THÈSE

présentée par

Francis JAMBON

pour obtenir le titre de
DOCTEUR de L'UNIVERSITÉ JOSEPH FOURIER - GRENOBLE 1
(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)
Spécialité : **Informatique**

ERREURS ET INTERRUPTIONS DU POINT DE VUE DE L'INGÉNIERIE DE L'INTERACTION HOMME-MACHINE

Date de soutenance : Jeudi 5 décembre 1996

Composition du jury :

Président :	M. Jean CAELEN
Directeur de thèse :	Mme Joëlle COUTAZ
Rapporteurs :	Mme Marie-France BARTHET
	M. Guy BOY
Examinateurs :	M. Philip D. GRAY
	M. Dominique SCAPIN

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

Université Joseph Fourier - Grenoble 1

Remerciements

Je remercie tout d'abord, Jean Caelen, Joëlle Coutaz, Marie-France Barthet, Guy Boy, Phil Gray, et Dominique Scapin d'avoir accepté de faire partie de mon jury. Je leur en suis très reconnaissant.

Je remercie les membres d'EURISCO où s'est déroulé mon "stage d'été" dont Didier Lafrique, Cheikh Sow, Helen Wilson, Laurent Karsenty, sans oublier Alfred Attipoe et Markus Durtewitz. Je remercie aussi les membres du CENA de Toulouse, dont Stéphane Chatty, François-Régis Colin, Damien Figarol, et Francis Preux pour leur accueil. Parmi les Toulousains, je n'oublie pas Philippe Palanque, Stéphane Sire, et Rémi Bastide qui m'ont indiqués d'intéressantes références, en particulier pour Rémi sur le projet THÉRÈSE. Merci aussi à Philippe Brun du LRI pour les discussions fructueuses que nous avons eues ensemble et pour l'envoi d'un schéma bien utile.

Je remercie également M. Lacour de la SNCF grâce à qui nous avons pu faire le trajet Grenoble-Paris dans la cabine d'un TGV Atlantique, et également mon oncle, George Rampon, ancien contrôleur aérien, et M. Chazal de la DGAC pour m'avoir permis de visiter "avec un ancien" les installations du contrôle aérien de Satolas.

Je remercie aussi les membres de mon équipe qui m'ont permis d'effectuer ma thèse dans un cadre agréable. Citons Joëlle Coutaz, Laurence Nigay, Daniel Salber, Laurent Aublet-Cuvelier, Gaëlle Calvary, et François Bérard. Merci aussi à Philippe Mulhem, François Paradis, Catherine Berrut, et Marie-France Bruandet de l'équipe M'RIM sans oublier Christelle Meunier ni Pascale Gonnot de l'équipe ARCADE qui m'ont soutenu dans les deniers moments de cette thèse, et à tous les autres que j'ai oubliés.

Je loue la patience de l'équipe technique et administrative du LGI puis du CLIPS dont Bernard Cassagne, Bernard Martinet, Yves Chiaramella, Liliane Di-Giacomo, Nicole Hervet, Sylvie Pons, Valérie Heitz, et sans oublier la dernière arrivée, Annie Majastre.

Je n'oublie pas non plus ceux qui sont partis comme Gilles Ambone dont j'ai sacrifié avec Solaris 2.5™ la SparcStation 1™ préférée, Nathalie Lischetti et son MacApp™ adoré, ou Sandrine Balbo dont j'ai honteusement "copié-collé" quelques schémas à partir de son mémoire de thèse.

Merci aussi à Jean-Luc Parouty de L'IMAG pour avoir mis en place la retransmission de ma soutenance sur le "MBone". Ce fut une première à L'IMAG, et une réussite.

Toute ma reconnaissance va à mes parents, à Jean, à Francisque et à Sylvie pour m'avoir soutenu pendant ces années de thèse, et pour avoir eu la gentillesse d'être venus à Grenoble afin d'assister à ma soutenance, et pour avoir aidé à son organisation.

Enfin, je ne remercierais pas mon Macintosh™, mais plutôt les mésanges (à tête bleue) qui viennent manger à la fenêtre de mon bureau, et m'ont permis de patienter lors des "restart" de cet admirable outil qu'est l'ordinateur…

L’homme, `la machine`, *l’environnement*.

AVANT-PROPOS

Innombrables sont aujourd'hui les machines complexes ou sophistiquées mises à notre disposition. Ces machines, intégrant le plus souvent un ou plusieurs automatismes, permettent de décharger l'homme[1] de tâches ennuyeuses, répétitives, ou encore dangereuses. Aujourd'hui l'homme n'est plus obligé d'agir directement sur son environnement, il peut faire agir la machine à sa place. La bonne réalisation d'une tâche dépend alors de la qualité de la communication entre ces deux protagonistes, et c'est à ce moment qu'intervient l'interface homme-machine.

La bonne maîtrise des interfaces homme-machine est l'un des enjeux primordiaux des années à venir. Cette maîtrise doit permettre à l'ensemble de la société l'accès au progrès technique. L'apparition récente du système ShowView[2] destiné à la programmation des magnétoscopes familiaux illustre cette nécessité. Ce système permet maintenant aux utilisateurs "normaux" de programmer leur magnétoscope avec une chance certaine de réussite. Il est intéressant de noter que l'hebdomadaire Télérama entend par "normaux" les utilisateurs non polytechniciens… Malheureusement les exemples de dispositifs de la vie courante non utilisés, ou ne remplissant pas leur fonction en raison d'une interface mal conçue abondent. Le système SOCRATE de réservation informatisé de la SNCF en est l'un des plus connus. Nous ne pouvons pas tous être des spécialistes en systèmes vidéo ou en informatique distribuée. Par contre, nous devons tous pouvoir avec un minimum d'effort programmer notre magnétoscope ou bien prendre un billet de chemin de fer. C'est à l'interface homme-machine de faire la liaison entre le monde parfois obscur de l'électronique et de l'informatique, et l'homme. Sans cela, nous verrons l'apparition d'une partie de notre société être atteinte de "technopathie", c'est-à-dire mise en marge du progrès technique, dont les

---

1 Ici "homme" se réfère au sens latin, c'est-à-dire à l'être humain en général.

2 ‹‹ Le ShowView, (autrement dit programmeur instantané de magnétoscope) ressemble à une petite machine à calculer, et sa manipulation est aussi simple. Vous tapez sur le clavier le code qui figure dans nos programmes à coté de l'émission que vous voulez enregistrer, vous placez votre ShowView sur votre magnétoscope et le tour est joué. ›› [Télérama 1993].

membres n’en profitent pas tout simplement parce qu’ils ne savent pas l’utiliser.

En Occident, l’homme est passé en un peu plus d’un siècle d’une société basée sur l’effort physique à une société basée sur l’effort intellectuel. Le besoin d’énergie musculaire a été progressivement remplacé par la nécessité de traiter l’information. L’énergie, qui était autrefois limitée à la seule force musculaire, est maintenant disponible en grande quantité et à faible coût. La compétence de l’homme n’est plus de produire cette énergie, mais de la diriger convenablement. En corollaire, une quantité d’énergie colossale est maintenant potentiellement disponible sous le contrôle d’un seul être humain. Les conséquences d’une erreur ou d’une malveillance qui étaient autrefois limitées peuvent devenir aujourd’hui dramatiques. La presse grand public s’est fait l’écho d’accidents nucléaires ou aéronautiques aux circonstances parfois mal déterminées où le rôle de l’interface homme-machine a été évoqué. Il est en effet encore difficile de comprendre, et de faire comprendre, comment un homme a pu faire une erreur aux conséquences catastrophiques, alors qu’il avait les possibilités techniques et les connaissances pour l’éviter ! Blâmer un opérateur fautif et l’entraîner à nouveau ne résout souvent rien. Nous devons admettre que, malgré l’expertise et l’entraînement, d’autres erreurs surviendront si la collaboration entre l’homme et sa machine ne se déroule pas en respectant les capacités et les rôles de chacun.

TABLE DES MATIÈRES

INTRODUCTION

1. Un “dessin industriel” pour la spécification des IHMs

Aujourd’hui notre capacité à décrire les interfaces homme-machine souffre d’un retard certain sur notre habileté à les réaliser. Nous concevons en effet des interfaces que nous ne savons pas décrire simplement, si ce n’est par la réunion de leur code informatique et du matériel utilisés. Même sans évoquer les interfaces exotiques comme les gants numériques ou autres casques de réalité virtuelle, la spécification d’interfaces aujourd’hui entrées dans la vie courante est délicate. Citons comme exemple l’interface du Publiphone, le téléphone public de France Télécom™, qui par ses nombreuses possibilités d’interruptions pose déjà de nombreux problèmes de description [Jambon 1995].

Imaginons ce que seraient l’architecture et la mécanique si on se contentait de décrire un bâtiment ou un engrenage par sa seule réalisation finale ? De même, quelle place tiendraient aujourd’hui les bureaux d’étude sans le dessin industriel qui permet la communication entre le concepteur, le maître d’œuvre et l’acheteur ? Les concepteurs d’interfaces homme-machine manquent aujourd’hui de cet outil indispensable qu’est le dessin industriel, adapté à leurs besoins. Le travail présenté dans ce mémoire contribue à répondre à ce besoin d’un “dessin industriel” adapté à la spécification des interfaces homme-machine.

Tout d’abord, nous devons replacer ce “dessin industriel”, que nous nommerons formalisme de spécification, dans le cadre de son utilisation. Celle-ci s’effectue principalement au cours de la phase de conception d’un système interactif. Nous évoquerons les méthodes utilisées au cours de cette phase dans la première section de cette introduction, en mettant l’accent sur les spécialités mises à contribution. Puis, dans la deuxième section, nous détaillerons le contexte et les motivations de ce travail de recherche dirigé par les particularités des systèmes critiques. La troisième section quant à elle, explicite notre démarche de recherche. Enfin, la quatrième section de cette introduction détaille le plan et guide la lecture de ce mémoire.

2. Méthodes de conception des IHMs

L’utilisation des formalismes de spécification s’inscrit dans le contexte plus général des méthodes de conception de l’interface homme-machine d’un

système interactif. La démarche de conception utilisée fait le plus souvent appel à deux spécialités, l'ergonomie cognitive et le génie logiciel, que nous détaillerons respectivement aux paragraphes 2.1 et 2.2 de cette section. Appliquées au processus de conception d'une interface homme-machine, ces spécialités complémentaires s'opposent à la fois par leurs méthodes et leurs résultats. Certains auteurs ont pourtant tenté de les rapprocher, nous évoquerons les résultats de leurs travaux au paragraphe 2.3.

