : elle peut résulter d'une intensité excessive et inversement compromettre le rythme courant.

2.1. Fatigue

« Le terme de fatigue désigne à la fois un sentiment vécu, n'apparaissant qu'à travers le récit personnel de celui qui l'a ressenti, et un ensemble de signes notés et enregistrés par un observateur impartial. Cette dualité, permet d'opposer la fatigue subjective et la fatigue objective. Seule cette dernière peut être réellement définie avec précision par le physiologiste. Pour Scherrer, la fatigue est une baisse d'activité, d'un système vivant (cellules, tissus, organes...) pour une incitation constante, liée à l'activité de ce système et réversible par sa cessation transitoire. Cette définition s'applique aux activités aussi bien musculaires que psychosensorielles de l'organisme entier ; elle concerne le travail professionnel, l'activité sportive et les contraintes de la vie en société. La fatigue normale doit être distinguée de manifestations apparemment identiques, mais d'origine pathologique que l'on préfère nommer asthénie. Une baisse d'activité, lorsqu'elle n'est plus réversible par le repos, est un signe de vieillissement, soit d'usure (plus ou moins prématurée), soit d'une altération de la structure » [Universalis 93].

La fatigue correspond soit à une baisse de la vigilance lorsqu'une attention trop soutenue est demandée, soit à une augmentation du seuil de sensibilité des organes sensoriels » [Universalis 93]. Ses manifestations sont

nombreuses et les réflexes ne sont pas épargnés : impossibilité de suivre le régime initial, augmentation des erreurs ou des omissions, accroissement de la période de latence, contraction plus progressive et moins ample [Wright 73]. Les remèdes dépendent alors de la nature et des causes de la fatigue. Le repos ou le changement d'activité sont favorables à la récupération. Mais moins que les actions curatives, ce sont les consignes prescriptives qui nous intéressent ici. Issues d'approches multidisciplinaires associant ergonomes, médecins et statisticiens, elles restent spécifiques à un domaine donné. Elles aboutissent notamment à un aménagement du travail professionnel [Harris 91]. Cible privilégiée de ces analyses, le domaine aéronautique a fait l'objet de nombreuses études. L'objectif est d'assurer une performance maximale des opérateurs. [Harris 91] s'intéresse par exemple aux personnels de cabine effectuant des vols courts-courriers. Il révèle l'existence d'interactions significatives entre le grade et le sexe des sujets, leur ancienneté, leur aéroport de base et le type d'avions empruntés.

A ne pas confondre avec ce concept de fatigue, la notion de charge dont l'intrication est forte : si la fatigue peut résulter d'une charge excessive, elle peut, à l'inverse, devenir facteur de charge, par l'amoindrissement des ressources disponibles induit [Spérandio 92].

2.2. Charge de travail

Un consensus est aujourd'hui atteint pour admettre que la charge de travail quantifie l'écart qui peut exister entre « les exigences du travail et la capacité disponible de l'opérateur pour y faire face » [Spérandio 92]. C'est bien de la différence entre l'estimation de cette complexité et des ressources mentales disponibles (intelligence, capacité perceptive, mnésique, etc.) dont il s'agit [Doppler 94]. Shopenhauer disait d'ailleurs : « ce qui est important pour l'homme, ce n'est pas ce qui lui arrive mais ce qu'il pense qu'il lui arrive ».

Les recherches menées dans le domaine tendent à confirmer l'existence de trois facettes, s'exprimant elles-mêmes en termes de charge : les aspects physique, cognitif et psychique[53] [Wisner 89]. Présents dans toute activité, chacun d'entre eux peut engendrer une surcharge ou une souffrance. Il faut alors savoir qu'en interrelation les uns avec les autres, une surcharge de l'un peut augmenter la charge de l'autre. On perçoit ainsi mieux la multiplicité des facteurs qui peuvent agir sur les capacités cognitives des opérateurs. Et on comprend encore mieux pourquoi l'erreur est humaine et qu'il est préférable d'apprendre à l'apprivoiser [Portejoie 97]. Nous nous focalisons ici sur la charge cognitive.

Ce sont essentiellement les études menées en Intelligence Artificielle qui ont permis de prouver son existence et de l'évaluer. Cette discipline se réfère à l'Intelligence qui permet de trouver la solution à un problème de mathématiques ou de physique où toutes les données de l'énoncé sont nécessaires et suffisantes. Il est alors possible de suivre le raisonnement oral, d'en mesurer la complexité et de proposer des moyens de la réduire. Mais dans la réalité, et en particulier dans le contrôle de procédés, les problèmes s'avèrent souvent sans solution et des compromis s'imposent. Les données disponibles dépassent alors le strict nécessaire, il faut en sélectionner, certaines sont peu fiables, suspectes, surprenantes. Bien souvent, c'est une combinaison spatio-temporelle entre données qui aide à la décision. Et les différences interindividuelles, notamment l'expertise, entrent alors en jeu [Wisner 89].

Ainsi, ce ne sont pas tant les caractéristiques de la tâche prescrite qui sont pertinentes, mais celles de la tâche réelle. Leur nombre est variable tant diffèrent les niveaux de formation et de familiarité de ces opérateurs [Wisner 89]. Le caractère de ces derniers et notamment leur sensibilité au stress intervient aussi.

2.3. Stress

Le "stress" est aujourd'hui un sujet d'actualité. Depuis plusieurs siècles, de nombreux ouvrages lui sont consacrés, dans son acception la plus large du terme. Ce sont les réponses de l'organisme à tout changement de l'environnement qui sont étudiées, ces adaptations visant la santé et la survie du sujet. Ainsi tout changement de température extérieure ou d'absorption d'eau par exemple mettent en œuvre des mécanismes dont le rôle est d'empêcher toute modification importante des variables physiologiques [Vander 85].

A l'origine de cette interprétation, se placent les conceptions de Claude Bernard. Selon lui, toutes les réactions d'un sujet n'ont qu'un but : celui de maintenir la constance des conditions de vie dans le milieu intérieur, c'est à dire l'équilibre du milieu aqueux où baignent toutes les cellules constitutives de l'organisme. Cannon a créé, pour

[53] L'aspect psychique peut être défini en termes de conflits au sein de la représentation consciente ou inconsciente des relations entre la personne (ego) et la situation (dans ce cas : l'organisation du travail).

qualifier cette fonction fondamentale, le mot « homéostasie » [Cannon 74]. L'évolution de la pensée et des techniques médicales au cours des cinquante dernières années a été si largement influencée par la thèse de ces deux grands physiologistes qu'elle a longtemps sous-estimé l'importance du syndrome d'adaptation [Universalis 93]. Aujourd'hui, de nombreuses expériences sont menées pour en cerner principes et limites. Elles contribuent parallèlement à figer une terminologie encore hésitante.

2.3.1. Terminologie

Le mot stress est un anglicisme dont l'équivalent français est variable suivant le contexte. Le terme vient du vieux français "estrece", lui-même dérivé du latin "strictus", dont la traduction est « serré », « pressé », « comprimé ». La signification de ce mot est sujette à de nombreuses interprétations. Selon la définition de Selye ([Selye 62] cité dans [Mallion 82]), le premier à employer ce terme en médecine, le « stress » est un état qui se manifeste par un «syndrome spécifique englobant tous les changements aspécifiques survenus dans un système biologique ». Ce concept a fait faire un grand pas à la compréhension des phénomènes physiopathologiques. Depuis affiné par des théoriciens pour une prise en compte de la subjectivité sociopsychophysiologique du phénomène (aussi appelée holisme phénoménologique dans le jargon médical), le stress peut être vu comme une perturbation psychophysiologique de l'homéostasie résultant d'une discordance existante ou perçue entre l'individu et son environnement social [Clancy 93].

Le concept d'homéostasie, évoqué dans cette définition, désigne « l'équilibre de l'organisme au sein de son environnement » [Universalis 93]. C'est un concept dynamique. En éternelle fluctuation, il oscille autour d'une valeur moyenne, nominale, vers laquelle l'organisme s'efforce de converger. Le stress désigne en fait un déséquilibre conséquent, c'est-à-dire pour lequel un seuil, dit seuil de stress, est franchi. Atteint sous l'effet d'un ou plusieurs stimuli, les conséquences varient tant selon la nature de ces contraintes que des différences interindividuelles.

Cette définition du stress correspond à une vision globaliste du terme. Elle dénote à la fois le stimulus agressant, la réaction qu'il déclenche et le mécanisme par lequel on aboutit à une adaptation. Cette définition ne fait pas l'unanimité et certains auteurs restreignent la portée de ce terme au seul stimulus. Pour une vision complète du phénomène, nous préférons opter pour la première définition.

Si les termes d'émotion, d'anxiété et d'angoisse sont souvent pris pour synonymes du concept de stress, il convient de les différencier. Ils n'y sont pourtant pas étrangers : ils peuvent en résulter ou, au contraire, le générer.

- L'émotion désigne «un trouble subit, une agitation passagère causés par un sentiment vif de peur, de surprise ou de joie» [Larousse 94].
- L'anxiété désigne « un état de tension interne, éprouvé comme déplaisant et pénible par le sujet. C'est un état d'attente d'un événement potentiel, qui en surgissant, mettrait en danger l'intégrité de la personne. L'anxiété répond à un risque, c'est-à-dire à un danger latent, d'origine extérieure. Dans cette mesure, l'anxiété a une valeur adaptative, puisqu'elle constitue en quelque sorte une préparation psychologique à la menace et oriente les efforts du sujet pour y parer grâce à l'attention et à la prudence » [Wisner 89].
- L'angoisse est également « un état d'attente péniblement ressenti, mais cette fois la menace est subjective et provient de l'intérieur. Le sujet sait reconnaître son origine individuelle et endogène. L'angoisse résulte d'un conflit intra-psychique, c'est-à-dire une contradiction située à l'intérieur de l'appareil mental. L'angoisse est donc fonction de la structure de la personnalité de chaque sujet et de son histoire passée » [Wisner 89].

Etant donné cette terminologie, étudions la subjectivité du phénomène.

2.3.2. Subjectivité

La subjectivité du stress tient selon [Clancy 93] à de multiples facteurs : la nature du stimulus, le contexte social, la personnalité de l'individu, son vécu, etc. [Smelick 79] (cité dans [Mallion 82]) compare la diversité de ces situations à l'image d'un sablier : « un spectre large de stimuli converge vers un goulot commun et s'élargit ensuite sous la forme de multiples réactions adaptatives qui peuvent être modifiées par des agents particuliers, dits "conditionnants" : principalement, le stimulus mais aussi l'organisation mentale de l'individu ».

Si le stimulus en soi ne semble pas toujours contenir la qualité émotionnelle, c'est qu'il l'acquiert à travers la signification que lui donne le sujet [Mallion 82]. Ce caractère subjectif de la stimulation émotionnelle peut obéir à plusieurs critères :

- dans certains cas, il s'agit de stimuli déclenchant l'émotion à coup sûr chez tous les êtres vivants, comme l'annonce du décès d'un cher par exemple ;
- il peut aussi s'agir de conditions qui ne sont émouvantes que pour les individus d'un certain niveau culturel : le caractère de la perception repose alors sur des critères d'apprentissage ;
- enfin, il est des causes d'émotion propres à un seul individu et même à un seul moment de son existence : les critères émotionnels se réfèrent alors aux conditions affectives du moment actuel.

Holmes partage cette conviction de l'existence d'invariants, provoquant, à coup sûr, une décharge émotionnelle. [Clancy 93] en rapporte les statistiques, elles permettent d'établir une échelle de criticité (Figure 116).

Evénements	Criticité
Décès d'un conjoint	100
Divorce	73
Séparation maritale	65
Blessure ou maladie personnelle	53
Mariage	50
Grossesse	40
Problèmes sexuels	39
Changement de responsabilité professionnelles	29
Réussite personnelle marquante	28
Problèmes hiérarchiques	23
Changement de conditions de travail	20
Changement d'activité sociale	18
Vacances	13
Noël	12

Figure 116 : Contraintes et criticité (adapté de [Clancy 93])

Concernant maintenant les variations interindividuelles constatées dans la perception d'une même contrainte, [Clancy 93] évoque plusieurs facteurs. Il mentionne l'hérédité, la personnalité, l'éducation, l'environnement et le vécu. Il est aujourd'hui prouvé que l'histoire antérieure du sujet influence sa réaction aux différentes contraintes psychosensorielles. L'historique de ses rapports avec ses environnements successifs est entretenu dans ce qu'on appelle la mémoire nerveuse. Celle-ci se souvient des expériences passées, plaisantes ou déplaisantes, ce qui permet de tenter de reproduire les premières et d'éviter les secondes. Mémoire nerveuse qui nous permet d'accumuler les « éléments » capables de fournir un nouveau comportement grâce à la fonction imaginative. Mémoire qui se souvient surtout des actions inefficaces ou douloureuses, ou de celles punies ou qui risquent de l'être, par la socio-culture.

Constitutionnelles ou acquises, ces contraintes influencent la réaction mise en œuvre par l'organisme pour lutter contre le déséquilibre homéostasique occasionné : nous étudions cette réaction dans la section suivante.