2.1. Ergonomie cognitive

L'ergonomie cognitive[3] intervient tout au long du processus de développement d'une interface homme-machine. En amont de l'étape de conception, elle a alors pour but l'analyse de l'existant et la définition des besoins. Ceux-ci sont alors exprimés de manière plus ou moins structurée et plus ou moins formelle. La classification des approches en ergonomie cognitive proposée par Long et Dowell montre les limites actuelles de cette discipline vis-à-vis de la démarche de conception des interfaces homme-machine [Long & Dowell 1989]. Long et Dowell organisent les approches de l'ergonomie cognitive sur la base du savoir et de la pratique. Le savoir est supposé soutenir la pratique dans la recherche d'une solution au problème posé, ici la conception des interfaces homme-machine. Les auteurs distinguent l'approche artisanale[4], l'approche des sciences appliquées, et l'approche de l'ingénierie :

Pour *l'approche artisanale*, le savoir découle de l'expérience, de la créativité, voire de l'intuition du concepteur. La pratique de l'approche artisanale se rapproche de la programmation exploratoire ou du prototypage rapide : d'emblée, un prototype est mis en œuvre sans nécessairement de spécifications préalables. Les hypothèses de conception sont évaluées, les leçons tirées, de nouvelles solutions proposées et mises en œuvre. Le savoir en *sciences appliquées* repose sur la connaissance scientifique de théories, de modèles, et de lois développés pour expliquer ou prédire un ensemble de phénomènes. La pratique se manifeste par le cycle spécification, mise en œuvre, et évaluation. La non-complétude et la non-garantie des effets du

3 En référence à [Long 1995] "human factors" a été traduit par "ergononie cognitive", "software engineering" par "génie logiciel", et "human computer interaction engineering" par "ingénierie de l'interaction homme machine".

4 En référence à [Long et al. 1989] "craft discipline" a été traduit par "approche artisanale", "applied science discipline" par "approche des sciences appliquées", et "engineering discipline" par "approche de l'ingénierie".

savoir exigent une évaluation avec retours arrière. *L'approche de l'ingénierie* se caractérise quant à elle par l'expression formalisée et opérationnelle du savoir. La pratique se distingue par une étape de spécification complète du système suivie de la mise en œuvre et de l'évaluation.

La classification de Long et Dowell permet non seulement de faire un bilan des approches en ergonomie cognitive, mais aussi d'entrevoir l'évolution et les progrès à venir. Si actuellement l'approche des sciences appliquées paraît être la plus réaliste compte tenu de nos connaissances, l'approche artisanale est encore très usitée en particulier dans le milieu industriel. Les raisons en sont le plus souvent un "savoir" assez limité, mais également un manque de conviction vis-à-vis du rapport coûts sur bénéfices d'une approche plus systématique. A plus long terme, c'est vers l'approche de l'ingénierie qu'il faut se tourner, afin de rapprocher l'ergonomie cognitive du génie logiciel.

2.2. Génie logiciel

Le génie logiciel offre un cadre normalisé et structuré d'aide à la production et à la maintenance de logiciels. L'aspect formel du génie logiciel intervient utilement à divers titres dans le processus de développement, par exemple en vérification de la conformité entre spécifications et réalisation. Introduit dans les années 70 par Royce sous la forme du modèle en cascade [Royce 1970], affiné sous la forme du modèle en V [McDermind & Ripkin 1984], puis du modèle en spirale [Boehm 1988], le génie logiciel dispose aujourd'hui d'un large éventail de modèles de processus de développement :

Le modèle en cascade, dans sa version initiale, comprend quatre étapes successives : ‹ analyse des besoins, › conception du système, fi mise en œuvre et tests unitaires, fl intégration et test du système. Face aux exigences de la pratique, la possibilité d'effectuer des retours arrière a été introduite et l'étape de conception du système a été détaillée en conception du système, conception globale, et conception détaillée. *Le modèle en V* reprend les étapes du modèle en cascade affiné et étend les activités de test qui sont rassemblées dans la pente montante du V. La pente descendante du V ne comprend alors que les étapes de spécification et de mise en œuvre avec, pour chaque étape, un vis-à-vis constitué d'une batterie de tests. *Le modèle en spirale* est plus récent. Il mieux adapté à la réalisation de produits nouveaux où une phase de tâtonnements est nécessaire. Cette phase se manifeste par une série de boucles comprenant comme passage obligé la réalisation d'un prototype, de plus en

plus élaboré et s'approchant du produit final, au fur et à mesure que l'on se rapproche de la dernière boucle. Cette boucle finale est alors comparable au modèle en V car à cette étape les spécifications exactes du système à réaliser ont été élaborées.

À la base de tous les modèles de processus de développement, on retrouve à quelques différences près le cycle *spécification → mise en œuvre → tests*. La présence de ce cycle met en lumière, outre la nécessaire phase de test destinée à éliminer un maximum d'erreurs et de défauts de conception, la nécessité de spécifier avant toute réalisation. Néanmoins cette spécification, supposée ici complète et non ambiguë, est assez différente de celle contenue dans les approches de l'ergonomie cognitive : elle s'intéresse ici principalement au noyau fonctionnel du système, et très peu à son interface avec l'utilisateur. Certes il existe des spécifications externes de l'interface, mais elles prennent le plus souvent la forme de descriptions informelles, incomplètes, et ambiguës. Elles ouvrent ainsi la voie à des interprétations erronées, en particulier lorsque ces spécifications sont utilisées par une personne ou une équipe différente de celle qui les a définies.

2.3. Méthodes mixtes

Parallèlement aux méthodes fondées uniquement sur les modèles du génie logiciel et aux méthodes basées essentiellement sur les approches des sciences cognitives, nous relevons quelques tentatives de rapprochement. Parmi les méthodes qui tentent de construire des ponts entre ergonomie cognitive et génie logiciel, nous décrivons ci-après la méthode MUSE associée à JSD, DIANE associée à MERISE, et le projet NEIMO représentatifs des approches possibles.

Les auteurs de MUSE (Method for USability Engineering), Lim et Long, se sont donnés comme objectif de rapprocher jusqu'à fusionner l'ergonomie cognitive et le génie logiciel afin de permettre à leur union, l'ingénierie de l'interaction homme-machine, de devenir une technique unifiée et efficace [Lim & Long 1994]. La démarche de MUSE est d'intégrer à une méthode de génie logiciel existante, choisie par le concepteur, les étapes de la méthode de l'ergonomie cognitive définie dans MUSE. Ces étapes sont indiquées en grisé sur la figure 0-1. Elles recouvrent une étape d'analyse de l'existant, suivie d'une étape de synthèse de la conception, pour finir par une étape de spécification. Les auteurs de MUSE souhaitent ainsi exploiter la structure, le

cadre normalisé, et l'implantation industrielle des méthodes du génie logiciel pour y introduire celles de l'ergonomie cognitive [Long 1995]. MUSE a été conçue de façon à pouvoir s'intégrer à toute méthode de génie logiciel. Les liens entre MUSE et la méthode choisie, indiqués par des flèches en pointillés sur la figure 0-1, doivent être définis lors de l'initialisation de MUSE avec chaque nouvelle méthode de génie logiciel. Comme illustration, MUSE a été intégrée par ses auteurs à la méthode JSD (Jackson System Developpement) [Jackson 1983].

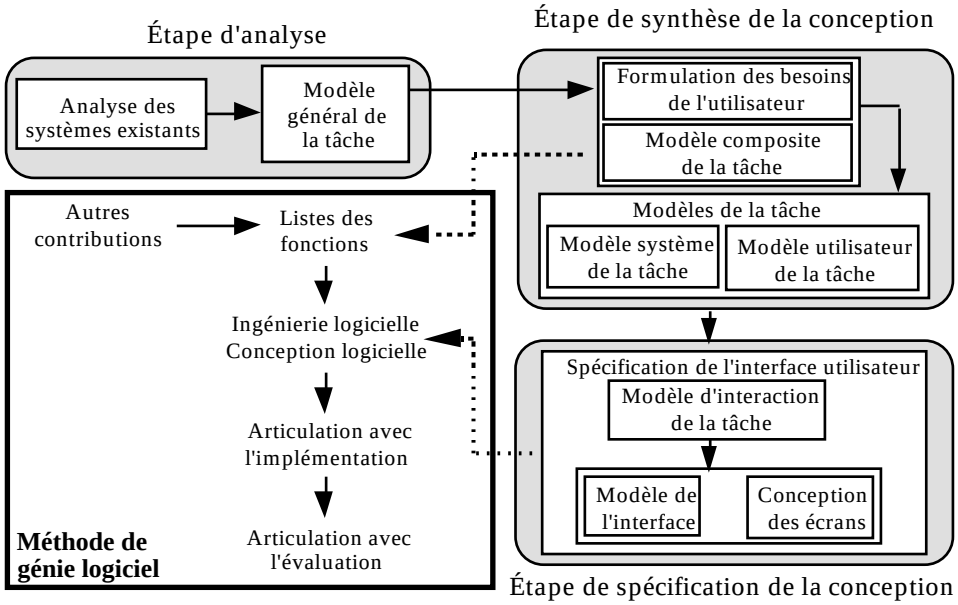

Figure 0-1 : Diagramme schématique de MUSE et ses liaisons avec une méthode de génie logiciel. Figure issue de [Lim et al. 1994] et traduite dans [Balbo 1994].

La méthode DIANE quant à elle doit être vue comme une extension de la méthode MERISE [Barthet 1988]. Cette extension a pour but d'intégrer la prise en compte des facteurs humains dans la méthode de génie logiciel MERISE. DIANE, avec DIANE$^+$ [Tarby 1993] intervient aux principaux niveaux du cycle de vie d'une application informatique, allant de l'étude du système existant jusqu'à la génération d'une partie du code, comme le montre la figure 0-2. Autant MUSE/JSD est destinée aux ergonomes, autant DIANE/MERISE, de part son ancrage dans MERISE, est avant tout dirigée vers les informaticiens [Palanque, Long, et al. 1994]. Elle intègre en particulier des listes de recommandations ergonomiques à leur usage. De plus, DIANE s'oriente, avec DIANE$^+$, vers la génération automatique de code. L'utilisateur de DIANE$^+$ est donc avant tout un informaticien ayant peu de connaissances

en ergonomie. Elle se rapproche en cela du projet NEIMO que nous évoquons maintenant.

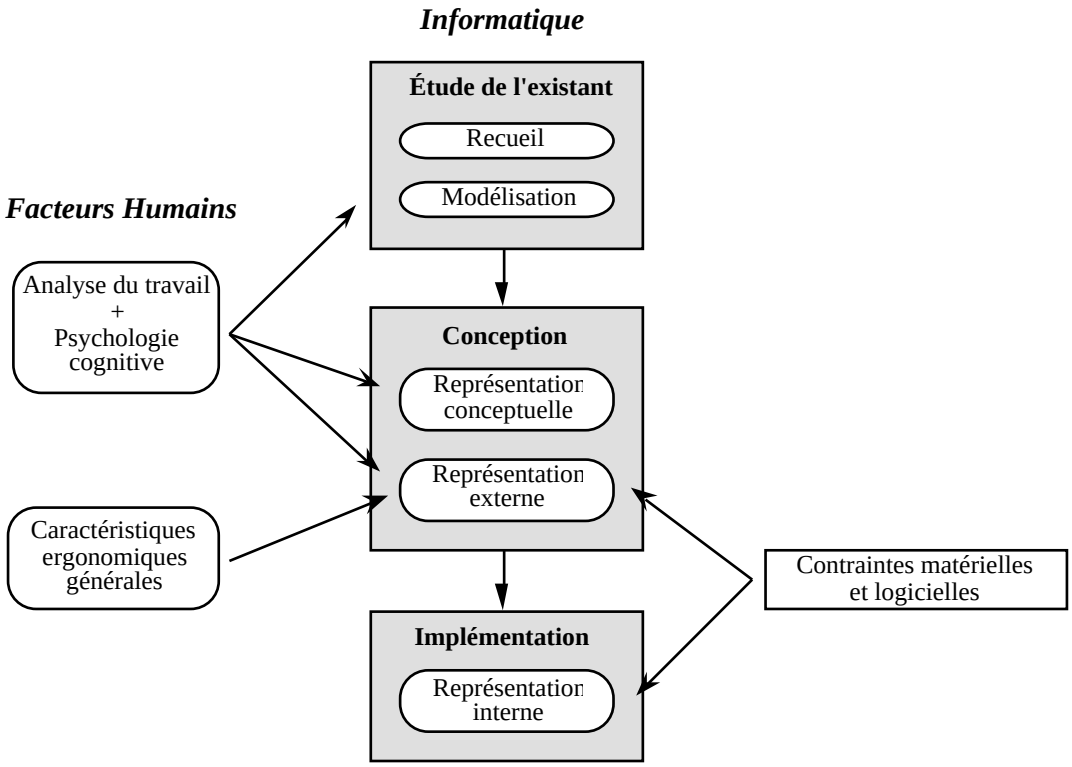

Figure 0-2 : Intégration de DIANE+ dans le cycle de vie d'un logiciel interactif [Palanque et al. 1994].

Le projet NEIMO (NouvellEs Interfaces et MOdélisation) a pour but la ‹‹ consolidation des espaces de conception et des techniques développés jusqu'ici pour la modélisation logicielle des systèmes interactifs et l'extension au cas des nouvelles interfaces : collecticiel, "mediaspace", et réalités augmentées ›› [Coutaz 1994]. Le projet NEIMO n'est pas comparable à MUSE/JSD ou DIANE/MERISE, au sens où il doit être considéré comme un cadre de recherche plutôt que comme un outil finalisé. Fortement inspiré des méthodes du génie logiciel, NEIMO intègre également dès les premiers stades de conception des étapes consacrées aux évaluations ergonomiques. Elles sont en cas de nécessité suivies de retours arrière comme le montrent les flèches situées à gauche sur la figure 0-3. C'est pourquoi NEIMO peut être vu comme la démarche comparable à DIANE et inverse de MUSE, c'est-à-dire une tentative du génie logiciel de s'intégrer à l'ergonomie cognitive, ici par le moyen de l'évaluation ergonomique. Cependant, NEIMO se distingue de DIANE au sens où il s'intéresse principalement aux "nouvelles interfaces" et désire faire appel de manière extensive aux outils automatisés.

MUSE, DIANE et NEIMO mettent en lumière la faisabilité d'une liaison entre ergonomie cognitive et génie logiciel. Néanmoins, au-delà des limitations de performance dues à l'approche de recherche qui a conduit à la réalisation de MUSE, DIANE et NEIMO, une critique générale subsiste : MUSE, DIANE et NEIMO héritent du domaine qui les a vu naître et la mise en place de passerelles vers l'autre domaine est encore une opération délicate. La construction de telles passerelles entre ces deux spécialités complémentaires est pourtant nécessaire, afin d'améliorer les méthodes de conception actuelles. L'un des éléments indispensables à cette construction est la disponibilité de formalismes de description réalisant le lien entre le point de vue de l'ergonomie cognitive, et celui du génie logiciel. Nous évoquerons la réalisation de ces formalismes à la section 4 de cette introduction, après avoir introduit le contexte et les motivations de ce travail de recherche dans la section 3 ci-après.

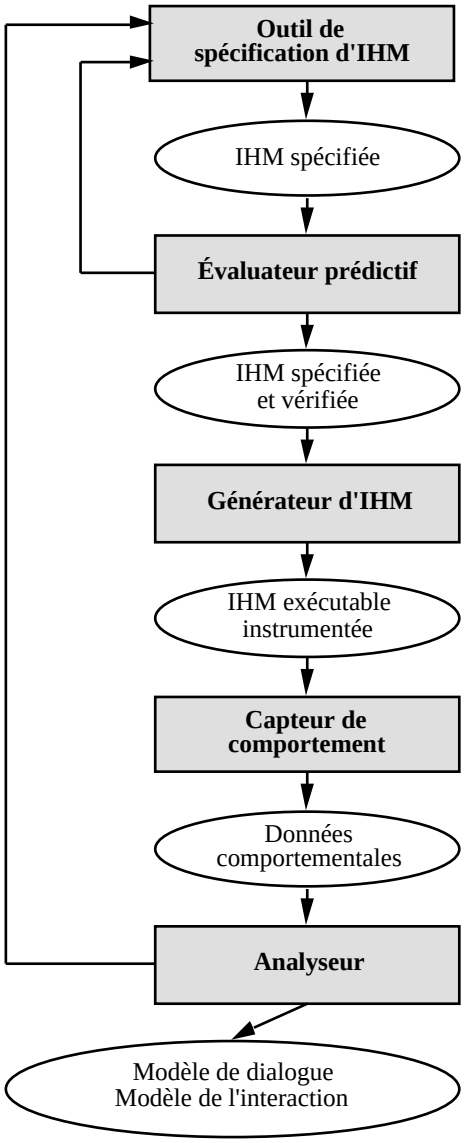

Figure 0-3 : Synoptique simplifié du projet NEIMO. Adapté de [Coutaz 1994].

3. Contexte et motivations

Les systèmes interactifs disposent tous par définition d'une interface homme-machine, et les systèmes critiques n'échappent pas à la règle. Cependant, comme nous l'avons souligné dans l'avant-propos de ce mémoire, une mauvaise interaction entre l'homme et la machine dans de tels systèmes peut avoir des conséquences dramatiques. C'est pourquoi une attention particulière doit leur être apportée, en particulier lors de l'étape de conception.

La conduite des systèmes critiques se caractérise par le respect de la procédure, synonyme de sécurité. En cela ils s'opposent aux systèmes non-critiques, comme en bureautique, où le non-respect des règlements s'effectue

souvent dans le but d'améliorer l'efficacité du système. Le contexte de ce travail à trait aux systèmes critiques. C'est pourquoi la sécurité y est considérée comme une qualité primordiale des systèmes étudiés, au détriment si besoin est de l'efficacité.

L'amélioration de la démarche de conception de ces systèmes a motivé le travail de recherche objet de ce mémoire. Afin de préciser le domaine, il convient d'étudier en premier lieu les particularités de tels systèmes vis-à-vis de leur interface homme-machine. C'est l'objet du paragraphe 3.1. Nous nous intéressons ensuite, au paragraphe 3.2, aux conséquences de ces particularités sur la démarche de conception d'un système interactif.

3.1. Les systèmes critiques, réactifs, et temps-réel

La *criticité* d'un système est un concept mal cerné. Intuitivement on pense aux systèmes qui, s'ils ne remplissent pas correctement leur fonction, mettent en jeu la vie d'êtres humains. C'est le cas par exemple des systèmes médicaux destinés à prendre en charge les fonctions vitales d'un patient. La notion de criticité englobe aussi les systèmes qui en cas de défaillance peuvent provoquer des catastrophes non seulement humaines, mais aussi économiques ou bien encore stratégiques. Les réseaux de transactions bancaires ou de communications militaires en sont de bons exemples. La notion de criticité ne dépend pas seulement du type de système utilisé mais plus de l'utilisation qui en est faite. Un traitement de texte peut être critique s'il sert à rédiger des rapports médicaux vitaux, alors qu'un logiciel de calcul de navigation pour aéronef peut ne pas être critique s'il est utilisé dans un simulateur. Beaucoup moins critiques, mais faisant toujours partie des risques informatiques, il apparaît de plus en plus dans notre vie quotidienne ce que l'on peut appeler des "tracasseries" informatiques. Elles peuvent aller de l'erreur sur une réservation de billets d'avion, jusqu'à la modification impromptue du classement d'un concours[5] suite à une "erreur d'ordinateur". A ce propos, le forum de discussion "comp.risks" [Neumann 1996] donne un très bon aperçu des problèmes pouvant affecter gravement notre quotidien en raison de défaillances informatiques.

Un système est dit *réactif* lorsqu'il interagit avec son environnement. Cette définition est souvent associée à celle de système *temps-réel* car les systèmes

[5] Cette information ne provient pas du forum "comp.risk" mais a été vécue par l'auteur (Concours ENSI-DEUG 1989).

réactifs sont le plus souvent aussi soumis à des contraintes de temps de réponse. En la matière, l'exemple-type est celui du contrôle de procédé industriel qui est à la fois réactif et temps-réel. Mais c'est aussi le cas, ou plutôt ce devrait être le cas, des interfaces homme-machine. En effet, une interface doit répondre immédiatement aux sollicitations de son utilisateur, et ne pas rester figée sur tel ou tel problème système en attendant le déclenchement d'un chien de garde interne. Ceci est d'autant plus vrai dans le cas des systèmes critiques. Remarquons que ces définitions proviennent du génie logiciel et s'appliquent principalement aux systèmes qui interagissent avec un environnement non humain. La défaillance du système est le plus souvent prise en compte mais pas celle d'une partie de l'environnement sur laquelle le système n'a pas toute autorité : l'opérateur. Ce que l'on appelle parfois un peu rapidement "l'erreur humaine" constitue un vrai problème de conception. De même, la notion d'interruption dans les systèmes réactifs devient beaucoup plus difficile à traiter quand l'un des protagonistes est humain et qu'il ne gère pas nécessairement en interne une pile d'événements !

3.2. Démarche de conception

Concevoir un logiciel industriellement est encore une entreprise difficile et parfois risquée. Une erreur de conception, lorsqu'elle se manifeste, peut avoir des conséquences financières très importantes pour la société qui a créé le logiciel en cause. La figure 0-4 montre que plus la détection de l'erreur est proche de la livraison, plus le coût de sa correction est important, et ce de manière exponentielle. Afin de répondre à ce problème de coût et de qualité de développement, le génie logiciel a fait émerger méthodes et outils permettant de réduire autant que possible la détection tardive des erreurs. La norme ISO-9000 et les normes associées en sont un exemple parmi d'autres.

Le cas des logiciels critiques est encore plus crucial car, outre les risques de drames humains, les conséquences financières d'un bug peuvent dépasser de plusieurs ordres de grandeur le coût de réalisation du logiciel incriminé. Pour tenter de minimiser ces risques, le génie logiciel a mis au point des méthodes formelles permettant de prouver la conformité de tout ou partie d'un logiciel aux spécifications. Une remarquable bibliographie sur ce sujet peut être consultée dans [Jacquet, Ledru, et al. 1996]. Même si de nombreuses recherches restent encore à effectuer, ces méthodes ont montré leur efficacité dans les systèmes de transport automatisés comme dans le cas de la ligne A du RER parisien. La réduction des coûts et des risques passe donc par un

nécessaire processus de conception qui lui même impose une spécification préalable. Il est beaucoup plus facile de corriger une erreur sur les spécifications que sur le système déjà réalisé. C'est à cette étape de spécifications que nous allons maintenant nous intéresser.