2.3.3. Réaction

C'est le déséquilibre homéostasique au-delà du seuil de stress qui provoque la réaction psychophysiologique de l'organisme. Elle met en œuvre, selon la nature de la contrainte, deux types de stress : l'eustress et le distress. Tandis que l'eustress résulte de contraintes quotidiennes (les eustressors), le distress est généré par des stimuli extrêmes, de longue durée ou inhabituels (les distressors). Alors que les premières contraintes sont des contraintes salutaires qui contrôlent l'homéostasie en vue de la rétablir, les secondes génèrent de l'angoisse, malsaine pour l'organisme. Prévert parlait du « silencieux vacarme de l'angoisse, le bruit qui ne fait pas de bruit ». Rappelons que c'est la manifestation de conflits intérieurs que l'on ne parvient pas à résoudre [Wisner 89]. L'expérience montre que l'eustress s'accompagne d'un sentiment de bien-être, contrairement au distress qui se caractérise par des sensations psychologiques extrêmement négatives [Clancy 93] (Figure 117).

Eustress	Distress
Augmentation de l'acuité mentale	Diminution de l'attention aux détails, négligence, faibles performances professionnelles
Plaisir - bonheur	Crises émotionnelles, tristesse, mélancolie
Euphorie	Léthargie, apathie

Figure 117 : Effets comparés des eustress et distress

Si malgré la réaction de l'organisme, l'équilibre homéostatique n'est pas recouvré suffisamment vite, alors l'eustress se transforme en distress (Figure 117). Ce laps de temps est subjectif. Il dépend de l'origine du déséquilibre et de l'efficacité du contrôle homéostatique. Pour illustrer ce phénomène, [Clancy 93] prend l'exemple du boxeur : la conversion de l'eustress en distress correspond aux premiers coups, aux premières fatigues qui déstabilisent le sportif et viennent à bout de son moral d'acier.

Soit le contrôle homéostatique interne permet alors de rétablir l'équilibre, soit le stade de la maladie est atteint et le patient nécessite un suivi, voire un traitement médical [Clancy 93] (Figure 118).

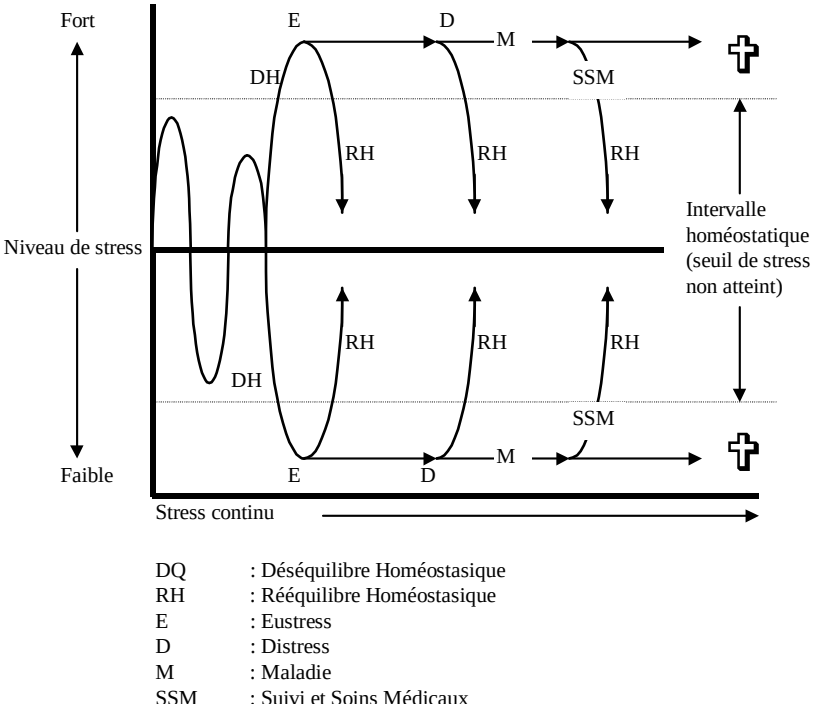

Figure 118 : Equilibre et déséquilibre homéostatiques (adapté de [Clancy 93])

En résumé, l'organisme s'efforce de maintenir un équilibre interne par le biais de réactions. Selon l'efficacité de cette défense, les oscillations autour de la valeur nominale seront plus ou moins nombreuses. Ce sont maintenant ces réactions que nous étudions. Elles respectent, d'après Selye, un schéma triphasé constitué d'une réaction d'alarme, d'une phase de résistance puis d'épuisement [Logeay 90].

Alarme

L'alarme est la perception d'une contrainte qui initie le processus de réaction. Une phase de commentaires est alors activée pour déterminer les actions adaptatives optimales et vérifier le potentiel de ressources disponibles à cet égard. C'est pendant cette phase de commentaire intérieur que l'émotion, ou réaction psychologique, s'extériorise [Mallion 82]. Son ampleur dépend, rappelons-le, du degré de la menace, de l'environnement, de la personnalité du sujet, etc. Comme toutes manifestations émotionnelles, ces réactions sont parfaitement automatisées. L'alarme est en fait contrôlée, de façon prédominante, par le système nerveux sympathique qui affecte les organes effecteurs viscéraux tels le cœur, le cerveau ou les muscles. Elle se manifeste par une accélération du rythme cardiaque, une augmentation de la profondeur respiratoire, une hypertension artérielle, une affluence sanguine au niveau de ces organes, et au contraire une irrigation réduite des viscères, de la peau,

etc. et un arrêt de la digestion [Clancy 93]. D'où certains dénominateurs communs, malgré les différences interindividuelles : l'immobilisation soudaine, la pâleur du visage, les sursauts, tremblements, maladresses ou sueurs.

Des variations internes, et notamment hormonales [Mallion 82], s'opèrent alors pour rétablir l'équilibre homéostasique. Si, malgré ces combinaisons complexes, le déséquilibre persiste et, si la contrainte demeure, alors l'organisme enchaîne sur la phase dite de résistance.

Résistance

La phase de résistance ou d'adaptation est contrôlée de façon prédominante par le système endocrinien. La plupart des hormones excrétées sont alors des agents hyperglycémiques ravitaillant les cellules en énergie pour lutter contre l'agression. [Clancy 93] souligne la notion de capital en matière d'adaptation énergétique : par la métaphore d'un compte bancaire où seuls les retraits sont possibles, il indique que chaque trouble sociopsychophysiologique affecte le montant disponible. Un vieillissement irréversible est alors enclenché.

En cas d'échec de l'adaptation, la phase d'épuisement est amorcée.

Epuisement

Le contrôle homéostasique ayant échoué, les signes de l'alarme réapparaissent. La maladie est alors déclarée et une intervention médicale est indispensable. L'origine du stress doit être identifiée et traitée pour que l'équilibre puisse à nouveau s'établir. En cas d'échec, la survie du patient est en jeu.

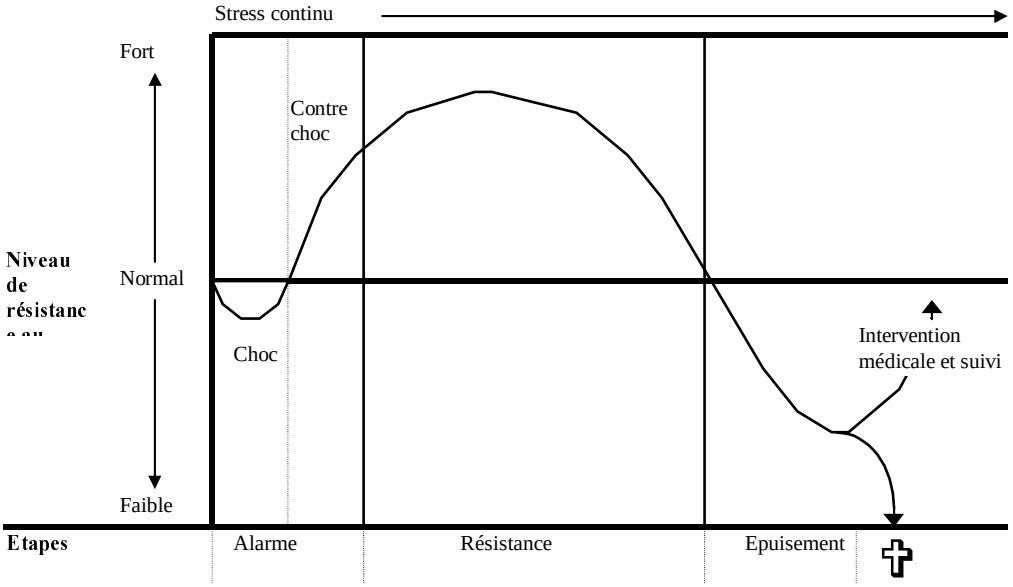

Figure 119 : Réaction triphasée au stress (adapté de [Clancy 93])

Cette étude nous ramène finalement à une question fondamentale : le stress est-il bénéfique ou nuisible ? Il est bien difficile d'y répondre. D'une manière générale, les réactions neurovégétatives observées en situation de stress peuvent, à l'extrême, être bénéfiques ou néfastes. Ces réactions sont bénéfiques lorsqu'elles permettent une extériorisation par une réaction adaptée où l'individu se réalise pleinement. Mais, dans certains cas, les sujets sont inhibés dans leurs possibilités d'action et d'extériorisation. On observe alors un effet de tunellisation de la pensée sans passage possible à l'acte : le sujet se bloque sur la contrainte psychosensorielle perçue et tout autre stimulus lui échappe par ailleurs.

3. Synthèse

En résumé, rappellerons-nous que :

- « ce qui est important pour l'homme, ce n'est pas tant ce qui lui arrive mais ce qu'il pense qu'il lui arrive » Schopenhauer ;
- et sa réaction est d'autant plus amplifiée qu'il ressent une incapacité à surmonter la contrainte et l'importance des conséquences de cette incompétence [Clancy 93].

La conviction d’une parfaite maîtrise de la situation est donc fondamentale. Le contrôle de l'événement est alors efficace et peu de perturbations biologiques ou physiologiques, centrales ou périphériques en résultent. Ce n'est que lorsque ce contrôle devient impossible que les perturbations apparaissent. Aussi, protéger nos opérateurs de tout stress n’est pas optimal : d’une part, l’individu serait incapable de surmonter un danger imprévu ; d’autre part, cette alternative supprimerait définitivement l’opportunité d’un eustress, dont les bienfaits ont été décrits.

La collaboration homme-machine doit donc, en conclusion, être suffisamment étroite pour que l’homme ne ressente jamais de sentiment d’impuissance face à une situation critique. Les programmes de simulation et d’entraînement sont alors fondamentaux pour que l’individu, non seulement, s’approprie l’interface, mais apprenne aussi à surmonter des situations délicates. Notons enfin que, des critères personnels intervenant dans les réactions de l’individu au stress, la définition d’un profil adapté d’opérateurs est fondamental.

CHAPITRE II : L'HOMME ET L'ECRAN

Avec la démocratisation de l'informatique, l'introduction massive d'écrans, tant dans les monde scolaire que professionnel, soulève de nombreuses questions relatives à la santé des personnes concernées. Les publications dans le domaine témoignent de cet intérêt massif. [Chevaleraud 84] en propose une analyse originale. Sur la base d'une comparaison entre le travail sur écran et la réception d'un spectacle, il démontre la spécificité du premier. Tant les modes d'utilisation, la nature de l'information, la durée de l'observation et l'environnement diffèrent entre ces deux optiques.

- Alors que le spectacle est contemplé de loin, sur du matériel choisi, dans une ambiance personnalisée, le travail sur écran s'effectue en vision rapprochée ou intermédiaire, sur un appareil et dans un environnement imposés.
- Tandis que le spectateur choisit son programme dans un but culturel ou décontractant, le travailleur subit l'information, sans toujours y trouver un quelconque intérêt.
- Si le spectateur est libre d'interrompre à tout moment son plaisir, le travailleur se doit, au contraire, d'effectuer la mission dans le temps imparti.
- Enfin, tandis que le spectateur maîtrise des facteurs extérieurs, tels le niveau sonore, la température, l'éclairage, le réglage de l'appareil, les fumées, etc., le travailleur les subit. Et l'expérience montre qu'elles peuvent nuire considérablement à sa concentration, son confort, sa sérénité, c'est-à-dire finalement son bien-être.

Le travail sur écran est donc bien spécifique et justifie les recherches dans le domaine. Afin d'en identifier les éléments pertinents contribuant à la performance d'un opérateur impliqué dans une tâche informatisée, nous rappelons, dans un premier temps, les processus visuels mobilisés pour une performance perceptive. Nous mentionnons alors les contraintes et astreintes liées à ce type de travail puis en précisons les contre-indications.

1. Performance visuelle

Organe central à la vision, l'oeil nous permet de percevoir des informations dans notre environnement. Transmises au cerveau, ces données sont alors traitées en relation avec la mémoire. C'est la relation entre ces deux activités, visuelle perceptive et cognitive, qui conditionne essentiellement la qualité du travail et le confort de l'opérateur. Elle détermine la dite « performance visuelle », essentiellement fonction de deux processus psychophysiologiques fondamentaux : la sensibilité aux luminances et la discrimination des formes.