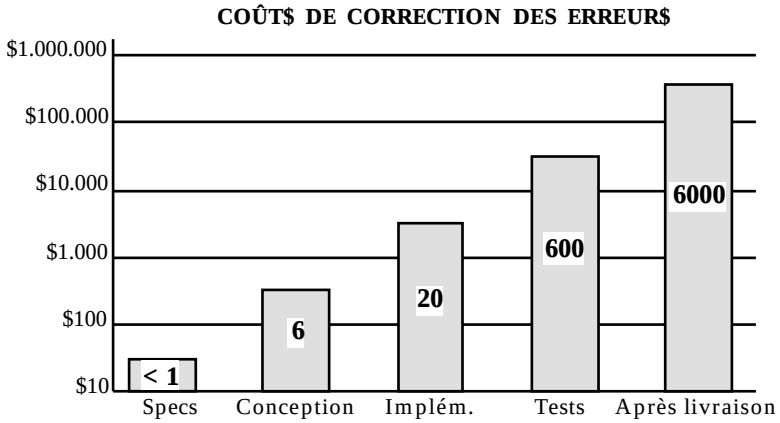

Figure 0-4 : Coûts de correction des erreurs d'après une étude interne de Hewlett-Packard. Les nombres indiqués dans les colonnes correspondent aux heures nécessaires à la correction.

L'interface homme-machine pose de nombreuses difficultés aux concepteurs car les outils de spécification à notre disposition sont limités. De bons résultats ont pourtant été obtenus pour quelques cas d'étude, mais le passage vers les systèmes à portée industrielle reste problématique. Le pouvoir d'expression de ces outils est encore trop faible ou leur utilisation trop complexe. La démarche de conception encore assez courante dans l'industrie[6] est la réalisation pure et simple de l'interface puis son évaluation tout aussi subjective. S'il reste du temps et un peu d'argent à y consacrer au sein du projet, on effectue alors éventuellement quelques modifications… Cette caricature de démarche de conception ne doit pas faire oublier que le cycle de développement le plus fameux est encore celui en étoile de Hix et Hartson [Hix & Hartson 1993]. Comme le montre la figure 0-5, ce cycle en étoile est centré sur l'évaluation autour de laquelle gravitent les différentes activités de conception. L'évaluation a lieu à la fin de chaque activité, juste avant de passer à une activité suivante. Les éventuelles modifications à apporter sont alors très coûteuses car elles arrivent tard au sein de chaque activité. En conséquence, ce type d'évaluation a posteriori, dite sommative, est bien moins efficace qu'une évaluation a priori, dite formative, ayant lieu

6 L'auteur ne citera pas ses sources…

tout au long de chaque activité, et nécessitant donc des retours en arrière moins importants.

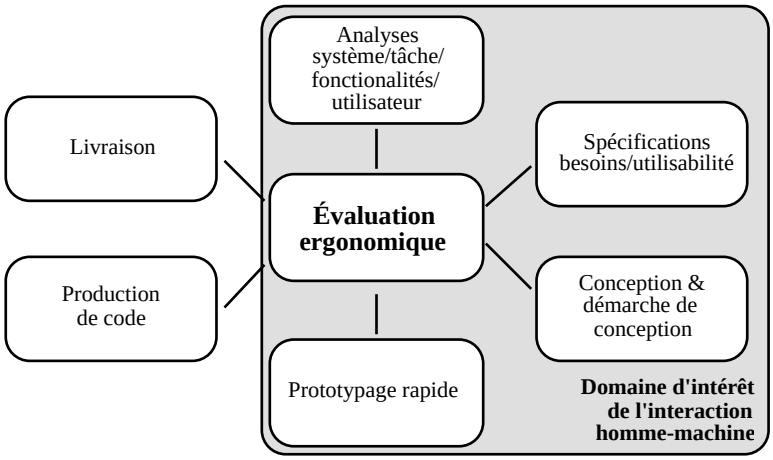

Figure 0-5 : Cycle de vie de étoile pour le développement des systèmes interactifs. Traduit de [Hix et al. 1993].

Lorsque l'interface homme-machine est destinée à un système critique, d'autres problèmes viennent s'ajouter aux faiblesses des outils de conception. En particulier, la notion d'erreur et d'interruption devient une donnée prépondérante qu'il convient de prendre en compte. Les systèmes critiques sont maintenant réalisés de manière générale avec des méthodes formelles, mais le plus souvent leur interface homme-machine n'en dispose pas. Les erreurs et les interruptions sont le plus souvent occultées, y compris dans les langages de spécification eux-mêmes. C'est à cette lacune que le travail présenté dans ce mémoire tente de répondre, selon une approche de recherche que nous allons maintenant détailler.

4. Approche de recherche

Concevoir une interface homme-machine, nous l'avons vu, nécessite une étape préalable de spécifications à l'aide d'un outil de description. Or il est très tentant de construire l'Outil de description parfait, à la fois simple à utiliser, et possédant un pouvoir d'expression aussi complet qu'ouvert aux futures découvertes. C'est probablement le rêve de tout chercheur travaillant sur le sujet. Mais c'est aussi le cauchemar de tous ceux qui devront l'utiliser. Il sera peut-être parfait, mais ne conviendra probablement qu'à celui qui l'a créé. En effet, la démarche de conception d'une interface homme-machine ne ressemble pas à un monolithe qu'un seul individu peut réaliser derrière son

écran. De nombreuses compétences sont nécessaires, correspondant à plusieurs métiers. Tout bon artisan possède et maîtrise les outils de sont art. Il en est de même pour ceux chargés de concevoir les interfaces homme-machine. L'ergonome analyse la tâche avec sa propre méthode, l'ingénieur fait de l'ingénierie avec sa propre notation. Quant au programmeur, il réalise son code informatique dans son langage. Faire utiliser le C^{++} orienté-objet à un ergonome revient à donner un couteau à bois à un bûcheron. A chaque besoin correspond son outil de description. Il ne doit être ni trop étendu et complexe, ni trop simpliste et étriqué. Il doit juste convenir aux besoins, ni plus, ni moins. De nombreux formalismes existent, certains plus complets que d'autres, les uns plus à la mode que les autres, les uns convenant mieux à tel ou tel problème que les autres. A chaque nouvelle thèse apparaît un nouveau formalisme qui, comme les autres, conviendra plus ou moins à tel ou tel type de problème.

Concevoir une interface homme-machine en respectant un cheminement linéaire débutant par l'analyse de l'existant et la définition du nouveau système, pour aboutir à la réalisation informatique de l'interface est en pratique peu réaliste. En effet, de nombreux retours arrière sont nécessaires à tous les niveaux de conception, conséquences des phases d'évaluation. D'autre part, de nombreux choix a priori concernant les couches basses de l'interface sont très souvent effectués bien avant toute idée d'ergonomie. Ces contraintes portent sur l'aspect matériel de l'interface, taille de l'écran ou nombre de boutons possibles, et même parfois sur les bibliothèques logicielles et langages de programmation à utiliser. Hix et Hartson affirment que la conception d'un logiciel impose à la réalisation ses spécifications, mais reçoit aussi en retour des contraintes et problèmes d'ordre technique comme l'illustre la figure 0-6 [Hix et al. 1993]. Le logiciel final est l'aboutissement d'un compromis entre ces deux ensembles de contraintes. Cependant, pour des raisons économiques, la réalisation a souvent le dernier mot… Par ailleurs, la phase de spécification d'une interface homme-machine peut déjà être le lieu de nombreux retours en arrière. En effet, cette phase initiale est souvent effectuée par plusieurs personnes, dont le client final, qui dès ce stade du processus de conception, peut voir certains problèmes et demander des modifications.

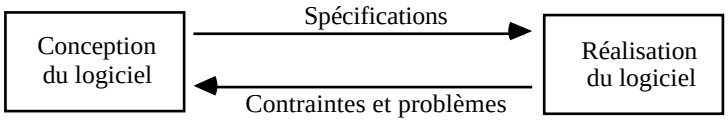

Figure 0-6 : Distinction entre conception et réalisation de logiciel. Traduit de [Hix et al. 1993].

Les deux points évoqués ci-dessus nous amènent à envisager une approche pragmatique : plutôt que de recréer ex nihilo un nouveau formalisme, la démarche suivie dans ce mémoire s'appuie sur la définition de concepts pour ensuite en dériver des propriétés et des extensions aux formalismes actuels. Cette approche aborde quelques-unes des pierres d'achoppement des formalismes de spécification actuels, d'une manière aussi générale que possible. L'identification de ces difficultés a été dictée par les applications envisagées, les systèmes critiques, où erreurs et interruptions doivent être étudiées avec soin. Bien que ces concepts soient généraux et puissent s'appliquer à tout formalisme de spécification, des formalismes cibles, UAN, MAD, et les réseaux de Petri, ont été choisis pour illustrer la méthode. Enfin, définir un concept puis en déduire des extensions aux formalismes existants ne suffit pas. En effet, la conception d'une interface homme-machine ne ressemble pas à un monolithe, mais est constituée d'une multitude d'étapes actuellement peu reliées entre elles. C'est pourquoi la seconde partie de la contribution de ce mémoire est l'étude de ces liaisons. Elles doivent permettre de rendre cohérente la notion d'erreur et d'interruption sur l'ensemble de la phase de spécification d'une interface homme-machine en autorisant l'usage de passerelles entre les formalismes de spécification. Ce passage une fois défini peut être automatisé, ainsi le travail de chacun des spécialistes ne devra plus être initié à partir d'une base floue à chaque phase de conception, mais s'inspirera de la structure et des résultats de la phase précédente.

5. Organisation du mémoire

Ce mémoire est divisé en deux parties de volumes comparables. La première s'intéresse principalement aux concepts et à la modélisation. Elle a pour sujet l'automatisation, l'erreur humaine, et les interruptions. Elle intéressera probablement en premier les ergonomes. Cette première partie regroupe les chapitres I et II :

• Le premier chapitre insiste sur l'importance de l'automatisation et de ses conséquences sur l'erreur humaine. Il propose un tour d'horizon des

recherches sur l'automatisation, principalement issues du domaine aéronautique. Ce chapitre résume en outre l'état des recherches sur l'erreur humaine, en provenance des sciences cognitives.

• Le deuxième chapitre s'intéresse à la prévision des risques d'erreur et à leur correction. Une représentation graphique de l'erreur, associée à sa correction est proposée. Puis, les relations entre erreurs et interruptions sont évoquées, ainsi qu'une proposition de modélisation de ces dernières.

La seconde partie de ce mémoire est beaucoup plus appliquée. Elle a pour objet les formalismes de spécification. Elle intéressera tous ceux qui désirent utiliser des spécifications formelles. Cette seconde partie regroupe les chapitres III et IV :

• Le troisième chapitre propose une taxinomie des classifications des formalismes de spécification destinés aux interfaces homme-machine, et en propose un guide d'utilisation. Ce guide est illustré par l'exemple du choix de trois formalismes, UAN, MAD, et les réseaux de Petri, qui a été fait dans le cadre de ce mémoire.