1.1. Sensibilité aux luminances

La luminance détermine, dans une direction donnée, l'aspect lumineux d'une source ou d'une surface éclairée. La sensation visuelle de luminosité et notamment la sensation de clarté en dépendent [X 35-103]. Particulièrement affectée par les effets d'accommodation[54], la performance visuelle est surtout dégradée dans le sens des luminances décroissantes. Pouvant atteindre, dans ce cas, plusieurs dizaines de minutes, elle est, dans l'autre sens, de l'ordre de la seconde [X 35-103]. Aussi est-il important de respecter un certain équilibre des luminances dans le champ visuel[55] de l'opérateur. Composé de la zone de travail[56] et d'une zone dite périphérique[57], celui-ci doit offrir à l'opérateur une homogénéité entre espaces. Ceci est d'autant plus important que, dans l'exécution d'une tâche informatisée, l'opérateur commute souvent entre écran et instructions papier. Une luminance régulière doit être assurée, et tout éblouissement notamment proscrit. Le réflexe pupillaire protège certes la rétine de ces agressions, mais les mécanismes musculaires d'accommodation (réglage optique de chaque oeil et convergence des deux axes visuels) et de rétrécissements exigent un certain temps de réponse.

On distingue deux types d'éclairement :

- L' « éblouissement perturbateur » ou « incapacité par éblouissement » résulte de la présence d'une source brillante dans le champ visuel. Tel un voile s'interposant entre l'œil et la tâche visuelle, celle-ci provoque

[54] L'accomodation désigne « l'augmentation de la puissance optique de convergence de l'œil, dans la vision rapprochée (moins de 6 m), par diminution du rayon de courbure de la face antérieure du cristallin » [X 35-103].
[55] « Etendue angulaire des directions de l'espace dans laquelle un objet peut être perçu lorsque la tête et les yeux sont immobiles » [X 35-13].
[56] « Région de l'espace où se trouve la tâche à accomplir et où il convient de distinguer le détail à percevoir et le fond sur lequel il se détache » [X 35-103].
[57] « Portion de l'espace environnant, perçue au-delà de la zone de travail, lorsque les yeux sont dirigés vers la tâche » [X 35-103].

une baisse de l'acuité visuelle[58] et de la sensibilité lumineuse : c'est la luminance dite de voile[59] [X 35-103]. Elle dépend du flux lumineux incident et de l'angle que forme la direction de la source avec l'axe visuel.
- L' « éblouissement inconfortable » n'engendre aucune incapacité visuelle mais une gêne, contribuant, à terme, à la fatigue. Le désagrément ressenti est une fonction directe de la luminance de la source, de l'angle solide sous lequel elle est vue, de son contraste avec le fond et une fonction inverse de l'angle que forme la direction de la source avec l'axe visuel [X 35-103].

Si la prévention impose de tenir compte de l'équilibre des luminances, de l'éblouissement, des décrets fixent aussi des normes en matière de reflets. Sources d'inconfort, de sensations désagréables, ils perturbent l'opérateur et peuvent occasionner une incapacité visuelle transitoire.

Traité par la rétine, le signal physique est ensuite transmis au cerveau. Le processus de reconnaissance de formes y est alors activé.

1.2. Discrimination des formes

La discrimination des formes est un processus psychophysiologique complexe de reconnaissance visuelle de l'environnement. Il sollicite essentiellement trois fonctions : la vision des contrastes, l'acuité visuelle et la perception des profondeurs et distances.

- Concernant la vision des contrastes, rappelons que le contraste est une « appréciation subjective de la différence d'apparence entre deux parties du champ visuel vues simultanément ou successivement. Il peut s'agir d'un contraste de couleur, d'un contraste de luminance, d'un contraste simultané ou successif » [X 35-103]. La vision des contrastes repose sur l'existence d'un seuil dit « seuil de contraste » ou « seuil différentiel de luminance » en-dessous duquel la distinction entre deux plages contiguës devient impossible.
- L'acuité visuelle mesure « la capacité de perception distincte d'objets paraissant très rapprochés » [X 35-103]. Elle dépend, à la fois, des performances physiologiques de l'appareil de vision et des caractéristiques de l'environnement. Mentionnons notamment les qualités optiques de l'œil (dimensions et sphéricité, transparence des milieux, courbure des surfaces et accommodations), les facteurs rétiniens, l'intégrité des voies nerveuses, la luminance et le contraste [X 35-103].

Dans la perception des profondeurs et distances, interviennent tant les fonctions physiologiques que psychologiques ou intellectuelles. Y contribuent, par exemple respectivement, la variation de la convergence ou le souvenir de la taille et de la forme d'objets familiers. L'intégration cérébrale de la perception visuelle est nécessaire à l'évaluation mentale de ces profondeurs et distances dans le champ environnant. Ainsi, est-ce bien la qualité de cette perception qui conditionne toute la vision. Examinons en les contraintes et astreintes.

2. Contraintes et astreintes

Si la sensibilité aux luminances et la discrimination des formes conditionnent la performance visuelle, la littérature rapporte bon nombre de contraintes, supposées l'affecter par ailleurs. Parmi elles, la vision en lumière intermittente et la vision d'éclats brefs :

- « La perception du mouvement d'un objet éclairé par une source intermittente est modifiée par effet stroboscopique : il peut en résulter une apparence de ralentissement, d'immobilité ou d'inversion de sens, qui, s'ils ne sont pas recherchés pour eux-mêmes, sont des facteurs potentiels de risque » [X 35-103]. Les variations de lumière sont perçues si la fréquence de fluctuation est suffisamment faible. C'est le phénomène du « papillotement[60] » ressenti généralement comme une gêne et facteur de fatigue visuelle.
- Concernant les éclats brefs de lumière, notons qu'ils sont perçus avec un certain retard, mais de façon persistante. Leur luminosité apparaît, par ailleurs, plus intense que celle d'une source plus durable, de même luminance.

[58] « Capacité de perception distincte d'objets paraissant très rapprochés » [X 35-103].
[59] « Effet de voile produit par des sources ou des surfaces éblouissantes situées dans le champ visuel qui conduit à une réduction du contraste perçu, donc à une diminution de la performance visuelle et de la visibilité » [X 35-103].
[60] « Impression de fluctuation de la luminance ou de la couleur se produisant lorsque la variation du stimulus lumineux se situe entre quelques hertz et la fréquence de fusion des images (50 à 75 hertz) » [X 35-103].

La vision des couleurs est aussi invoquée pour l’instabilité qu’elle présente : désaturation progressive des objets colorés longuement examinés et des objets très éclairés, difficultés de la mémorisation des couleurs, etc. Notons, à ce sujet, la subjectivité du rendu des couleurs : non seulement l’aspect coloré des objets dépend de la répartition spectrale de la source lumineuse, mais aussi de l’adaptation chromatique de l’observateur. Lors d’un changement d’illuminant, une distorsion de couleur est notamment perçue.

La densité des informations à percevoir et leur complexité doivent aussi être intégrées : des études révèlent en effet que chez les travailleurs sur écran, le nombre de fixations visuelles est proportionnel à la densité des informations contenues dans le texte, alors que la durée de ces fixations est fonction de la difficulté des codes employés. Ainsi, la fatigue visuelle dépend-elle aussi de la façon dont les informations sont présentées, codées, structurées, de la qualité du dialogue homme-machine et de son intégration à l’environnement [Spérandio 87].

S’il est certain que tout travail sur écran induit une charge sensorielle et mentale, il est néanmoins difficile d’en parler, en général, tant sont variées tâches et conditions d’exercice. Notons néanmoins, de façon générale, la transposition des contraintes, avec une prépondérance des astreintes mentales et parfois l’émergence de nouvelles astreintes physiques : l’opérateur ne voit plus l’objet sur lequel il travaille, il doit l’imaginer à partir de données symboliques [Doppler 94]. La charge mentale ainsi induite, à l’instar de la charge de travail, est définie comme « le rapport entre, d’une part, l’ensemble des exigences mentales du travail, lié au traitement d’information effectué ou requis, et, d’autre part, la capacité des ressources mentales disponibles de l’opérateur » [Spérandio 92]. Par ressources mentales, on entend « l’attention, la perception, la mémorisation, la représentation mentale, le raisonnement, l’apprentissage, le langage » [Spérandio 92]. Les processus affectifs n’entrent pas directement dans cette énumération, mais « ne sont pas étrangers au concept de ressources cognitives, au moins par l’intrication des effets » [Spérandio 92].

3. Synthèse

En conclusion, le travail sur écran sollicite de manière indiscutable la fonction visuelle et est source potentielle de fatigue psychosensorielle. Conséquence possible d’une charge excessive prolongée, elle dépend toutefois des stratégies oculomotrices mises en œuvre, elles-mêmes dépendantes d’une grande panoplie de facteurs : types d’écrans, types de tâches, conception des interfaces, etc. [Spérandio 92]. Ressentie de manière variable, cette contrainte souffre d’un effet de contagion, augmentant exponentiellement le nombre de ses détracteurs. Sont néanmoins indéniables l’effort mental accru et parfois l’anxiété provenant de l’incertitude de la compréhension [Wisner 89].

CHAPITRE III : L'HOMME ET LA MACHINE

L’évolution des techniques place l’homme dans des activités particulières qui sollicitent différemment ses compétences [Keravel 97]. On assiste à une reconfiguration du travail humain consistant désormais à contrôler les machines : surveillance, détection d’aléas ou reprise de situations déviantes, telles sont typiquement les tâches qui leur incombent désormais. Ces activités requièrent une attention rapide, précise et interventionniste, et ceci d’autant plus que les risques encourus sont importants. Le maître mot est alors la fiabilité et la « bête noire » l’erreur humaine. La solution s’exprime alors en termes de qualité du couplage homme-machine : nous en étudions les modalités dans la deuxième section. Nous commençons pas les antipodes : fiabilité et erreur humaine.

1. Fiabilité et erreur humaine

La fiabilité globale d’un système dépend de la qualité de la coopération homme-machine établie. Elle se décline en fiabilités technique et humaine, la première influençant, à l’évidence, la seconde. Rien de tel, en effet, qu’un dysfonctionnement pour déstabiliser l’utilisateur, fausser son modèle conceptuel de l’outil. Rien de tel pour favoriser l’erreur humaine.

Aucun consensus n’est encore atteint aujourd’hui pour définir, de façon unanime, l’erreur humaine. A son sens le plus générique, le terme d’erreur « couvre tous les cas où une séquence planifiée d’actions mentales ou physiques ne parvient pas à ses fins désirées, et quand ces échecs ne peuvent être attribués à l’intervention du hasard » [Reason 93]. L’approche la plus classique consiste à considérer l’erreur comme un écart à une norme. Mais cette référence est elle-même discutée : est-ce la tâche prescrite par les concepteurs, celle représentative d’un opérateur type du domaine ou enfin l’activité individuelle d’un utilisateur donné ? Ne peut-elle pas aussi être subjective et se référer alors à l’intention de l’opérateur ? [Jambon 96]

Face à ce large spectre et tant d’indécision, les taxonomies fleurissent. [Reason 93] en propose une méta-classification, sur la base de trois niveaux : le comportemental, le contextuel et le conceptuel, qui correspondent respectivement aux questions quoi, où et comment [Reason 93]. Selon ces considérations, la production des erreurs s’articule autour de trois principaux aspects : « la nature de la tâche et les conditions dans lesquelles elle est réalisée, les mécanismes qui régissent l’activité et les particularités individuelles du sujet » [Reason 93]. [Reason 93] identifie trois types fondamentaux d’erreur selon les étapes cognitives auxquelles elles apparaissent :

- les fautes surviennent en phase de planification. Elles peuvent se définir comme « des déficiences ou des défauts dans les processus de jugement et/ou d’inférence, qui sont impliqués dans la sélection d’un objectif ou dans la spécification des moyens pour l’atteindre, indépendamment du fait que les actions basées sur ce schème de décision se déroulent ou non conformément au plan » [Reason 93] ;
- les lapsus apparaissent en phase de stockage d’une séquence d’actions. Généralement moins observables, ils résultent de défauts de mémoire ;
- les ratés interviennent en phase d’exécution. Potentiellement observables, ils se manifestent alors sous la forme d’actions visiblement non planifiées.

[Reason 93] en propose un parallèle avec le modèle de Rasmussen [Rasmussen 86]. Inspiré de l’organisation cérébrale en niveaux, il suggère une distinction tripartite des niveaux d’activité :

- au niveau basé sur les automatismes, l'activité humaine est contrôlée par des configurations mémorisées d'instructions préprogrammées. A ce niveau, les erreurs prennent deux formes : soit le déclenchement d'un comportement cohérent mais inopiné, soit des omissions ;
- le niveau basé sur les règles s'applique à la « résolution de problèmes familiers, dont les procédures sont contrôlées par des règles de production du type : si [état] alors [diagnostic] ou si [état] alors [action de remédiation]. Ici, les erreurs sont typiquement liées à de mauvaises classifications de situations, qui conduisent à l'application de règles erronées ou au rappel incohérent de procédures » [Reason 93] ;
- le niveau basé sur les connaissances déclaratives intervient dans les situations nouvelles, pour lesquelles une planification consciente en temps réel s’impose. A ce niveau, les erreurs sont dues à des limitations de ressources et à des connaissances incomplètes ou incorrectes.