• Le quatrième chapitre constitue l'application des concepts et modèles définis au chapitre II, aux formalismes choisis au chapitre III. Il décrit en détail les extensions et les traducteurs proposés. Ce chapitre regroupe également une série de tableaux synthétiques destinés à faciliter leur utilisation.

Ce mémoire suit un plan atypique au sens où il ne sépare pas explicitement l'analyse de l'existant des contributions proposées. En effet, il invite le lecteur à un cheminement révélant la démarche de recherche adoptée : des concepts vers les formalismes. À chaque étape de ce cheminement l'état de l'art est évoqué, éventuellement critiqué, puis des contributions sont proposées. Les liaisons entre état de l'art et contributions sont ainsi mises en lumière. Le lecteur intéressé par les seules contributions de ce mémoire est invité à se référer aux sections et paragraphes qui s'y rattachent. Les contributions présentes aux chapitres I et II sont indiqués par le symbole 🕮. Quant aux chapitres III et IV, ils y sont essentiellement dédiés. Il ne nous reste plus qu'à vous souhaiter une bonne lecture, exempte d'erreurs et non interrompue…

6. Bibliographie

[Balbo 1994] Balbo S. *Un pas vers l'évaluation automatique des interfaces homme-machine*. Thèse en Informatique : Université Joseph Fourier (Grenoble I), 5 Septembre 1994.

[Barthet 1988] Barthet M.-F. *Logiciels interactifs et ergonomie : Modèles et méthodes de conception*. Paris, France : Dunod, 1988.

[Boehm 1988] Boehm B.W. A spiral model of software development and enhancement. *IEEE Computer*, May 1988.

[Coutaz 1994] Coutaz J. *NEIMO (NouvellEs Interfaces et MOdélisation)*. CLIPS-IMAG, Avril 1994. Proposition de projet IMAG.

[Hix et al. 1993] Hix D. & Hartson H.R. *Developping user interfaces: Ensuring usability through product & process*. Newyork, USA : John Wiley & Sons, inc., 1993.

[Jackson 1983] Jackson M.A. *System developement*. Englewood Cliffs (NJ), USA : Prentice Hall, 1983.

[Jacquet et al. 1996] Jacquet P., Ledru Y., Nicollin X., & Potet M.-L. *Exposition "logiciels critiques"*. Médiathèque IMAG, 15 janvier - 15 avril 1996. Documentation.

[Jambon 1995] Jambon F. Interruptions et formalismes de scripts de tâches. *Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95), Toulouse, France*, 11-13 Octobre 1995. p. 161-168.

[Lim et al. 1994] Lim K.Y. & Long J.B. *The MUSE method for usability engineering*. United Kingdom : Cambridge University Press, 1994.

[Long 1995] Long J. Integrating human factors with software engineering for human-computer interaction. *Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95), Toulouse, France*, 11-13 Octobre 1995. p. 1-19.

[Long et al. 1989] Long J. & Dowell J. Conceptions of the discipline of HCI: Craft, Applied Science, and Engineering. *Fifth conference of the BCS HCI SIG*, 1989.

[McDermind et al. 1984] McDermind J. & Ripkin K. *Life cycle support in the ADA environment*. Cambridge University Press, 1984.

[Neumann 1996] Neumann P.G. Forum on risks to the public in computers and related systems ("comp.risks"). *Moderated Newsgroup*, 1996.

[Palanque et al. 1994] Palanque P., Long J.B., Tarby J.-C., Barthet M.-F., & Lim K.Y. Conception d'applications ergonomiques : une méthode pour informaticiens et une méthode pour ergonomes. *Ergonomie et Informatique Avancée (ERGO-IA'94), Biarritz, France*, 28-10 octobre 1994.

[Royce 1970] Royce W.W. Managing the development of large software systems. *WESTCON, California, USA*, 1970.

[Tarby 1993] Tarby J.-C. *Gestion automatique du dialogue homme-machine à partir de spécifiactions conceptuelles*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1993.

Première partie

CONCEPTS

Errare humanum est, perseverare diabolicum.

CHAPITRE I

AUTOMATISATION ET ERREUR HUMAINE

1. Introduction

‹‹ Ce jour-là tu auras commis une erreur, ou plutôt une faute. Bien sûr, il t'est déjà arrivé, comme tout le monde, de te tromper en programmant ton magnétoscope, mais là c'est différent. On n'a pas le droit de se tromper quand on a de telles responsabilités. Cet avion qui s'est écrasé par ta faute, cette usine qui a explosé, ou ce bateau qui a coulé à cause de toi, tu en étais responsable. Tu savais qu'il ne fallait pas utiliser cette fonction dans ce mode de fonctionnement, tu avais l'habitude pourtant. Tu n'aurais pas dû te tromper ! ›› Cette introduction, inspirée de la *Lettre ouverte à un futur ami* rédigée par Nicolet, Carnino, et Wanner [Nicolet, Carnino, & Wanner 1990] représente le discours type entendu presque invariablement après chaque catastrophe où les facteurs humains ont été mis en cause. En effet l'erreur et ses déclinaisons, les fautes, les lapsus, les ratés, etc. sont les mal-aimés de notre société technologique. Une toute petite erreur de manipulation peut avoir de nos jours des conséquences sans commune mesure avec l'erreur commise, et valoir à son auteur une avalanche de reproches. On a coutume de dire que "l'erreur est humaine", mais l'opérateur fautif est néanmoins presque toujours blâmé…

Pourtant l'erreur ne doit pas être vue comme une fatalité mais plutôt comme le prix à payer pour la formidable capacité d'adaptation et de traitement de l'information de l'être humain. Que ce soit pour reconnaître une situation, un visage, ou encore simplement pour marcher sur deux jambes sur un terrain irrégulier sans trébucher, l'homme est encore bien supérieur à la machine. Le cerveau humain possède en effet des capacités d'adaptation et de parallélisme hors du commun. Mais cette performance a un coût : l'erreur. Chercher à supprimer l'erreur humaine est une quête inutile et certainement vouée à l'échec. Une démarche plus cohérente consiste à apprivoiser l'erreur, à la domestiquer et à apprendre à la connaître de façon à réduire ses occurrences et à prévoir son comportement. L'être humain doit admettre que l'erreur surviendra, généralement au pire moment. L'homme doit alors pouvoir s'apercevoir de son erreur et être en mesure de la corriger.

Les machines ont acquis progressivement ce que l'on pourrait appeler leur indépendance. Elles sont passées de l'outil inerte que l'homme anime, comme une chignole, à l'automate animé par un comportement interne, comme une machine-outil. L'interaction homme-machine est ainsi passée de l'action

directe sur un outil à la supervision d'un automate. Même si le but est resté invariant, ici percer une pièce, la tâche de l'homme a été profondément modifiée. Les erreurs humaines qui étaient avant tout causées par un manque d'habileté sont maintenant dues le plus souvent à des erreurs de représentation mentale. Certains affirment que la machine ne commet jamais d'erreur. C'est probablement exact d'un certain point de vue. La tentation est alors grande de confier à la seule machine la commande de systèmes hautement critiques où aucune erreur n'est tolérable. L'erreur est ici de croire que la machine ne fait pas d'erreur. Il s'agit simplement d'autres types d'erreur. Et c'est cette différence qui permet au couple homme-machine d'être porteur d'un potentiel extraordinaire : il peut être conçu de façon à ce que les erreurs de l'un soient corrigées grâce à l'attention de l'autre. Encore faut-il accepter l'erreur humaine. Du côté de la machine, on appelle cela la fiabilité. Elle est quantifiée, normalisée, et fait partie par exemple de la certification des aéronefs de transport civil [Mellor 1993]. Par contre, le pilote de l'aéronef en question n'est pas certifié. Il est seulement entraîné à s'adapter à la machine, et surtout à ses pannes. Cet entraînement est sanctionné par une série de tests où l'erreur de l'homme est peu tolérée. Il serait peut-être temps d'adapter aussi la machine aux erreurs de l'homme.

Commettre une erreur suite à une interruption inopinée est une chose courante : vous étiez parti chercher le pain, vous avez rencontré un ami, après avoir bavardé vous êtes rentré chez vous sans le pain… De même, corriger une erreur nécessite habituellement une interruption de la tâche en cours : vous étiez en train de taper un numéro de téléphone lorsque votre doigt a glissé sur une mauvaise touche, vous avez dû raccrocher et recommencer l'appel. Les interruptions et les erreurs sont deux notions très liées, l'une provoquant souvent l'autre. De plus, les interruptions sont omniprésentes aussi bien dans la vie quotidienne que lors de la conduite de systèmes critiques. Pourtant elles n'ont à ce jour, contrairement aux erreurs, que peu été étudiées du point de vue de l'interaction homme-machine. C'est donc un domaine encore à défricher où l'on peut s'inspirer en partie des résultats obtenus sur les interruptions au niveau des systèmes d'exploitation, mais aussi de la notion de récupération d'erreur de la psycho-ergonomie.

Ce chapitre s'intéresse principalement à l'erreur humaine. Celle-ci a été mise à l'actualité en raison de l'importance croissante de l'automatisation dans notre société. C'est pourquoi nous évoquerons tout d'abord

l'automatisation et ses conséquences sur l'erreur humaine, dans la section 2 de ce chapitre. La troisième section s'intéresse, quant à elle, à la nature de l'erreur et à ses classifications. Enfin, la quatrième et dernière section conclut ce chapitre en mettant l'accent sur les erreurs latentes au sein des organisations.

2. Automatisation

Les automates, même les plus simples, font partie de notre environnement quotidien. La définition que donne le dictionnaire Robert [Robert 1992] est vaste : "Toute machine animée par un mécanisme intérieur". Un radioréveil, un four électrique à thermostat, ou un système de traitement de texte sont des automates. Ils simplifient notre quotidien et permettent d'éviter de nombreuses erreurs : se réveiller à la bonne heure ou ne pas faire brûler un gâteau. Mais leur comportement parfois hasardeux génère d'autres erreurs : passer une demi-heure pour prendre un billet de chemin de fer, ou encore être réveillé à six heures un samedi matin parce que l'on a oublié d'arrêter le radioréveil à la veille du week-end. Pourtant, qui accepterait de se passer de ces automates ? De la plus simple montre au cockpit complexe d'un aéronef illustré figure I-1, les automates ont été introduits parce qu'ils répondent à des besoins : la migration vers la machine de tâches subalternes, mais aussi comme palliatif à certaines déficiences humaines. Pour des raisons historiques l'automatisation a surtout été étudiée en aéronautique. Les recherches sur l'automatisation des cockpits de Boy et Hollnagel [Boy & Hollnagel 1993] en Europe, Wiener [Wiener 1989 ; Wiener & Curry 1980] et Billings [Billings 1991 ; Billings 1996] à la NASA en sont des exemples.

Cependant l'automate n'est pas un outil anodin. Son utilisation doit répondre à des besoins précis et réaliser des fonctions ou rôles bien définis que nous présenterons au paragraphe 2.1. Mais, du point de vue de l'interaction homme-machine, l'automatisation n'a pas que des avantages. Le paragraphe 2.2 montre que son introduction présente également des risques et des effets pervers. Nous sommes aujourd'hui dans une situation intermédiaire où l'automatisation est indispensable mais encore assez immature. Cette situation n'est cependant pas définitive. Le paragraphe 2.3 conclut cette section en indiquant quelques voies de recherche.