En référence à ce modèle, devenu norme de marché aujourd’hui, les ratés, lapsus et fautes surviennent respectivement aux niveaux automatismes, règles et connaissances. Tandis que les fautes surviennent lors de la formation d’une intention, les ratés et lapsus résultent de défaillances à des niveaux subordonnés : sélection de

l'action, exécution et stockage de l'intention. Bien plus subtiles et dangereuses, les fautes sont difficiles à détecter. [Reason 93] les organise en deux catégories selon qu'elles traduisent une défaillance ou un manque d'expertise. Tandis que dans le premier cas, elles correspondent à l'application inappropriée d'un plan ou d'une solution préétablie, la deuxième alternative schématise le cas où, à défaut d'une routine prête à l'emploi, l'individu se développe un plan d'actions sur la base de principes et de connaissances, pertinentes ou non. Il s'avère en effet que « les sujets humains préfèrent agir par reconnaissance de configurations spécifiques au contexte, plutôt que de calculer ou d'optimiser » [Reason 93]. Ce modèle s'appuie, non seulement sur l'extraordinaire facilité avec laquelle la mémoire humaine encode, stocke et retrouve ensuite un ensemble de représentations schématiques, virtuellement sans limite, mais encore sur la rationalité limitée, la rationalité paresseuse et la persistance de la prévision.

Cette tendance à recourir, si possible, à des règles familières vise à limiter la charge cognitive. Connu sous le nom de « rationalité limitée », ce principe postule un ensemble borné de ressources, notamment attentionnelles, et vise à en économiser le stock disponible. Cette rationalité sévit ainsi typiquement lors de raisonnements temps réel, d'une grande puissance de calcul. C'est le niveau connaissance du modèle de Rasmussen qui en est la principale victime [Reason 93]. S'allie à cette rationalité, la rationalité dite «réticente ». De nature à conduire à une confiance excessive dans des indices apparemment familiers, elle privilégie l'application facile de solutions bien connues. Plutôt que de favoriser l'exploration de nouvelles voies potentiellement profitables, elle nous dirige inlassablement vers les sentiers battus. En somme, les êtres humains s'acharnent toujours à reconnaître des configurations, plutôt que de calculer et d'optimiser. Ils accordent une confiance excessive à la justesse de leurs connaissances et se complaisent dans l'observation des données validant leur choix, négligeant tout signe discordant. Mais attention : « à celui qui n'a qu'un marteau, tout problème apparaît comme un clou ! » [Reason 93].

Ainsi, la présentation des informations est-elle un point clé dans la fiabilité des systèmes. Nous en discutons dans la section suivante.

2. Collaboration homme-machine

[Keravel 97] recense trois objectifs potentiellement visés dans l'émission d'une information :

- contrôler la présence physique de l'opérateur ;
- lui transmettre une information et compter sur sa contribution réfléchie ;
- déclencher sa participation pour :
  - vérifier une donnée ;
  - transmettre une information ;
  - intervenir interactivement sur le système ;
  - stopper sa propre action ou interrompre immédiatement le système.

Pour chacun de ces objectifs, [Keravel 97] précise la nature de l'information correspondante et la contribution humaine attendue (Figure 120).

Objectif de l'information	**Nature de l'information**	**Contribution humaine attendue**
Contrôler la présence physique de l'opérateur	Stimulation réactogène	Réactions motrices
Informer l'opérateur	Etats, paramètres	Compréhension de la situation
Impliquer l'opérateur	Avertissement / alerte / alarme	Vérification d'une cohérence Transmission de données Diagnostic Action différée Arrêt immédiat de l'action opérateur ou du système

Figure 120 : Typologie de la contribution humaine à la Sûreté de fonctionnement d'une situation. Adapté de [Keravel 97]

Cette étude conclut que l’information est transmise sous forme d’indication, d’avertissement, d’alerte ou d’alarme et qu’elle conditionne la réaction de l’opérateur (Figure 121) :

- un avertissement est un « élément qui induit une prise de conscience et provoque une organisation particulière » [Keravel 97]. Il se réfère à des faits survenus dans l’évolution du système, à des contraintes ou à des aléas. [Keravel 97] en propose une typologie selon la profondeur de l’impact des faits latents ou déclarés :
  - au niveau 1, les avertissements sont corrélés à un risque (potentiel) d’incohérence, de déviance. Ils incitent l’opérateur à intervenir ponctuellement de façon autonome ;
  - au niveau 2, les avertissements répondent à une anomalie de coordination des données. Ils engagent l’opérateur à transmettre des éléments d’information ;
  - au niveau 3, les avertissements dénoncent la non pertinence de l’évolution du système, compte tenu de la situation courante. Ils incitent l’opérateur à un diagnostic après analyse de cette situation et de l’état transitoire du système ;
- une alerte vise à induire une intervention, dans un délai fixé et pour de conditions données, suite à une problématique identifiée, de profondeur connue. C’est souvent l’extrapolation dans le temps des anomalies courantes qui permet d’évaluer la gravité des conséquences potentielles ;
- l’alarme est le dernier stade de ce crescendo. Elle vise à induire une action précise et immédiate suite à un risque identifié, de gravité imminente. La réaction escomptée est une interruption de l’action opérateur en cours, un arrêt d’urgence du système ou le déclenchement d’une prise en charge, etc.

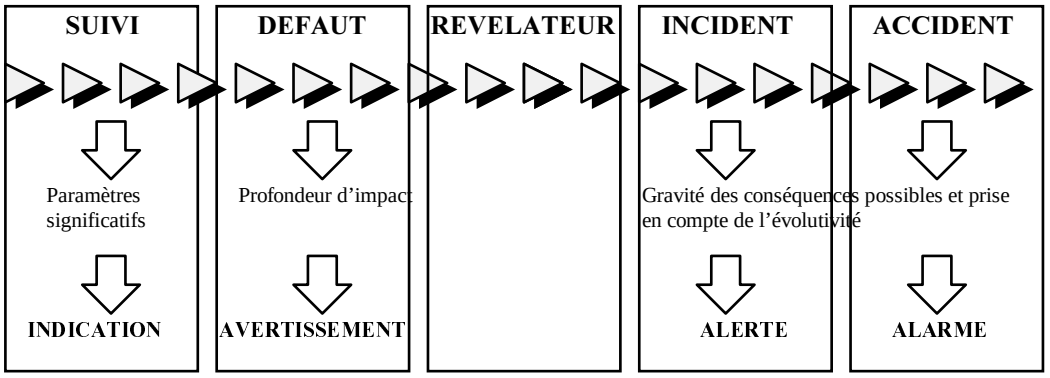

Figure 121 : Types fonctionnel d’information [Keravel 97].

[Keravel 97] s’intéresse alors aux comportements induits par une alarme. Elle recense cinq types de réaction : l’ignorance, la maîtrise de gestion, l’implication à un niveau superficiel, une exigence d’investigation complémentaire, la mise en œuvre d’une procédure d’investigation pour en déceler la cause (Figure 122).

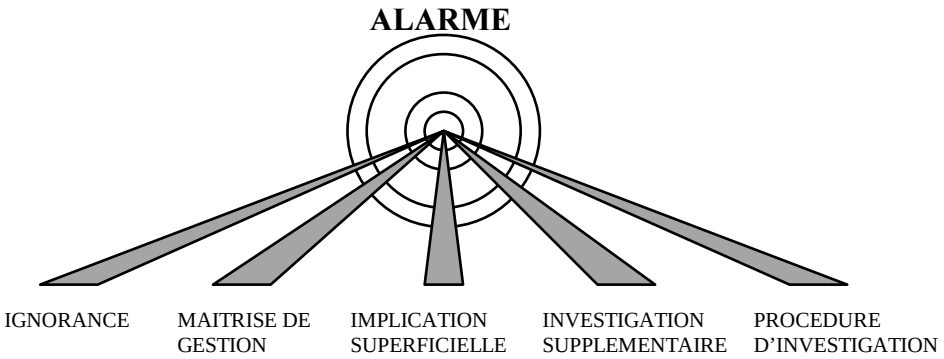

Figure 122 : Les prises de décision possibles en cas d'alarme [Keravel 97].

[Keravel 97] constate que les investigations complémentaires préconisées par l'alarme ne sont pas systématiquement menées. L'opérateur est parfois tenté par des résolutions superficielles, qui s'avèrent insuffisantes et retardent finalement les investigations recommandées. Schématiquement, les comportements humains face à une alarme sont de trois types :

- l'opérateur s'implique pour traiter l'alarme ;
- l'opérateur est conscient d'un dysfonctionnement et de la nécessité d'une intervention, mais la diffère par manque d'information ou de moyens de réaction ;
- l'opérateur se contente de noter l'anomalie.

Il est donc toujours important, et notamment dans la conception des alarmes, de cerner la place et le rôle accordés à l'opérateur, de définir la contribution humaine attendue, d'optimiser les moyens et de prévoir les défaillances humaines. De la qualité et de la rigueur des réponses apportées dépendra grandement la fiabilité de l'opérateur. C'est bien du partage des tâches entre l'homme et la machine dont il s'agit.

Il n'existe hélas, en la matière, aucune méthode miracle permettant d'optimiser l'allocation des tâches. A défaut d'heuristiques, [Charles 96] propose une grille d'analyse et rappelle qu'une bonne solution consiste bien souvent en un compromis entre le tout manuel et le tout automatique (Figure 123).

	AUTOMATISMES	OPERATEURS HUMAINS
☺	• rapidité • possibilité d'un traitement important de l'information • caractéristiques constantes dans le temps (pas de phénomène de perturbation ou de fatigue) • grande fiabilité	• vision d'ensemble d'un problème • possibilités d'anticipation • souplesse, adaptation à des événements non prévus • mise en œuvre immédiate d'algorithmes très complexes (reconnaissance de formes, par exemple)
☹	• pas d'adaptation à une évolution de l'environnement • traitements complexes difficilement fiabilisables • logique déterministe	• subjectivité pouvant entraîner des réactions difficilement prévisibles • vitesse de réaction faible • fiabilité insuffisante dans l'accomplissement d'opérations simples et monotones • difficulté pour accomplir un travail très long (fatigue)

Figure 123 : Critères de partage des tâches entre l'homme et la machine [Charles 96].

[Charles 96] préconise :

- d'automatiser les fonctions
  - déterministes et insensibles aux événements extérieurs ;
  - répétitives et d'algorithmique peu complexe ;
  - requérant un temps de réponse incompatibles des capacités humaines ;

- de déléguer à l'opérateur les fonctions
  - d'algorithmes très complexes mais traités simplement par l'homme (la reconnaissance des formes par exemple) ;
  - dépendantes d'événements extérieurs ;
  - nécessitant une logique de raisonnement floue ou l'appréciation d'une situation dans son ensemble.

[Keravel 97] confirme cette approche en déclarant que dans un «environnement incertain et non stationnaire, l’homme est indispensable en matière de flexibilité, fiabilité, adaptation aux changements (anticipation) et gestion des aléas (résolution de problèmes en logique floue) ».

Malgré l’émergence de ces critères, le partage des tâches reste, en pratique, des plus délicats. Comme le souligne Pierre Baud [Chambost 96], des problèmes inédits apparaissent régulièrement, soulevés par les progrès techniques censés contribuer à la sécurité : en affranchissant les pilotes de tâches susceptibles de provoquer des erreurs, on introduit de nouveaux concepts, qui, à leur tour, soulèvent le problème de la représentation mentale. C’est un des paradoxes soulevés par [Amalberti 89] : si un système d'aide vise à mieux coupler l'homme à la machine, il génère aussi son propre problème de couplage. Il semblerait ainsi que si l’automatisation élimine les petites erreurs, elle induit de grosses maladresses [Wiener 89]. C’est a priori le cas du mont Sainte Odile [Jambon 97].

Aussi, si le rôle de l’interface Homme-Machine est de coupler l’homme à la machine, son enjeu est bien de réduire l’occurrence des erreurs humaines. Média de communication entre deux logiques différentes, deux langages différents, l’IHM est l’unique passerelle entre les spécificités fonctionnelles de l’homme et de la machine [Amalberti 89] : elle joue un rôle fondamental dans l’élaboration du modèle mental. 10 % des accidents aériens sont imputables au matériel et à ceux qui l’emploient : le poids de l’interface homme-machine y est unanimement reconnu [Renaudie 89].

3. Synthèse

Tandis que dans le passé, les erreurs opérationnelles relevaient en général d’une charge de travail inadaptée, elles semblent plutôt provenir aujourd’hui d’une mauvaise représentation mentale de la situation [Chambost 96]. C’est la qualité de ce modèle mental qui semble conditionner la faculté des opérateurs à maîtriser réellement l’expérience et à récupérer des situations dégradées [Charles 96]. Comme le souligne Pierre Baud [Chambost 96], plusieurs cas peuvent se présenter : soit, le pilote est influencé par une idée préconçue qui fausse son interprétation, soit il n’exploite que partiellement les multiples indicateurs à sa disposition. Les techniques de vérifications croisées, l’étude du partage des tâches, l’enseignement des facteurs humains sont instituées à ces fins.

ANNEXE B : SMALLTALK

CatchIt est aujourd'hui implémenté en Smalltalk, dont nous proposons ici une présentation synthétique. L'ambition reste pragmatique, l'objectif bien ciblé : il ne s'agit pas de présenter Smalltalk dans l'absolu, mais d'apporter au lecteur les éléments d'information nécessaires à la compréhension de la mise en œuvre. Aussi, après en avoir rappelé les grandes lignes, nous développons les aspects fondamentaux exploités dans CatchIt. Nous les organisons suivant les facettes fonctionnelles de Smalltalk que nous présentons dans cette première partie.

1. Généralités

« Smalltalk est à la fois un langage de programmation, un système d'exploitation, un environnement de programmation, une méthodologie de conception et de programmation » [Mével 87]. S'il excelle dans le prototypage d'applications, c'est non seulement pour son caractère semi-interprété, mais aussi la forte réutilisation logicielle qu'il permet, grâce à l'absence de frontière entre l'application et le système : c'est la notion de machine virtuelle, que nous développerons après avoir présenté le langage.

Cet effort d'unification est une caractéristique essentielle de Smalltalk. L'économie du concept y est permanente : elle prône une réutilisation logicielle intensive, permise par le principe du « tout-objet ». Nombres, caractères, listes, dessins, programmes, processus, outils, etc. sont des objets : ils mémorisent des informations, y accèdent et répondent, par ailleurs, à des messages portant typiquement sur ces informations propres. Les classes système ne dérogent pas à la règle : elles sont accessibles à l'utilisateur et candidates à la réutilisation.

Seuls les messages permettent de manipuler les données encapsulées dans les objets. Autrement dit, les messages constituent l'interface de ces objets et délimitent notamment le caractère privé/public des informations détenues. L'objet maîtrise l'image qu'il publie : il s'apparente à une boîte noire, pourvue de points d'entrée, dont il gère les sorties. Seule la connaissance de ces points d'entrée, et plus précisément de leur existence, est nécessaire : peu importe leur codage. C'est la notion d'encapsulation.

C'est bien souvent de la nature de l'objet que découlent les messages auxquels il est sensible. Aussi, les objets sont-ils regroupés en classes, l'une des notions fondamentales du langage. Elle fait l'objet d'une description dans la section suivante.

2. Le langage

Nous articulons cette description du langage en deux parties : tandis que la première en présente les notions fondamentales, la seconde introduit la syntaxe.

2.1. Notions fondamentales

Les notions que nous développons ici relèvent de la définition des classes : elles précisent les termes de classe, instance, superclasse, métaclasse, méthode, variable et pseudovariable. Nous les envisageons dans cet ordre.

2.1.1. Classes et instances

Une classe fédère des objets de même nature : ils sont définis par un même ensemble d'attributs et répondent aux mêmes messages. Parmi ces attributs, certains ont une valeur spécifique à l'instance, d'autres, au contraire, sont génériques, de valeur commune à toutes les instances : ils sont respectivement appelés variables d'instance et de classe. Cette subtilité permet, en fait, de discriminer les attributs selon un critère de généricité. Par exemple, dans la classe Voiture, les attributs longueur et largeur sont spécifiques aux instances et donc modélisés en variables d'instance. A l'inverse, le nombre de siège avant est fixe (2 en l'occurrence) et relève donc d'une variable de classe.

Dans la lignée du tout-objet, les classes sont elles-mêmes des objets, instances d'une classe dite métaclasse. Les métaclasses sont des classes, contenant elles-mêmes des variables d'instance, dites variables d'instance de méta-classe.

L'héritage entre classes permet de partager des éléments de description. Chaque classe hérite des propriétés de sa classe mère ou superclasse, à savoir : variables d'instance, de classes et méthodes, ces dernières pouvant, si nécessaire, être redéfinies. En tête de l'arborescence se trouve la classe Object, la seule à ne pas posséder de superclasse.

Ainsi, une classe est-elle définie par :

- son nom;
- sa superclasse;
- ses variables d'instance et de classe;
- ses méthodes et autres variables accessibles.

La section suivante envisage ce dernier point.

2.1.2. Méthodes et variables

Les messages sont implémentés par des méthodes : elles décrivent la façon dont l'objet répond au message. Toutes les instances d'une même classe utilisent la même méthode pour répondre au même message. Selon le cas, la valeur peut être spécifique à l'instance ou, au contraire, commune à la classe. La méthode est alors respectivement implémentée en méthode d'instance ou de classe, c'est-à-dire définie dans la classe ou sa métaclasse. Par exemple, dans la classe Voiture, les méthodes retournant les longueur et nombre de sièges avant sont respectivement implémentées en variables d'instance et de classe.

Une méthode est définie par :

- un nom ;
- et un corps.

Ce nom est le nom du message auquel répond la méthode. Le corps est une liste d'instructions manipulant des variables. Parmi ces variables, on distingue les variables proprement dites, c'est-à-dire des variables de valeur modifiable par affectation, des pseudo-variables, qui, à l'inverse, référencent toujours le même objet.

Smalltalk offre cinq types de variables :

- les variables d'instance, de type nommé ou indexé : tandis que les premières sont identifiées par un nom, les secondes le sont par un index afin d'en permettre la définition d'un nombre quelconque ;
- les variables de classe, partagées par toutes les instance d'une même classe ;
- les variables globales, partagées par tous les objets, et rangées dans un dictionnaire nommé Smalltalk ;
- les variables partagées, accessibles par les instances de plusieurs classes ;
- et enfin les variables temporaires, locales à une méthode.

Les pseudo-variables ne peuvent être modifiées par affectation. Parmi elles :

- les constantes, c'est-à-dire dans la lignée du tout-objet, des variables référençant constamment le même objet : *nil* (la valeur par défaut d'une variable), *true* et *false* (deux instances de la classe Boolean) ;
- *self* qui identifie le récepteur d'un message ;
- et *super* qui référence la superclasse.

Les notions fondamentales de définition de classe étant ainsi introduites, présentons la syntaxe associée.

2.2. Syntaxe

Nous adoptons ici la structuration de [Mével 87]. L'auteur identifie quatre types lexicaux :

- les littéraux représentent des objets prédéfinis tels que les nombres, les caractères, les chaînes;
- les objets variables dénotent des objets accessibles dans le contexte d'une expression;
- les messages;
- et enfin les blocs d'instructions qui représentent une expression dont l'évaluation est différée.

Nous les envisageons dans cet ordre.

2.2.1. Littéraux

Smalltalk compte cinq types de litttéraux :

- les nombres représentent des valeurs numériques et répondent typiquement à des messages d'ordre mathématique. Ils sont notés en base 10 : *2*, *2.0*, *-2*, etc. et *2 sin* signifie l'envoi du message *sin* à l'objet *2* ;
- les caractères incarnent les éléments de l'alphabet. Ils sont précédées du caractère $. Ainsi *$2*, *$s*, *$i*, *$$* représentent respectivement les caractères *2*, *s*, *i* et *$* ;
- les chaînes de caractère sont encadrées de quotes : '2 sin' ;
- les symboles dénotent des chaînes de caractères alphanumériques. Elles sont précédées du caractère # : #2sin ;
- les tableaux s'apparentent à des ensembles de littéraux, accessibles par un index numérique. La liste est encadrée de parenthèses, précédées par un # : #(2 'sin' #(2 #sin)).

Les littéraux étant définis, consacrons-nous maintenant à la description des variables.

2.2.2. Variables

A l'exception des variables indexées, toute variable porte un nom. Par convention, celles dont la portée est limitée à l'objet commencent par une lettre minuscule, contrairement à celles accessibles par plusieurs objets qui se distinguent par une première lettre majuscule.

L'affectation d'une variable s'effectue via le caractère dédié « := » :

- à sa droite, une expression dont l'évaluation retourne un objet ;
- à sa gauche, la variable à laquelle est affecté cet objet.

Par exemple, *unFloat := 2.0* permet d'affecter le réel 2.0 à la variable *unFloat.*

Afin d'enrichir ces exemples, présentons maintenant la syntaxe des messages.

2.2.3. Messages

Les messages interviennent dans des expressions et permettent notamment les interactions entre objets. Une expression implique un récepteur, un sélecteur et d'éventuels arguments. Tandis que le sélecteur est dénoté par un littéral, le récepteur et les arguments sont récursivement décrits par des expressions. Par exemple, 2+8 est une expression dans laquelle le récepteur est 2, le sélecteur + et l'argument 8.

Dans cet exemple, le sélecteur est un message dit binaire : il n'implique qu'un seul argument. Il existe deux autres types de messages : ceux dits unaires, n'acceptant aucun paramètre, et ceux dits à mots clés, en autorisant, au contraire, un nombre quelconque. Chaque argument y est introduit par un mot clé, suffixé du caractère ':'. Par exemple, l'expression suivante fixe le premier élément du tableau à la chaîne 'premier élément' :

```
tableau at: 1 put: 'premier élément'.
```

2.2.4. Méthodes

Le nom d'une méthode est celui du message auquel la méthode permet de répondre. Une méthode est constituée de son nom suivi de la déclaration d'éventuelles variables temporaires puis d'expressions séparées par des points. La déclaration des variables temporaires est délimitée par les caractères '|'. Une méthode retourne une valeur, via le caractère dédié '^'.

Considérons par exemple une méthode de calcul de moyenne de deux valeurs val1 et val2. La méthode est implémentée comme suit :

```
moyenne: val1 et: val2
|somme|
somme := val1 + val2.
^somme / 2
```

2.2.5. Blocs

Les blocs sont des objets utilisés dans les stuctures de contrôle : boucles, tests et boucles conditionnelles. Un bloc est une suite d'expressions, séparées par des points et encadrées par des crochets. Son évaluation ne se fait que sur requête explicite : c'est cette spécificité qui lui vaut son adéquation aux structures de contrôle.

Des arguments peuvent lui être passés en entrée. La syntaxe est la suivante : précéder chaque attribut du caractère ':' et séparer cette énumération du caractère '|'. Par exemple : [:i:j| var1 := i. var2 := j] définit un bloc d'instructions à deux paramètres d'entrée i et j.

L'exécution d'un bloc est explicitement déclenchée par l'envoi d'un message, à savoir :

- value, dans le cas d'un bloc sans paramètre;
- value: pour un bloc à un argument;
- value:value: dans le cas de deux arguments, etc.

Ainsi s'achève cette présentation du langage. Consacrons-nous maintenant à la description du système.

3. Le système

Du système Smalltalk, nous évoquons, en tout premier lieu, les notions d'image et de machine virtuelles. Nous nous consacrons alors à la gestion des processus puis détaillons le traitement des événements utilisateur. Nous présentons, à ces fins, le modèle d'architecture logicielle MVC.

3.1. Image et machine virtuelles

A ne pas confondre avec la notion de machine virtuelle, le terme d'image virtuelle désigne le système Smalltalk, c'est-à-dire, à l'origine, un ensemble de classes de base. L'absence de frontière entre le système et l'application se consigne par une cohabitation de ces classes prédéfinies et des classes applicatives : l'image se réfère alors à toutes ces classes, qu'elles soient d'origine système ou applicative.

La machine virtuelle désigne, à l'inverse, un ensemble de méthodes, dites primitives, implémentées en langage machine. Ces méthodes permettent de rompre les cycles d'évaluation d'expressions, en sollicitant non plus d'autres méthodes Smalltalk, mais ces méthodes compilées. Les opérations mathématiques en sont des exemples typiques : +, *, /, etc.

L'utilisateur peut enrichir cette machine virtuelle par la définition de primitives personnelles, à implémenter aujourd'hui en langage C. Il lui suffit alors de recompiler la machine virtuelle pour en obtenir une nouvelle version enrichie. Si cette extension nécessite une compilation, il en est de même pour les interventions pratiquées sur l'image virtuelle : sur l'envoi du message accept, le code source est compilé en langage machine. En revanche, l'exécution d'un programme reste à dominante interprétée : un CTRL-C en interrompt le déroulement ; des points d'arrêt peuvent être interactivement placés dans le source; etc. D'où le terme dédié de semi-interprétation pour refléter ce caractère hybride : compilation des méthodes, mais exécution interprétée.

3.2. Gestion des processus

Un processus représente un fil d'activité, décrit par une séquence d'expressions. C'est une instance de la classe Process. Le ramasse miettes Smalltalk en est un exemple. Plusieurs méthodes permettent de créer ces processus. Mentionnons, par exemple, la méthode 'forkAt: unePriorité' utilisée dans CatchIt : le récepteur en est un bloc d'instructions; elle permet, outre la création du processus, sa gestion parmi les processus concurrents activables. La machine virtuelle Smalltalk étant mono-processeur, une gestion de l'activation des processus s'impose en effet. Cette gestion est assumée par la classe ProcessorScheduler, via son unique instance Processor.

3.3. Modèle MVC

Le modèle MVC est un modèle à agents militant pour une structuration tripartite des informations : tandis que le Model se réfère à l'abstraction de l'objet manipulé, les Controller et View en désignent respectivement les présentations en entrée et sortie. Les relations entretenues par ses trois facettes sont les suivantes :

- les vue et contrôleur se connaissent mutuellement;
- il en est de même pour les vue et modèle ;
- le contrôleur connaît le modèle; en revanche, le modèle ne connaît pas le contrôleur.

Les classes StandardSystemView et StandardSystemController implémentent les mécanismes de base permettant de définir et de gérer les fenêtres au sens le plus général. Il est donc judicieux de dériver les classes applicatives de ces deux classes système : c'est la politique adoptée dans CatchIt.

Outre ces apports langage et système, c'est aussi une contribution méthodologique que fournit Smalltalk : nous l'envisageons dans la section suivante.

4. La méthodologie

C'est finalement l'ouverture du système qui confère à Smalltalk cette force méthodologique. L'absence de frontière incite à une réutilisation maximale et répond ainsi à une exigence de productivité : hormis les méthodes primitives, toutes sont accessibles au développeur. Il peut les appeler telles quelles ou les spécialiser, selon ses spécifications logicielles.