Figure I-1 : Poste de pilotage de l'A320 (Photo AIRBUS INDUSTRIE).

2.1. Le bon usage de l'automate

L'automate ne doit pas être considéré comme un gadget au risque d'être plus nuisible qu'efficace. Il remplit des fonctions qui peuvent aider considérablement l'opérateur dans sa tâche. Mais l'usage de l'automate doit être réfléchi, non pas motivé par les possibilités techniques du moment.

2.1.a. Critères de choix

Wanner, par exemple, suggère que ‹‹ le choix opérateur humain ou automatisme repose sur la philosophie suivante :

- si la réalisation d'une fonction présente des dangers physiques inacceptables pour l'homme,
- si la réalisation d'une fonction exige une habileté exceptionnelle de l'opérateur,
- si la réalisation d'une fonction exige un travail fastidieux et répétitif (comme le respect d'une valeur de consigne par une action simple, l'attente passive d'un événement rare, ou la surveillance d'un paramètre peu évolutif)

il faut l'automatiser.

[… Par contre,]

chaque fois qu'une décision repose sur des choix :

- qui ne peuvent se réduire à des algorithmes peu complexes, reposant sur des logiques faisant intervenir un nombre fini de paramètres internes et des états définis a priori,
- qui font intervenir une logique floue ou une évaluation qualitative de la situation,
- qui nécessitent une saisie de la situation par une méthode du type reconnaissance de forme,

il faut mettre l'homme dans la boucle en le laissant maître direct de l'action sur le système ›› [Wanner 1992]. Les critères de choix de Wanner sont issus de la pratique dans un domaine bien particulier, celui de l'aéronautique. Les automates de la vie courante sont beaucoup moins critiques. Pour ces automates, les critères de Wanner restent valides mais ils se sont pas nécessairement discriminants.

L'automatisation d'une tâche ne se décide souvent, à tord, en ne considérant que les gains attendus. Les inévitables coûts sont couramment minimisés ou même totalement occultés. L'utilisation d'un traitement de texte par exemple, suppose un important coût d'apprentissage pour un utilisateur novice. Ce coût s'amortit rapidement si le traitement de texte est utilisé régulièrement, mais il demeure élevé si son utilisation reste occasionnelle. Outre l'apprentissage, le coût global d'une tâche comprend le temps passé, mais aussi l'effort cognitif ou conatif nécessaire à l'utilisateur pour effectuer cette tâche. C'est pourquoi l'automatisation d'une tâche ne doit se faire que si le rapport gain sur coût joue en faveur de l'utilisateur. Il n'est pas vraiment intéressant d'utiliser un traitement de texte pour écrire une lettre par an, mais cela le devient pour écrire un mémoire de thèse.

Une classification plus étendue et plus précise des coûts pour l'utilisateur peut être consultée dans [Dorwell & Long 1989] lue dans [Balbo 1994]. Au sein des collecticiels, il est également nécessaire de distinguer quels utilisateurs supportent les coûts, et lesquels bénéficient des gains. Grudin a avancé l'idée que l'inégalité entre ceux qui fournissent du travail et ceux qui en bénéficient, est à l'origine de l'échec des agendas électroniques [Grudin 1988]. Le rapport gain sur coût n'a pas la prétention d'être calculable pratiquement. Il a simplement pour objet d'inciter les concepteurs à étudier toutes les facettes du problème. L'automate n'est pas la solution miracle à

tous les maux, et il n'est pas exempt de critiques comme nous le verrons à la section 2.2.

2.1.b. Fonctions et rôles de l'automate

D'après Hollnagel, Warren, et Cacciabue, un automate couvre trois fonctions essentielles [Hollnagel, Warren, & Cacciabue 1993]. Il peut servir de substitut, d'amplificateur, et de générateur de nouvelles fonctionnalités :

- L'automate se substitue à l'opérateur pour certaines tâches. Il devient alors une sorte de prothèse. Le radioréveil qui se substitue à la mémoire est un exemple de ce type d'automate très courant dans notre quotidien.
- L'automate permet d'amplifier les possibilités de l'opérateur. Il est alors considéré comme un outil. Le microscope, mais aussi les véhicules qui permettent de se déplacer plus vite et plus loin, font partie de ce type d'automate.
- L'automate permet aux opérateurs d'effectuer des tâches qu'ils ne pouvaient réaliser auparavant. Le pilotage des avions instables à l'aide des commandes de vol électriques en est un bon exemple. Il est repris au paragraphe 2.1.c.

En outre, Boy introduit une classification des rôles des agents artificiels et des opérateurs humains du point de vue de l'acquisition de connaissances dans les systèmes dynamiques [Boy 1993]. Cette classification distingue les rôles des agents ou opérateurs selon quatre catégories : action, perception d'information, apprentissage, et utilisation des capacités cognitives. Cette classification, plus précise mais ne montrant pas la "valeur ajoutée" de l'automate, complète celle de Hollnagel, Warren, et Cacciabue.

2.1.c. L'exemple des commandes de vol électriques

La plupart des avions de ligne fabriqués actuellement possèdent des commandes de vol électriques qui remplacent les classiques jeux de câbles et de poulies utilisés pour transmettre les ordres du pilote vers les gouvernes. Les ordres du pilote sur le manche sont maintenant transformés en informations électriques qui actionnent les gouvernes par l'intermédiaire de servocommandes. L'introduction des commandes électriques s'est accompagnée d'une entrée en force des calculateurs ayant pour rôle le filtrage des ordres du pilote. En conséquence, le pilotage ne s'effectue plus directement par déplacement des gouvernes, mais par définition d'objectifs :

monter, descendre. La tâche de maintien de la trajectoire a ainsi partiellement migré du pilote vers l'automate.

Les commandes de vol électriques ont été introduites pour la première fois sur un avion civil avec Concorde [Chevalier 1992] : la taille de l'appareil et la nécessité d'obtenir une grande précision dans le mouvement des commandes rendaient critique l'utilisation de câbles. Sur les avions militaires la maniabilité est une capacité vitale en combat aérien, mais elle a pour conséquence de provoquer l'instabilité de l'aéronef. Un appareil de type Rafale par exemple n'est plus pilotable avec les temps de réponse humains. L'introduction de commandes électriques est alors nécessaire. Celles-ci prennent le relais entre les ordres du pilote et le mouvement effectif des gouvernes de manière à rendre l'appareil pilotable artificiellement [Dupont 1994]. Les commandes électriques permettent également des gains en sécurité et en confort. Sur l'A320 une sécurité empêche l'appareil d'atteindre l'incidence de décrochage et le maintient ainsi pilotable [Airbus Industrie 1992]. Sur les avions de ligne de grande taille comme l'A340, les commandes électriques filtrent les ordres du pilote de manière à ne pas exciter la structure de l'aéronef à sa fréquence de résonance, ceci pour le confort des passagers.

2.2. Conséquences néfastes de l'automatisation

L'introduction de l'automate n'est pas sans contreparties : ses résultats sont parfois mitigés (§ 2.2.a). Elle entraîne en particulier une modification des compétences (§ 2.2.b), dont l'une des conséquences est l'apparition d'un fossé d'automatisation (§ 2.2.c). En outre l'automate est aussi faillible, et ses pannes influent sur les erreurs de l'opérateur (§ 2.2.d).

2.2.a. Des résultats mitigés

Le couple homme-automate n'est pas toujours harmonieux. Le grand public l'a découvert lors de l'entrée en service du système automatique de vente de billets SNCF, SOCRATE, et l'actualité nous l'a fait cruellement remarquer lors de récents accidents d'avion [Laronche 1993]. Même si l'automatisation est globalement une réussite, des problèmes épineux subsistent. Le domaine aéronautique nous fournit, une fois de plus, une base de données éloquente.

La fréquence, relativement aux heures de vol, des accidents d'avion n'a fait que diminuer depuis des décennies. L'amélioration de la fiabilité du matériel en est la raison principale. Dans le même temps, les erreurs humaines sont

demeurées tout aussi fréquentes. En conséquence, les causes d'accidents attribuées aux "facteurs humains" prédominent de nos jours. Depuis quelques années, du fait du caractère apparemment incompressible des erreurs humaines, on constate une stabilisation de la fréquence des accidents. L'augmentation constante du trafic aérien se traduit en conséquence par une augmentation du nombre d'accidents annuels, fort peu acceptée par les usagers. L'une des réponses des constructeurs à ce problème fut l'introduction massive des automates, retirant ainsi les équipages de la boucle de commande. Le résultat attendu, une diminution notable des erreurs humaines, se fait attendre. Car comme le montre Amalberti, l'automate a introduit d'autres types de problèmes liés aux facteurs humains, allant des erreurs lors de l'entrée de paramètres jusqu'aux problèmes sociaux [Amalberti 1992].

Selon le type d'étude, les auteurs estiment entre 50 et 90% les accidents graves de l'aviation générale dus aux facteurs humains. Néanmoins les auteurs sont assez unanimes pour dire que l'automatisation en soi ne constitue pas un problème, c'est le manque de retour d'information en provenance du système qui est critiqué [Norman 1991].

2.2.b. Modification des compétences

En raison de l'automatisation, l'homme est passé progressivement du rôle d'acteur à celui de superviseur. Le travail mental a progressivement replacé le travail manuel avec pour corollaire le risque de faire sortir l'opérateur de la boucle de commande [Norman 1991]. La compétence d'un pilote n'est plus uniquement axée sur son habileté physique, comme aux débuts de l'aviation, mais aussi et surtout sur ses capacités de planification. De ce fait, l'automatisation a tendance à normaliser les performances des opérateurs. Les performances des jeunes se rapprochant de celles des plus anciens, il peut en résulter des tensions sociales [Amalberti 1992]. De même, la modification des savoir-faire entraîne une perte de compétence dans le travail manuel dû au manque de pratique. La reprise en main d'un système lorsque les automates sont hors-ligne devient alors hasardeuse. L'une des ironies de l'automatisation est qu'en cas de panne l'homme, qui est alors supposé remplacer l'automate, doit avoir des performances équivalentes à celles de l'automate qui, précisément a été introduit en raison des mauvaises performances de l'homme…

2.2.c. Le fossé d'automatisation

La grande majorité des automates sont débrayables, le passage du mode automatique au mode manuel étant conservé pour des raisons de sécurité. Cependant il est illusoire de croire que passer en manuel est la solution miracle à tous les cas de panne identifiée d'un automate. L'homme n'est parfois plus capable pratiquement de se passer des automates bien qu'il en ait les capacités théoriques [Hollnagel et al. 1993]. Imaginons simplement une panne du réseau des feux tricolores d'une ville moyenne. Bien qu'il existe une procédure manuelle, la priorité à droite, le résultat sera immanquablement une série d'embouteillages. Bien plus critique, certains aéronefs militaires instables ne peuvent être pilotés sans leurs commandes de vol électriques, la reprise en manuel consiste alors à s'éjecter !