Si cette incitation à la réutilisation logicielle est prégnante dans Smalltalk, elle est largement facilitée par les consignes de modularité, inhérentes au modèle MVC. L'intrication y est forte : MVC offre un découpage fonctionnel modulaire, aisément implémentable par réutilisation des classes de base génériques, prêtes à l'emploi.

Retenons que la méthodologie Smalltalkienne tend à optimiser les efforts des développeurs. Elle s'articule autour de deux pivots fortement corrélés : les modularité et réutilisabilité logicielles. Mais si ces consignes sont, en pratique, tant respectées, c'est largement grâce à l'environnement de développement adjoint : il répond à cette même quête de productivité.

5. L'environnement

Si la relative lenteur à l'exécution de Smalltalk et surtout son non-déterministe ont souvent nuit à l'extension du produit, la convivialité et la richesse de son environnement ont, en revanche, toujours été reconnues. Smalltalk propose un cadre de travail intégré, permettant d'organiser, de parcourir et de manipuler toutes les classes et méthodes de l'image virtuelle. Des catégories permettent d'organiser classes et méthodes selon des critères personnels : par exemple, noyau fonctionnel et dialogue, pour les classes; initialisation et méthodes d'accès, pour les méthodes. Plus qu'une incitation à la structuration, c'est finalement le support méthodologique de Smalltalk que renforce cet environnement. Notons que ces principes ont d'ailleurs été depuis adoptés dans bon nombre de produits. Mentionnons l'OpenText de TNI ou Xemacs, qui proposent ces visions globales et structurées d'une application donnée.

Evoquons enfin les services à caractère ciblé qu'offre Smalltalk : il permet, par exemple, de connaître toutes les classes implémentant une méthode donnée, ou toutes les méthodes référençant une méthode, une classe ou une variable globale donnée.

Si cette description sommaire de Smalltalk visait, pour l'essentiel, à introduire les notions utilisées dans CatchIt, elle en a aussi montré, outre sa puissance, sa parfaite adéquation aux développements dits exploratoires. CatchIt entre dans ce cadre et a largement bénéficié du choix de cet environnement.

BIBLIOGRAPHIE

[Allen 71] : Allen, Lunenfeld, Alexander Driver Information needs, Highway Research Record, 366, 1971

[Amalberti 89] : Amalberti, R. La médecine aérospatiale face aux défis des futures interfaces homme-machine, Médecine Aéronautique et Spatiale, Tome XXVIII, N°110, 1989, pp 111-120

[André 96] : André, P. L'utilisation de Smalltalk dans trois Caisses régionales du Crédit Agricole, L'OBJET : Logiciel, bases de données, réseaux, Vol. 2, n° 5, pp 26-27

[Atwood 95] : Atwood, M.E., Burns, B., Girgensohn, A., Lee, A., Turner, T., Zimmermann, B. Prototyping Considered Dangerous, Human Computer Interaction, Interact'95, Nordby, K., Helmersen, P.H., Gilmore, D.J., Arnesen, S.A. (Eds), Chapman and Hall Press, 1995, pp 179-184

[Aublet-Cuvelier 96] : Aublet-Cuvelier, L., Carraux, E., Coutaz, J., Nigay, L., Portolan, N., Salber, D., Zanello, M.L. NEIMO, un laboratoire d'utilisabilité numérique : Leçons de l'expérience, ERGO IA 96, pp 149-160

[Badouel 96] : Badouel, D., Cointe, P. JAVA : quelque part entre Smalltalk et C++, L'OBJET : Logiciel, bases de données, réseaux, Vol. 2, N° 5, 1996, pp 15-22

[Balbo 94] : Balbo, S. Evaluation Ergonomique des Interfaces Utilisateur : Un Pas Vers l'Automatisation, Thèse de l'Université Joseph Fourier de Grenoble, Septembre 1994, 287 pages

[Bares 96] : Bares, M., Pastor, D. Principe d'un moteur d'interaction multimodale pour systèmes embarqués, Génie Logiciel, N°40, Juin 1996, pp 31-38

[Barthe 95] : Barthe, M. Ergonomie des logiciels, une nouvelle approche des méthodologies d'informatisation, MASSON 1995 Paris, 191 pages

[Barthet 88] : Barthet, M.F. Logiciels interactifs et ergonomie, modèles et méthodes de conception, Dunod Informatique, 1988, 219 pages

[Barthet 94] : Barthet, M.F., Liberati, V., Ponamale, M. ERGOVAL Outil d'aide à l'évaluation ergonomique des logiciels, ERGO IA'94, 1994, pp 482-490

[Bastien 93] : Bastien, J.M., Scapin, D. Ergonomic Criteria for the Evaluation of Human-Computer Interfaces, Rapport technique INRIA, N°156, Juin 1993, pp 79

[Bennet 89] : Bennett, W.E., Boies, S.J., Gould, J.D., Greene, S.L., Wiecha, C. Transformations on a dialog tree : Rule-based mapping of content to style, 2nd Symposium on UIST'89, Virginie, USA, 13-15 November, 1989, pp 67-75

[Berger 92] : Berger, P. C++ en route vers le monopole, Le Monde Informatique, Spécial Approche Objet, 15 Juin 1992, p 32

[Bier 93] : Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T. Toolglass and magic lenses : the see-through interface, SIGGRAPH'93, Conference proceedings on Computer graphics, 1993, pp 73-80

[Bodart 95a]: Bodart, F., Vanderdonckt, J. SIERRA ou les prémices d'un environnement complet pour la gestion des règles ergonomiques, IHM'95, 11-13 Octobre 1995, Toulouse, France, pp 185-192

[Bodart 95b] : Bodart, F., Hennebert, A.M, Leheureux, J.M., Provot, I., Sacré, B., Vanderdonckt, J. Towards a Systematic Building of Software Architecture : the TRIDENT Methodological Guide, DSVIS'95, Proceedings of the Eurographics Workshop, Palanque & Bastide (eds), 5-7 June 1995, Toulouse, France, pp 262-278

[Boies 88] : Boies, S.J., Bennett, W.E., Gould, J.D., Greene, S.L., Wiecha, C. The Interactive Transaction System (ITS) : Tools for Application Development, Tehnical Report, IBM T.J. Watson Research Center , New-York, 1988, pp 79

[Booch 94] : Booch, G. Software Engineering with ADA, 3rd Edition, The Benjamin/Cummings Publishing Company, 1994

[Boy 91] : Boy, G. Intelligent Assistant Systems, Knowledge-Based Systems, Volume 6, Academic Press, 1991, pp 361

[Boudigou 95] : Boudigou, D., Mével, J.P. L'IHM dans les systèmes embarqués, 4ièmes Journées Thématiques, Les Interfaces Hommes-Machines, 25-26 Octobre 1995, Brest

[Brisson 95] : Brisson, G., André, J. PROSPECT, Analyse et spécification de l'interface utilisateur d'un système interactif, Electricité de France, Direction des Etudes et Recherches, Service Informatique et Mathématiques Appliquées, HI-52/94/034, 16 Mai 1995

[Byrne 94] : Byrne, M.D., Wood, S.D., Sukaviriya, P.N., Foley, J.D., Kieras, D.E. Automating Interface Evaluation, CHI'94, Boston, Massachusetts, April 24-28, 1994, pp 232-237

[Cahier 92] : Cahier, J.P. L'objet modifie le travail du développeur, Le Monde Informatique, Spécial Approche Objet, 15 Juin 1992, p 39

[Calvar 92] : Calvar, J.O., Goubier, T. Auto-Apprentissage et Interfaces Homme-Machine, Stage ENSTB réalisé à THOMSON-CSF Radars et Contre-Mesures, Brest, 1992

[Calvary 96] : Calvary, G., Coutaz, J., Nigay, L., Salber, D. PAC^{+3} : un modèle d'architecture générique pour systèmes multi-utilisateurs, IHM'96, pp 125-130, 1996

[Calvary 97a] : Calvary, G., Coutaz, J., Nigay, L. From Single-User Architectural Design to PAC* : a Generic Software Architecture Model for CSCW, CHI'97, Atlanta, Georgia, pp 242-249, 22-27 March 1997

[Calvary 97b] : Calvary, G., Mével, J.P., Voirin, J.L. Spécificités des Interfaces Homme-Machine Tactiques Embarquées - Leçons de l'expérience, L'Interacteur, N° 4, Septembre 1997, p 6

[Calvary 97c] : Calvary, G., Coutaz, J., Voirin, J.L. Vers une Evaluation Automatique et Transparente de l'Adéquation Homme-Machine, IHM'97, Poitiers, Septembre 1997, pp 55-62

[Cannon 74] : Cannon, W.B. The emergency function of the adrenal medulla in pain and the major emotions, Am. J. Physiol., 1974, 33, pp 356-372

[Card 83] : Card, S.K., Moran, T.P., Newell, A. The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983

[Carraux 96] : Carraux, E. Support informatique pour l'analyse de l'interaction homme-machine, Rapport de DEA Sciences Cognitives CLIPS-IMAG Equipe IIHM, 25 juin 1996

[Cash 95] : Cash, J. L'émergence de la mentalité objet, Informatiques magazine, N° 12, Décembre 1995, p 64

[Chambost 96] : Chambost, G. Accidents d'avions : La chasse aux « facteurs humains », Science & Vie, Numéro 945, Juin 1996, pp 174-177

[Chang 97] : Chang, E.J., Dillon, T.S. Les méthodes de génie logiciel pour la conception des interfaces utilisateurs, Un panorama critique et une proposition d'approche, Génie Logiciel, N°43, Mars 1997, pp 2-23

[Charles 96] : Charles, J., Mission Cassiopée, Le facteur humain dans les vols habités, Qualité Espace Magazine, Numéro 29, N° ISSN : 1556 - 6558, Décembre 1996, pp 12-20

[Chevaleraud 84] : Chevaleraud, J.P. Travail sur terminaux d'ordinateurs, Exigences visuelles : mythe ou réalité ?, Le concours médical, 14-01-84, pp 109-113

[Clancy 93] : Clancy, J., McVicar, A. Sujectivity of stress, British Journal of Nursing, Volume 2, Numéro 8, 1993, pp 410-417

[Coad 93] : Coad, P., Yourdon, Y. Analyse Orientée Objets, Collection Méthodes Informatiques et Pratique des Systèmes, Masson, Prentice Hall Publ., 1993

[Coutaz 90] : Coutaz, J. Interfaces homme-ordinateur, Conception et réalisation, Dunod informatique, 1990

[Coutaz 94] : Coutaz, J., Balbo, S. Evaluation des interfaces utilisateur : Taxonomie et recommandations, IHM'94, 8-9 décembre, 1994, Lille, pp 211-218

[Coutaz 95a] : Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R., Four Easy Pieces for Assessing the Usability of Multimodal Interaction : the CARE Properties, INTERACT'95, Nordby, K., Helmersen, P.H., Gilmore, D., Arnesen, S. (eds), Chapman & Hall, 1995

[Coutaz 95b] : Coutaz, J. Is Usability Testing for Specialists Only ?, Summary of a workshop on usability testing, Proceedings of Engineering Human Computer Interaction, EHCI'95, Grand Targhee, USA, August 1995, Chapman&Hall Publ., pp 348-357

[Coutaz 98] : Coutaz, J. Le futur ne manque pas d'avenir, ERGOIA'98, 1998, à paraître

[Desurvire 92] : Desurvire, H.W., Kondziela, J.M., Atwood, M.E. What is Gained and Lost when using Evaluation Methods other than Empirical Testing, People and Computers VII, Procs of the HCI'92 Conference, September 1992, Monk, A., Diaper, D., Harrison, M.D. Eds, pp 89-102

[Détienne 95a] : Détienne, F., Rist, R. Introduction to Special Issue on Empirical Studies of Object-Oriented Design, Human-Computer Interaction, 1995, Volume 10, pp 121-128

[Détienne 95b] : Détienne, F. Design Strategies and Knowledge in Object-Oriented Programming : Effects of Experience, Human-Computer Interaction, Volume 10, Numbers 2&3, Lawrence Erlbaum, 1995, pp 129-170

[Diadem 94] : DIADEM, Guide de développement, Phases système, Spécification du logiciel, Guide de rédaction d'un livre des normes, Recueil de règles générales, Conception du logiciel, Modèle de conception, Exemple, 8 volumes, THOMSON-CSF Division Réseaux de Communication et Systèmes de Commandement, 1994

[Dix 93] : Dix, A., Finlay, J., Abowd, G., Beale, R. Human-Computer Interaction, Prentice Hall International (UK) Limited 1993, 570 pages

[Doppler 94] : Doppler, F. La charge mentale du travail contemporain, Homme & Travail - Santé, Numéro 7, Octobre-Novembre 1994, p 3

[De Rosis 94] : De Rosis, F., Cozza, M.T., De Carolis, B., Errore, S., Pizzutilo, S., De Zegher, I. Adaptative Interaction With Knowledge-Based Systems, AVI'94, Bari, Italy, June 1-4, 1994

[Elwert 95] : Elwert, T., Schlungbaum, E. Modelling and Generation of Graphical User Interfaces in the TADEUS Approach, DSVIS'95, Proceedings of the Eurographics Workshop, Palanque & Bastide (eds), 5-7 June 1995, Toulouse, France, pp 193-208