Comme l'illustre la figure I-2, l'augmentation du degré d'automatisation atteint par un système conduit à l'apparition d'un point de non retour en-dessous duquel le passage en manuel n'est plus possible. Lorsque le degré d'automatisation est réversible, le domaine de réversibilité où le passage en manuel est possible, s'étend jusqu'à une automatisation nulle. Par contre, lorsque le degré d'automatisation est devenu non réversible, le domaine de réversibilité ne s'étend plus jusqu'à cette automatisation nulle. Il apparaît alors un domaine de non réversibilité et le passage au contrôle manuel complet du système n'est plus possible. Ce fossé d'automatisation impose aux concepteurs de nombreuses contraintes sur la fiabilité de l'automate, mais aussi sur la fiabilité du système homme-machine dans son ensemble. Une erreur dans l'insertion d'un paramètre peut dans ce cas avoir des conséquences aussi désastreuses qu'une panne de l'automate.

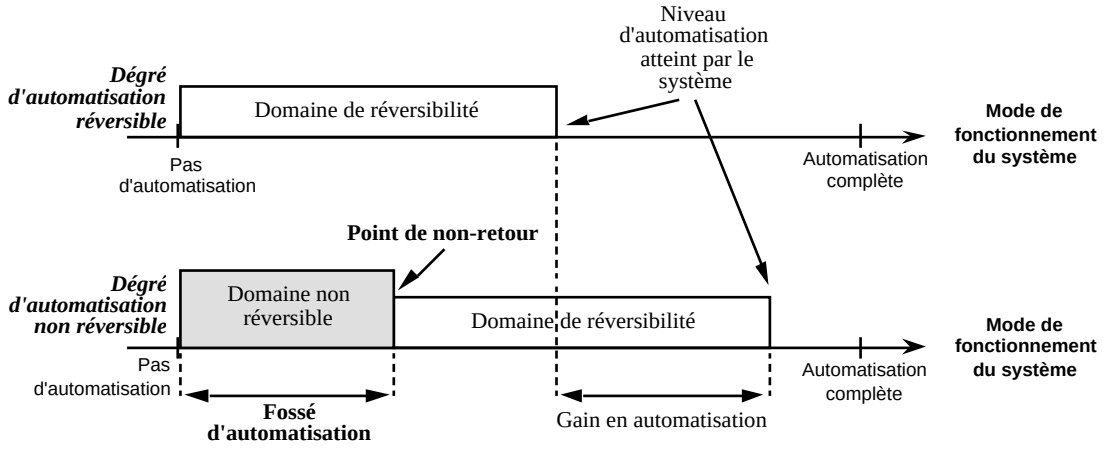

Figure I-2 : Fossé d'automatisation. Traduit et adapté de [Hollnagel et al. 1993].

2.2.d. Les pannes de l'automate et les erreurs de l'opérateur

Les cockpits modernes ont provoqué l'émergence d'un nouveau type de panne à laquelle les équipages étaient peu habitués : le bug informatique. De même qu'un câble peut se sectionner, un calculateur peut "planter". Ni plus, ni moins grave que la panne mécanique, le bug informatique est néanmoins méconnu des opérateurs. En corollaire, l'occurrence trop fréquente de ce type de panne, ayant un aspect "magique", provoque une perte de confiance des équipages vis-à-vis des automates qu'ils utilisent.

L'automatisation ne semble pas avoir réduit globalement la quantité d'erreurs, mais plutôt avoir déplacé les types d'erreur. D'après Sweet, l'automate permet d'éviter les petites erreurs mais crée l'opportunité à de graves erreurs d'apparaître [Sweet 1995]. Amalberti en dresse une liste détaillée [Amalberti 1992] :

• *Les erreurs de représentation* sont dues en partie à la grande complexité et aux nombreux échanges d'informations entre les calculateurs embarqués. La méconnaissance des interactions entre sous-systèmes peut alors provoquer de graves incidents en cas de panne. De plus, les possibilités de contrôle parfois restreintes des automates encouragent les équipages à détourner les procédures d'utilisation de leurs intentions premières. Ces procédures, utilisées hors de leur contexte, peuvent alors avoir des effets de bord insoupçonnés.

• *Les erreurs dans l'insertion de paramètres*, également appelées erreurs de doigt, sont dues au nombre important d'automates à paramètrer. Ces procédures répétitives de paramètrage sont aggravées par le fait que le pilote ne vérifie souvent que le résultat final et non pas les étapes intermédiaires.

• *Les erreurs liées à la détérioration de la coordination de l'équipage* ne sont pas dues à l'automatisation elle-même, mais ont été aggravées par le fait que les modes de communication homme-machine ont été modifiés. Les actions physiques à réaliser sur les planches de bord ont diminué et se limitent souvent à l'utilisation d'un clavier multiplexé. Le second pilote peut alors, en cas de mauvaise coordination de l'équipage, perdre une partie du contexte.

L'automatisation est une nécessité, mais ne doit pas être considérée comme la solution à tous les problèmes sécuritaires.

2.3. Une situation intermédiaire

L'automatisation n'a pas encore atteint sa maturité. Elle est dans une étape intermédiaire où de meilleurs résultats seraient obtenus en ayant soit moins d'automatisation, soit plus d'automatisation. De nombreux incidents et accidents sont évités grâce à l'automate, mais des incidents et accidents se produisent aussi à cause de l'automate. De plus, l'introduction de l'automatisation, comme toute nouveauté, provoque une perte temporaire de performance due à une nécessaire période d'adaptation [Hollnagel et al. 1993]. Cette période peut pourtant être réduite par une meilleure prise en compte des facteurs humains lors de la conception des systèmes. Il arrive cependant que la période d'adaptation se prolonge du fait de carences manifestes du système. La solution de facilité consiste alors à appliquer la "blame and train philosophy" : à chaque problème grave rencontré lors de l'utilisation d'un automate, l'opérateur fautif est blâmé, puis une nouvelle procédure plus contraignante est créée [Norman 1991]. En effet, remettre en cause la conception globale d'un système existant du jour au lendemain n'est pas réalisable techniquement ni économiquement. Loin de porter ses fruits, cette méthode a pour conséquence de rendre les procédures tellement complexes que les opérateurs sont tentés de les court-circuiter.

Afin d'y remédier, Amalberti propose deux approches complémentaires [Amalberti 1992]. À court terme, il est nécessaire d'adapter l'homme en ayant de nouvelles attitudes pédagogiques et en prenant également en compte la dimension sociale du problème. Loin de la "blame and train philosophy" cette attitude consiste à former l'opérateur aux problèmes spécifiques de l'automate dans les conditions les plus proches possibles de la réalité, et non pas de lui inculquer de nouvelles procédures. À long terme, il faut adapter l'automate non pas seulement aux contraintes techniques des systèmes, mais aussi et surtout à l'opérateur. En allant un peu plus loin dans le raisonnement, il faudrait même aller jusqu'à modifier les structures des sous-systèmes afin de simplifier la compréhension, pour l'opérateur, des conséquences de la panne d'un automate ou d'une erreur. Pour cela, il est avant tout nécessaire d'étudier avec soin les différents types d'erreurs des opérateurs, c'est l'objet de la section suivante.

3. L’erreur humaine

L’erreur humaine a été étudiée principalement à l’initiative de deux domaines très demandeurs : l’aéronautique et l’industrie de l’énergie nucléaire. Les résultats obtenus s’appliquent en premier lieu aux postes de pilotage des aéronefs et aux salles de contrôle des centrales nucléaires, mais ils ont également une portée générale. Les postes de conduite des véhicules, les salles de contrôle des industries à risque, mais aussi les interfaces homme-machine les plus courantes peuvent appliquer les résultats obtenus. Avant toute chose il est nécessaire de définir l’erreur. Sa nature fait l’objet du paragraphe 3.1. Le paragraphe 3.2 fait un tour d’horizon des différentes classifications de l’erreur humaine avec leurs motivations. Enfin cette section se conclut, par le paragraphe 3.3, sur l’aspect multifacette de l’erreur humaine, et sur le choix de ces facettes vis-à-vis de la conception des interfaces homme-machine.

3.1. La nature de l’erreur

Ânerie, bêtise, bévue, boulette, bourde, confusion, égarement, errements, faute, fourvoiement, gaffe, illusion, lapsus, malentendu, maladresse, manquement, mécompte, mégarde, méprise, paralogisme, quiproquo, sophisme, vice de raisonnement, etc. Tous ces synonymes [Bertaud du Chazaud 1994] indiquent que l’erreur prend de nombreuses formes dans le langage courant. Les psychologues de leur côté définissent l’erreur selon deux voies possibles [Grau & Doireau 1995] :

L’approche la plus classique consiste à considérer l’erreur comme un écart à la norme. Le problème est bien entendu la définition précise de cette norme. Trois référentiels peuvent alors être considérés :

• *Le référentiel de la tâche prescrite* telle qu’elle est définie par les concepteurs. L’erreur se définit ici comme le non respect d’une procédure ou la non application des recommandations d’un manuel d’utilisation. Cette définition a l’avantage d’être précise et de pouvoir s’appuyer sur une documentation de référence. Mais elle se rapproche plus d’une définition juridique que d’une réelle réalité psychologique.

• *Le référentiel des opérateurs de la profession*. La référence est l’ensemble des tâches effectuées par un opérateur type du domaine. La tâche de référence est alors l’adaptation effectuée par les opérateurs de la tâche

prescrite citée plus haut. Cette définition prend en compte la réalité de l'activité des opérateurs, mais laisse de côté leur expertise individuelle. En effet, à chaque niveau de qualification correspond une adaptation des tâches prescrites.

• *Le référentiel de l'opérateur lui-même*. C'est l'activité individuelle de l'opérateur qui est prise en compte. Cette définition dépend des buts spécifiques que l'opérateur se définit dans une situation spécifique. On retrouve ce point de vue avec la seconde approche explicitée ci-dessous.

La seconde approche consiste à donner comme norme subjective l'intention de l'opérateur. Reason [Reason 1993] affirme que l'erreur est inséparable de la notion d'intention. Il propose un algorithme (figure I-3) permettant de différentier les types de comportements en répondant par oui ou non à trois questions au plus. Il est à noter qu'il est toujours possible de répondre aux questions posées. Cet algorithme permet donc de déterminer dans tous les cas un type de comportement. Notons qu'il est difficile d'expliciter la différence entre la notion d'intention définie par Reason et la notion de référentiel de l'opérateur évoquée ci-dessus. En fait l'approche de Reason doit plutôt être vue comme un affinement de l'approche précédente, plutôt que comme une approche vraiment nouvelle.