[Farenc 96] : Farenc, C. Un Outil d'Aide à l'Evaluation Ergonomique des Interfaces Homme-Machine, IHM'96, Septembre 1996, Grenoble, pp 147-148

[Farenc 97] : Farenc, C. ERGOVAL : une méthode de structuration des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques, Thèse de l'Université Toulouse I, Janvier 1997, 204 pages

[Fischer 88] : Fischer, G., Morch, A. CRACK : A critiquing approach to cooperative kitchen design, Proceedings of the ACM International Conference on Intelligent Tutoring Systems, May 1988, pp 176-185

[Fischer 90] : Fischer, G., Lemke, C., Mastaglio, T., Morch, A.I. Using Critics to Empower Users, CHI'90, April, 1990, pp 337-347

[Fischer 93] : Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., Sumner, T. Embedding Computer-based critics in the contexts of design, InterCHI'93, 24-29 April 1993, pp 157-164

[Flanagan 97] : Flanagan, D. JAVA in a Nutshell, Second Edition, A Desktop Quick Reference, 1997, O' Reilly (Eds), pp 610

[Foley 88a] : Foley, J., Gibbs, C., Kim, W.C., Kovacevic, S. A knowledge-based user interface management system, CHI'88, pp 67-72

[Foley 88b] : Foley, J., Kim, W.C., Kovacevic, S., Murray, K. The User Interface Design Environment, GWU-IIST-88-04, Department of Electrical Engineering and Computer Science, Scool of Engineering and Applied Science, The George Washington University, Washington, D.C. 20052, January 88

[Gamboa-Rodriguez 98] : Gamboa-Rodriguez, F. Spécification et implémentation d'ALACIE, Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques, Thèse de l'Université Paris-Sud, UFR Scientifique d'Orsay, Octobre 1998, pp 265

[Gamma 94] : Gamma, E., Helm, R., Johnson, R., Vlissides, J. Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994

[Geib 98] : Geib, J.M., Gransart, C., Merle, P. CORBA : Des concepts à la pratique, Massons Editeur, 1998

[Gilmore 95] : Gilmore, D.J. Interface Design : Have we got it wrong ?, Human Computer Interaction, Interact'95, Nordby, K., Helmersen, P.H., Gilmore, D.J., Arnesen, S.A. (Eds), Chapman and Hall Press, 1995, pp 173-178

[Giry 95] : Giry, P. Facteurs déterminants de la vigilance du personnel opérateur et décideur à bord des bâtiments de combat, Institut de Médecine Navale du Service de Santé des Armées, Janvier 1995, pp 27

[Goldberg 84] : Goldberg, A. Smalltalk-80, The Interactive Programming Environment, Addison-Wesley, 1984

[Goodman 96] : Goodman, S.E. War, Information Technologies and International Asymmetries, Communications of the ACM, December 1996, Volume 39, Number 12, pp 11- 15

[Gorny 95] : Gorny, P. EXPOSE : an HCI-Counseling Tool for User Interface Designers, CHI'94 Special Interest Group, Trip Report in SIGCHI, Volume 27, Number 2, April 1995, pp 48-49

[Gould 88] : Gould, J.D. How to Design Usable Systems, Handbook of Human-Computer Interaction, M. Helander ed., Elsevier Science B.V., 1988, pp 757-789

[Hammontree 92] : Hammontree, M.L., Hendrickson, J.J., Hensley, B.W. Integrated data capture and analysis tools for research and testing on graphical user interfaces, CHI'92, Monterey, 3-7 May 1992, pp 431-432

[Hartson 90] : Hartson, H.R., Siochi, A.C., Hix, D. The UAN : A user-oriented representation for direct manipulation interface design, ACM Transactions on Information Systems, 1990, vol. 8, n°3, pp 181-203

[Harris 91] : Harris, D. Factors affecting mood and performance in cabin crew, Médecine Aéronautique et Spatiale, Tome XXX, Numéro 118, 1991, pp 162-165

[Hix 93] : Hix, D., Hartson, H.R., Developing User Interfaces, Ensuring Usability Through Product & Process, John Wiley & Sons, Inc., 1993, pp 381

[Holyer 93] : Holyer, A. Methods For Evaluating User Interfaces, Cognitive Science Research Paper No. 301, November 10, 1993

[IFIP 96] : IFIP Book, Design Principles for Interactive Software, Gram, C. and Cockton, G. (eds), Chapman & Hall, 1996

[INRS] : Les écrans de visualisation, Guide méthodologique pour le médecin du travail, INRS

[INT'L 95] : Human Factors INT'L's slogans, CHI'95, Denver, Colorado, USA, 1995

[Ishii 94] : Ishii, H. Seamless Media Design, CSCW'94 Technical Video Program, Issue 106, Future Visions, N° 10, 1994

[Jambon 96] : Jambon, F. Erreurs et interruptions du point de vue de l'ingénierie de l'interaction homme-machine, Thèse de l'Université Joseph Fourier de Grenoble, Décembre 1996, 225 pages

[Jambon 97] : Jambon, F. La responsabilité du concepteur d'une interface homme-machine face aux erreurs des utilisateurs : exemple de l'accident de l'Airbus A320 survenu au mont Sainte-Odile, L'Interacteur, Numéro 2, Avril 1997, p 6

[Jeanne 92] : Jeanne, F. Rendons à l'objet ce qui lui appartient, Le Monde Informatique, Spécial Approche Objet, 15 Juin 1992, p 32

[Jiang 92] : Jiang, J., Murphy, E.D., Bailin, S.C., Truszkowski, W.F., Szczur, M.R., Prototyping a Knowledge-based compliance checker for User-Interface Evaluation on Motif development environments, Motif'92 : Second Annual International Motif Users Meeting, Bethesda, MD : Open Systems, pp 258-268, 1992

[Johnson 91] : Johnson, H., Johnson, P., Task Knowledge Structures : Psychological basis and integration into system design, Acta Psychologica, 1991, pp 3-26

[Johnson 93] : Johnson, P., Wilson, S., Markopoulos, P., Pycok, J. ADEPT - Advanced Design Environment for Prototyping with Task Models, Proceedings of InterCHI'93, Amsterdam, The Netherlands, 24-29 April 1993, pp 56-57

[Kelly 92] : Kelly, C., Colgan, L., User Modelling and User Interface Design, HCI'92, September 1992, pp 227-238

[Keravel 97] : Kéravel, F. Les « alarmes », Des précautions d'usage indispensables, Qualité Espace Magazine, Numéro 30, N°ISSN : 1556-6558, Juin 1997, pp 40-47

[Kernighan 90] : Kernighan, B.W., Ritchie, D.M. Le Langage C, Manuels Informatiques Masson, pp 218, 1990

[Kieras 85] : Kieras, D.E., Polson, P.G. An Approach to the Formal Analysis of User Complexity, International Journal of Man-Machine Studies, 22, 1985, pp 365-394

[Kieras 95] : Kieras, D.E., Wood, S.D., Abotel, K., Hornof, A. GLEAN : A Computer-Based Tool for Rapid GOMS Model Usability Evaluation of User Interface Designs, In procs UIST'95, Pittsburgh, Pennsylvania, November 14-17, 1995, pp 91-100

[Kim 93] : Kim, W.C., Foley, J.D., Providing High-level Control and Expert Assistance in the User Interface Presentation Design, Ashlung, S., Mullet, K., Henderson, A., Hollnagel, E., White, T. (eds), INTERCHI'93, 1993, pp 430-437

[Kolski 89] : Kolski, C. Contribution à l'ergonomie de conception des interfaces graphiques homme-machine dans les procédés industriels : application au système expert SYNOP, Thèse de l'Université de Valenciennes et du Hainaut-Cambrésis, Janvier 1989

[Lagarde 91] : Lagarde, D., Batejat, D., Cizeau, J., Anton, G., Chalon, S., Pradella, S. Evaluation des états de vigilance chez le sujet humain, Médecine Aéronautique et Spatiale, Tome XXX, Numéro 117, 1991, pp 111-120

[Larousse 82] : Grand Dictionnaire Encyclopédique Larousse, 15 volumes, 1982

[Larousse 94] : Petit Larousse Illustré, 1994

[Lemke 90] : Lemke, A., Fischer, G. A cooperative problem solving system for user interface design, Proceedings of the Eighth National Conference on Artificial Intelligence, 1990, pp 479-484

[Leroy 92] : Leroy, C. L'objet désormais à la portée de l'entreprise, Le Monde Informatique, Spécial Approche Objet, 15 Juin 1992, p 31

[Lewis 90] : Lewis, C., Polson, P., Wharton, C., Rieman, J. Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces, Proceedings of the CHI'91 Conference on Computer-Human Interaction, Seattle, ACM, New-York, pp 235-242, April 1990

[Liberati 95] : Liberati, V., Farenc, C. Modélisation des connaissances d'ERGOVAL, un outil d'évaluation ergonomique des interfaces, IHM'95, Toulouse, Octobre 1995, pp 193-200

[Lidouren 90] : Lidouren, J. Les détecteurs d'alerte aéroportés, Cours de Contre-Mesures, THOMSON-CSF Radars et Contre-Mesures, Novembre 1990

[Logeay 90] : Logeay, P., Gadbois, C. Cadres théoriques généraux de la psychopathologie du travail, Psychologie et psychopathologie du travail, Documents pour le médecin du travail, Numéro 41, 1er trimestre 1990, INRS, pp 37-43

[Lonczewski 96] : Lonczewski, F., Schreiber, S. The FUSE-System : an Integrated User Interface Design Environment, CADUI'96, J. Vanderdonckt (eds), 1996, pp 37-56

[Löwgren 90] : Löwgren, J. A Knowledge-Based Tool for User Interface Evaluation and its Integration in a UIMS, Interact'90, 1990 , pp 395-400

[Luo 93] : Luo, P., Szekely, P., Neches, R. Management of interface design in HUMANOID, INTERCHI'93, Amsterdam, The Netherlands, 24-29 April 1993, pp 107-114

[Mallion 82] : Mallion, J.M. Stress et Tension Artérielle, Editions Médicales SPECIA, 1982

[Märtin 96] : Märtin, C. Software Life Cycle Automation for Interactive Applications : The AME Design Environment, CADUI'96, J. Vanderdonckt (eds), 1996, pp 57-73

[McCall 77] : McCall, J. Factors in Software Quality, General Electric Ed., 1977

[Mével 87]: Mével, A., Guéguen, T. Smalltalk-80, Eyrolles eds, 1987, pp 240

[Myers 86] : Myers, B.A., Buxton, W.A.S. Creating Highly Interactive and Graphical User Interfaces by Demonstration, SIGGRAPH'86, 18-22 August 1986, Dallas TX, Computer Graphics, 20, pp 249-258

[Myers 92] : Myers, B.A., Rosson, M.B. Survey on user interface programming, Proceedings of the CHI'92 Conference, Monterey, California, 3-7 May 1992, pp 195-202

[Neal 83] : Neal, A.S., Simons, R.M. Playback : a method for evaluating the usability of software and its documentation, CHI'83, December 1983, pp 78-82

[Neboit 94] : Neboit, M., Perception, anticipation et conduite automobile, Le Travail Humain, Tome 37, n°I/1974 , pp 53-72

[Nielsen 93] : Nielsen, J. Usability Engineering, Academic Press Professional, 1993, p 362

[Nitsche-Ruhland 95] : Nitsche-Ruhland, D., Zimmermann, G. CritiGUI-Knowledge-based Support fort the Interface Design Process in Smalltalk, EWHCI'95, Moscow, Russia, July 1995

[Normand 92] : Normand, V. Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation, Thèse de l'Université Joseph Fourier-Grenoble I, Spécialité Informatique, Avril 1992, 258 pages

[Orga 94] : ORGA S.A.R.L. (Systèmes, Analyse, Risques, Logistique), Cours de Guerre Electronique, Janvier 1994

[OTAN 96] : Revue de l'OTAN, Le Conseil de l'OTAN entérine le premier programme de normalisation OTAN, Edition n°5, Septembre 1996, Vol. 44, pp 28

[Paty 95] : Paty, J. Cerveau et connaissance : introduction aux sciences cognitives, Cours de neuropsychologie, Université de Bordeaux II, 1995, 100 pages

[Poltrock 95] : Poltrock, S., Grudin, J. Groupware and Workflow : A Survey of Systems and Behavioral Issues, Conference on Human Factors in Computing Systems, may 7-11 1995, Denver Colorado USA

[Portejoie 97] : Portejoie, B. Ariane 5 et fiabilité humaine : Opération de consolidation de la production à la SEP, Qualité Espace Magazine, Numéro 30, N°ISSN : 1556-6558, Juin 1997, pp 49-51

[Preece 94] : Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T. Human-Computer Interaction, Addison-Wesley, 1994

[Pressman 87] : Pressman, R.S. Software engineering, a Practioner's Approach, McGraw-Hill Book Company, 1987

[Puerta 96] : Puerta, A. The MECANO Project : Comprehensive and Integrated Support for Model-Based Interface Development, CADUI'96, J. Vanderdonckt (eds), 1996, pp 19-35

[Puerta 97] : Puerta, A.R. A Model-Based Interface Development Environment, IEEE Software, July/August 1997, pp 40-47

[Rasmussen 86] : Rasmussen, J. Information Processing and Human-Machine Interaction : An approach to Cognitive Engineering, North-Holland, 1986

[Ravden 89] : Ravden, S., Johnson, G. Evaluating Usability of Human-Computer Interfaces, A practical method, Ellis Horwood Books in Information Technology, 1989, pp 126

[Reason 93] : Reason, J. L'erreur humaine, Le travail humain, Collection dirigée par Jean-Michel Hoc, Presses universitaires de France, 1993

[Reiterer 93] :Reiterer, H., The Development of Design Aid Tools for Human Factor Based User Interface Design, Proceedings of IEEE International Conference on System, Man and Cybernetics, Volume 4, IEEE, 1993, pp 361-366

[Rémy 96] : Rémy, C. Les chemins de fer belges adoptent l'objet, 01 Informatique, N° 1418, 6 Septembre 1996, p 33

[Renaudie 89] : Renaudie, J.F. L'interface homme-machine, facteur clé de la certification des avions de transport civils, Médecine Aéronautique et Spatiale, Tome XXVIII, N°110, 1989, pp 149-158

[Rettig 93] : Rettig, M., Simons, G. A project planning and development process for small teams, Communications of the ACM, Volume 36, Number 10, October 1993, pp 45-52

[Rumbaugh 95] : Rumbaugh, J. et al. Modélisation et conception orientées objet, Masson Paris Prentice Hall - London, 1995, 516 pages

[Rumbaugh 97] : Rumbaugh, J., Jacobson, I., Booch, G. Unified Modeling Language Reference Manual, Addison Wesley (Eds), 1997

[Sagory 95] : Sagory, P. Kronos, un logiciel d'aide à l'analyse ergonomique, Le Mensuel de l'ANACT, Avril 1995, p 22

[Salber 95] : Salber, D. De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs, L'exemple de la communication homme-homme médiatisée, Thèse de l'Université Joseph Fourier-Grenoble I, Spécialité Informatique, 8 septembre 1995, 303 pages

[Scapin 89] : Scapin, D.L., Pierret-Golbreich, C. MAD : Une méthode analytique de description des tâches, IHM'89 , Sophia-Antipolis, Mai 1989, pp 131-148

[Scapin 90] : Scapin, D.L. Des critères ergonomiques pour l'évaluation et la conception d'interfaces utilisateurs, Acte du XXVI Congrès de la SELF, 3-5 Octobre, Montréal, Canada, 1990

[Scavénius 92] : Scavénius, R. Premières récoltes fructueuses pour les utilisateurs, Le Monde Informatique, Spécial Approche Objet, 15 Juin 1992, pp 38-39

[Schlungbaum 96] : Schlungbaum, E., Elwert, T. Automatic User Interface Generation from Declarative Models, CADUI'96, J. Vanderdonckt (eds), 1996, pp 3-17

[Schreiber 94] : Schreiber, S. The BOSS-System : Coupling Visual Programming with Model-Based Interface Design, DSVIS'94, Eurographics Workshop, 1994, pp 41-59

[Sears 95] : Sears, A. AIDE : A step toward metric-based interface development tools, UIST'95, November 14-17, 1995, pp 101-110

[Sebillotte 94] : Sebillotte, S et al. , Note de recherche concernant le formalisme MAD, Projet de Psychologie Ergonomique pour l'Informatique, INRIA Rocquencourt, Novembre 1994, pp 31

[Selye 62] : Selye, H. Le stress de la vie (Traduction Verdun P.), Gallimard éd., Paris, 1962

[Senach 90] : Senach, B. Evaluation de l'ergonomie des interfaces homme-machine : du prototypage aux systèmes experts, ERGO IA'90, 1990, pp 85-106

[Shlaer 92] : Shlaer, S., Mellor, S. Modeling the World in states, Yourdon Press, Computing Series, 1992

[Siochi 91] : Siochi, A.C., Hix, D. A study of computer-supported user interface evaluation using maximal repeating pattern analysis, CHI'91, New Orleans, May, 1991, ACM New-York, pp 301-305

[Smelick 79] : Smelick, G., Adrenocortical feedback control of pituitary-adrenal activity, Pituitary adrenal and the brain, Dewied, D., Jaw, M., Weijne, N., Progress Brain Research 32, Elsevier ed., Amsterdam, 1979, pp 21-23

[Smith 86] : Smith, S.L., Mosier, J.N. A design evaluation checklist for user-system interface software, Report #MTR-9480 EDS_TR_84-358, The MITRE Corporation, Bedford, MA, 1982

[Spérandio 87] : Spérandio, J.C. Introduction à l'ergonomie des logiciels, Le Travail et l'écran, Dossier documentaire, Agence Nationale pour l'Amélioration des Conditions de Travail, 1987, pp 23-29

[Spérandio 92] : Spérandio, J.C. La charge mentale des personnels travaillant sur écran, Ecran et Vision, Paris-Maison de la Chimie, 8-9 décembre 1992, pp 77-85

[Stroustrup 92] : Stroustrup, B. Le langage C++, 2ème édition, AT&T Bell Laboratories, Addison-Wesley, Publishing Company, Inc. 1992, 667 pages

[Sukaviriya 90] : Sukaviriya, P., Foley, J.D. Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help, UIST'90, 1990, pp 152-166

[Sukaviriya 94] : Sukaviriya, P., Kovacevic, S., Foley, J.D., Meyers, B.A., Olsen, D.R., Schneider-Hufschmidt, M. Model-Based Interfaces, What are They and Why Should We Care ?, UIST'94, November 2-4, 1994, pp 133-135

[Sumner 97] : Sumner, T., Bonnardel, N., Kallak, B.H. The Cognitive Ergonomics of Knowledge-Based Design Support Systems, CHI'97, 22-27 march 1997, Atlanta, Georgia, pp 83-90

[Szczur 94] : Szczur, M. Usability testing - on a budget : a NASA usability test case study, Behaviour & Information Technology, 1994, Vol. 13, N° 1 and 2, pp 106-118

[Szekely 92] : Szekely, P., Luo, P., Neches, R. Facilitating the Exploration of Interface Design Alternatives : The HUMANOID Model of Interface Design, CHI'92, May 3-7, 1992, Monterey, California, pp 507-515

[Szekely 93] : Szekely, P., Luo, P., Neches, R. Beyond interface builders : model-based interface tools, INTERCHI'93, Amsterdam, The Netherlands, 24-29 April 1993, pp 383-390

[Szekely 94] : Szekely, P. User Interface Prototyping : Tools and Techniques, Software Engineering and Human-Computer Interaction, ICSE'94 Workshop on SE-HCI : Joint Research Issues, Lecture Notes in Computer Science 896, Taylor, R.N., Coutaz, J. (Eds), Sorrento, Italy, May 1994, pp 76-92

[Szekely 95] : Szekely, P., Sukaviriya, P., Castells , P., Muthukumarasamy, J., Salcher, E. Declarative interface models for user interface construction tools: the MASTERMIND approach, Engineering for Human-Computer Interaction, Bass, L.J. & Unger, C. Eds, Chapman & Hall, 1995, pp 120-150

[Szekely 96a] : Szekely P., Retrospective and Challenges for Model-Based Interface Development, Design, Specification and Verification of Interactive Systems'96, Proceedings of the Eurographics Workshop in Namur, Belgium, June 5-7 1996, F. Bodard, J. Vanderdonckt (eds)

[Szekely 96b] : Szekely P., Retrospective and Challenges for Model-Based Interface Development, Computer-Aided Design of User Interfaces, Proceedings of CADUI'96, J. Vanderdonckt (eds), Presses Universitaires de Namur, pp xxi-xliv

[Tani 94] : Tani, M., Horita, M., Yamaashi, K., Tanikoshi, K., Futakawa, M. Courtyard : Integrating A Shared Large Screen and Individual Screens, CSCW'94 Technical Video Program, Issue 106, Applications and Methodologies, N° 4, 1994

[Tarby 93] : Tarby, J.C. Gestion Automatique du Dialogue Homme-Machine à partir de Spécifications Conceptuelles, Thèse de l'Université Toulouse I, 20 septembre 1993, 257 pages

[Tardif 96] : Tardif, C. L'Europe s'attaque aux problèmes cognitifs, Air & Cosmos/Aviation International, Numéro 1581, 4 octobre 1996, p 35

[Turk 90] : Turk, M., Pentland, A. Eigenfaces for recognition, Journal of Cognitive Neuroscience, septembre 1990

[UIMS 92] : The UIMS Workshop Tool Developers : A Metamodel for the Runtime Architecture of an Interactive System, SIGCHI Bulletin, 24, 1, January 1992, pp 32-37

[Universalis 93] : Encyclopédia Universalis Corpus, 1993

[Valentin 93] : Valentin, A., Vallery, G., Lucongsang, R. L'Evaluation Ergonomique des Logiciels, Une démarche itérative de conception, Editions de l'ANACT, Collection Outils et Méthodes, 151 pages, 1993

[Vander 85] : Vander, A.J., Sherman, J.H., Luciano, D.S. Physiologie humaine, McGraw-Hill, Editeurs-Montréal, 1985

[Vanderdonckt 93]: Vanderdonckt, J., Bodart, F. Encapsulating Knowledge For Intelligent Automatic Interaction Objects Selection, InterCHI'93, 24-29 April 1993, pp 424-429

[Vanderdonckt 94]: Vanderdonckt, J. Automatic Generation of a User Interface for Highly Interactive Business-oriented Applications, Conference Companion of CHI'94, Boston, Massachusetts, USA, 24-28 April 1994, pp 123-124

[Waern 95] : Waern, Y., Hägglund, S., Ramberg, R., Rankin, I., Harrius, J. Computational advice and explanations - behavioural and computational aspects, Interact'95, pp 203-206

[Welch 97] : Welch, B. Practical Programming in Tcl and Tk, 2nd Edition, Prentice Hall, 1997

[Wiecha 90] : Wiecha, C., Bennett, W., Boies, S., Gould, J., Greene, S., ITS : A Tool for Rapidly Developing Interactive Applications, ACM Transactions on Information Systems, Vol. 8, No 3, July 1990, pp 204-236

[Wiener 89] : Wiener, E. Erreur et prévention de l'erreur dans les avions de technologie avancée, Médecine Aéronautique et Spatiale, Tome XXVIII, N°110, 1989, pp 121-125

[Wilson 95] : Wilson, S., Johnson, P. Empowering users in a task-based approach to design, DIS'95, Michigan, USA, August 23-25, 1995, pp 25-31

[Wisner 89] : Wisner, M. Ergonomie et Neurophysiologie du Travail, Analyse de la situation de travail, Méthodes et techniques, Conservatoire National des Arts et Métriers, Cours B3, 1989-1990

[Wisner 90] : Wisner, A., Denoeud, B., Millanvoye, M. Ergonomie et Neurophysiologie du Travail, Neurophysiologie du travail, Conservatoire National des Arts et Métriers, Cours B3, 1990-1991

[Wright 73] : Wright, S. Physiologie Appliquée à la Médecine, Flammarion Médecine-Sciences, 12ème édition, 1973

[X 35-103] : Norme Expérimentale X 35-103, Principes d'ergonomie visuelle applicables à l'éclairage des lieux de travail, Juin 1980

GLOSSAIRE

ADEPT : Advanced Design Environment for Prototyping with Task Models

AIDE : semi-Automated Interface Designer and Evaluator

ARO : Aire de Renseignements Opérateur

CatchIt : Critic-based Automatic and Transparent tool for Computer-Human Interaction Testing

CLIPS : Communication Langagière en Interaction Personne-Système

CME : Contre-Mesure Electronique

CoTaS : COncepts -TAsks - Strategies

CriMePro : Critères et Métriques en Proactivité

CriMeRéact : Critères et Métriques en Réactivité

DEGRE : Domain Encapsulation by Genericity Refinement and Evolutivity

DIADEM : Dialog Architecture and Design Method

EMA : Evaluation par Mécanisme Automatique

EXPOSE : Expert system for Phase-Oriented Software Ergonomic counseling

VDDE : Voice Dialog Design Environment

ESPRIT : Examination Space for Proactive and Reactive Innovative Tools

FUSE : Formal User Interface Specification Environment

GE : Guerre Electronique

GL : Génie Logiciel

GLEAN : GOMS Language Evaluation and ANalysis

IHM : Interface Homme-Machine

IIHM : Ingénierie des Interfaces Homme-Machine

ITS : Interactive Transaction System

KRI : Knowledge-based Review of user Interfaces

LSB : Less Significant Byte

MBD : Model-Based user interface Design

MB-IDE : Model-Based Interface Development Environments

Mobi-D : Model-Based Interface Designer

MPE : Mesure de Protection Electronique

MRE : Mesure de Recherche Electronique

MRP : Maximal Repeating Pattern

MSB : Most Significant Byte

OSF : Open Software Foundation

OTAN : Organisation du Traité de l'Atlantique Nord

PATCH : Presentation and Abstraction for Task and Concept Hooks

PROSPECT : Procédé pour SPEcifier un système interactif Centré sur la Tâche

RCC : Réseaux de Communication et Systèmes de Commandement

RCM : Radars et Contre-Mesures

Romi : Robust Human-Machine Interaction

TADEUS : TAsk-based DEvelopment of USer interface Software

UIDE : User Interface Design Environment

UIMS : User Interface Management Systems

USAGE : the UIDE System for semi-Automated GOMS