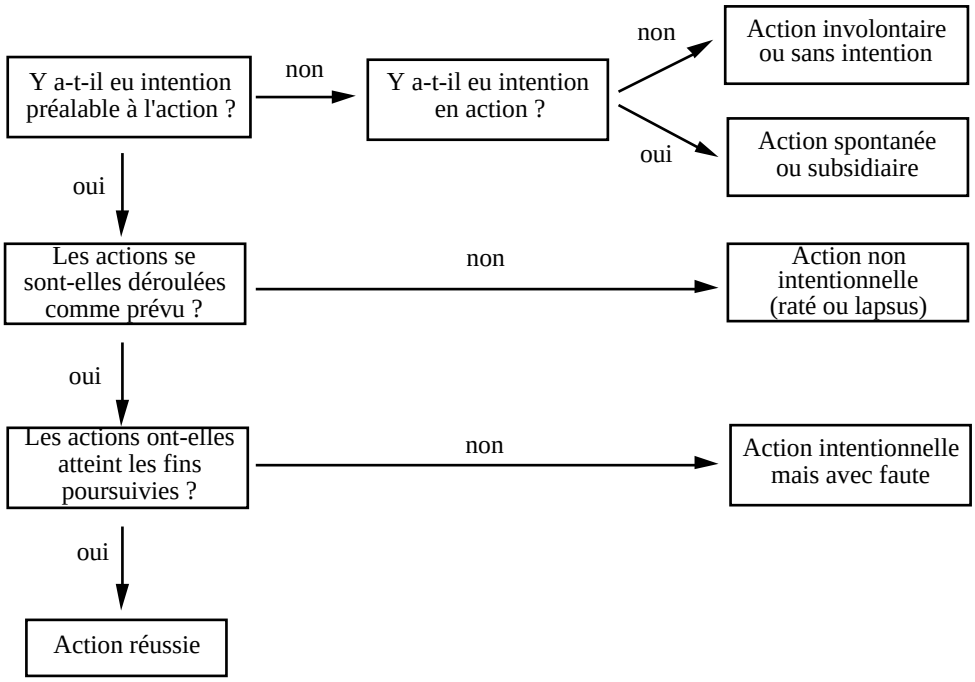

Figure I-3 : Algorithme permettant de différentier les types de comportement intentionnels : comportement sans intention, comportement non intentionnel (ratés et lapsus) et comportement intentionnel mais avec faute [Reason 1993].

Les référentiels de norme évoqués sont à rapprocher des trois types de procédures définis dans DIANE+ [Tarby 1993] :

• *La procédure prévue* correspond au déroulement attendu de l'interaction pour atteindre un but. Elle est généralement définie lors des spécifications par les concepteurs. Elle est à rapprocher du référentiel de la tâche prescrite bien qu'elle englobe également une série de tâches optionnelles.

• *La procédure effective* est celle définie pour un type d'utilisateur lors de la phase de conception. Il est possible de la rapprocher du référentiel des opérateurs de la profession. Cependant dans l'esprit de DIANE+, cette procédure permet principalement de définir des procédures différentes en fonction de critères comme l'expertise des utilisateurs ou le niveau de confidentialité. La procédure effective peut aussi être définie pour un utilisateur particulier. L'utilisateur utilise cette procédure, qu'il a définie lui-même, afin d'optimiser ses sessions de travail. Dans ce cas elle est à rapprocher de la norme de l'utilisateur lui-même.

• *La procédure minimale* est un peu différente car elle constitue le noyau dur des procédures prévues et effectives. C'est-à-dire les tâches minimales à réaliser pour atteindre le but fixé. En particulier les tâches facultatives en ont été retirées. On peut la voir comme une norme minimale, c'est-à-dire l'ensemble des tâches critiques à réaliser sans erreur pour atteindre le but fixé. Elle est également un compromis entre efficacité et sécurité.

Cependant DIANE+ effectue principalement la différenciation entre ces types de procédures sur l'utilisation ou non de tâches facultatives, sur l'enchaînement des tâches, et sur l'entité déclenchant la tâche. La notion d'erreur n'y est en effet pas intégrée.

3.2. Les classifications des erreurs

Les classifications des erreurs humaines publiées dans la littérature sont quasiment aussi nombreuses que les auteurs du domaine eux-mêmes. Selon Reason, les classifications des erreurs publiées dans la littérature peuvent elles-mêmes faire l'objet d'une taxinomie [Reason 1993]. Cette meta-classification comporte trois niveaux :

• *Le niveau conceptuel* exploite les hypothèses sur les mécanismes cognitifs impliqués dans la production d'erreur. Ce type de classification est plus

basé sur une modélisation théorique de l'erreur que sur des données observables.

• *Le niveau comportemental* est le plus superficiel. Il s'intéresse à la partie observable du comportement erroné. Les classifications de ce niveau peuvent s'intéresser à la caractéristique formelle de l'erreur, comme par exemple le mauvais ordonnancement des actions, mais aussi à ses conséquences, comme les dommages causés.

• *Le niveau contextuel* s'intéresse aux caractéristiques formelles de l'erreur mais aussi à des hypothèses causales. Il représente une sorte d'intermédiaire entre les deux autres niveaux de classification. Il permet en particulier de mettre en évidence l'interaction complexe entre des facteurs déclenchants locaux et les mécanismes d'erreur sous-jacents.

En d'autres termes deux grandes approches s'opposent. La première s'intéresse au rôle de la tâche et de la situation dans la production d'erreur. C'est l'approche contextuelle chère au praticien. La seconde est plus théorique et s'intéresse aux régularités et aux formes remarquables d'erreur. Il n'y a pas de consensus sur une classification fédératrice. Il semble que cela ne soit pas envisageable. En fait, chaque classification diffère sensiblement selon sa finalité théorique ou pratique. Il n'existe pas de bonne classification répondant à tous les souhaits. Seules certaines classifications répondent mieux aux besoins d'un problème particulier.

En conséquence, nombreuses sont les classifications de l'erreur. Certaines d'entre elles retiennent l'attention. La plus connue est sans nul doute la classification associée au modèle des niveaux d'activité de Rasmussen (§ 3.2.c). Elle fait partie des classifications basées sur un modèle théorique. Les travaux de Rasmussen, complétés par ceux de Reason sont détaillées ci-après. Mais avant cela, il est nécessaire de s'intéresser aux classifications plus appliquées utilisées par les praticiens du domaine. Destinés à une utilisation opérationnelle, elles ne disposent pas, en général, d'un modèle théorique. Ces classifications, même si elles présentent de nombreuses lacunes, ont l'avantage d'être plus aisément utilisables dans le contexte industriel. Deux de ces classifications sont décrites ci-après (§ 3.2.a et 3.2.b).

3.2.a. Classification destinée à l'analyse de la fiabilité humaine

Swain et Guttmann [Swain & Guttmann 1983] se sont intéressés à l'évaluation de la fiabilité humaine appliquée aux opérateurs des centrales

nucléaires. Les auteurs utilisent une classification des erreurs limitée aux actions humaines en sortie uniquement. Swain et Guttmann considèrent l'opérateur comme centre de l'étude, donc les actions en sortie sont les actions qu'effectue l'opérateur sur le système. Le tableau I-4 reproduit cette classification qui ne s'intéresse qu'à la partie visible de l'erreur. Elle ne prend pas en compte les mécanismes internes qui l'ont déclenchée. De ce fait, elle ne permet pas de prévoir ni même prévenir efficacement l'occurrence de ces erreurs. Néanmoins, cette vision symptomatique de l'erreur permet aux concepteurs d'envisager systématiquement les procédures de récupération simples, qui s'intéressent principalement aux symptômes de l'erreur et assez peu à leurs causes internes.

Erreurs d'ommission
(1) Omission de la tâche complète
(2) Omission d'une étape de la tâche

Erreurs d'exécution
(1) Erreur de sélection
- Sélection du mauvais dispositif de contrôle
- Positionnement erroné d'un dispositif de contrôle (inversion, mauvaises connexions, etc.)
- Distribution du mauvais ordre ou information (oralement ou par écrit)

(2) Erreur de séquencement
(3) Erreur de temporisation
- Trop tôt
- Trop tard

(4) Erreur quantitative
- Pas assez
- Trop

Tableau I-4 : Classification des actions humaines erronées en sortie. Tableau traduit de [Swain et al. 1983].

La classification présentée précédemment ne constitue en fait qu'une infime partie des recherches de Swain et Guttmann. Ces auteurs se préoccupent principalement d'évaluer la probabilité d'erreur dans la réalisation d'une tâche. Pour cela ils ont élaboré la méthode THERP, sur laquelle nous reviendrons dans la section 4 consacrée à l'analyse de la fiabilité humaine.

3.2.b. Classification de Nicolet, Carnino, et Wanner

Nicolet, Carnino, et Wanner proposent une classification comprenant sept erreurs type [Nicolet et al. 1990]. Les auteurs reconnaissent que ce découpage est quelque peu arbitraire, la plupart des défaillances humaines présentant entre elles un certain degré de dépendance :

• *Les erreurs de perception* sont multiples et ont des causes variées. Un opérateur peut ne pas percevoir une information ou un signal car celui-ci peut être fugace, masqué, peu contrasté, ou encore noyé dans un grand nombre de stimuli. Un opérateur peut également ne pas percevoir un signal de façon consciente et l'acquitter de manière automatique. Par exemple, un utilisateur habitué à ce que son ordinateur lui demande confirmation de chaque destruction de fichier finira par acquitter cette confirmation sans même lire le message d'avertissement consciemment. Un jour, croyant détruire un seul fichier l'utilisateur détruira l'ensemble de son disque, malgré la présence d'un message d'avertissement différent cette fois-ci.

• *Les erreurs de décodage* sont dues à une mauvaise interprétation d'un signal correctement reçu. L'utilisation des verrines lumineuses rouge et verte en est un exemple intéressant. L'interprétation des feux routiers rouge et vert ne provoque fort heureusement pas de problèmes d'interprétation. Néanmoins sur un appareil électrique, le rouge peut signifier qu'il est en fonctionnement, comme les thermostats des fours électriques, ou bien qu'une panne est apparue, comme pour les congélateurs. L'interprétation des couleurs porte souvent à confusion, car l'association entre les couleurs et leur signification est arbitraire.

• *Les erreurs dues au non-respect d'une procédure ou d'une réglementation* ne sont pas limitées au domaine aéronautique ou nucléaire. Elles font aussi partie de la vie quotidienne. Les opérateurs, en particulier les experts, estiment mal les risques et ont tendance à simplifier les procédures car "l'expérience prouve" que rien n'arrive jamais. C'est le cas en particulier des voyageurs de chemin de fer qui, habitués à ce qu'aucun train ne circule habituellement à l'heure où ils descendent du train, traversent les voies au lieu d'emprunter le passage souterrain. Un jour[7], exceptionnellement, un express ayant dix minutes de retard est passé sur la voie utilisée par ces voyageurs imprudents…

[7] Cet accident s'est malheureusement réellement produit.

• *Les erreurs de modèle ou de représentation* sont parmi les erreurs les plus dangereuses car difficiles à détecter par leur auteur. Elles peuvent en effet prendre un caractère “diabolique”, l’opérateur ayant tendance à faire “coller” les signaux qu’il reçoit à son modèle mental, rendant ainsi l’invalidation de ce dernier très difficile. Les modèles sont nos représentations mentales de la réalité qui nous entoure, mais ils n’en sont que la caricature. Cette représentation permet à l’opérateur de synthétiser les informations reçues, de les corréler, de les interpréter en fonction de ce qu’il connaît d’un système. Si le modèle est trop éloigné de la réalité du système, alors la compréhension des phénomènes est entachée d’erreur. Ce fut le cas en particulier de bon nombre de problèmes de biologie et de médecine avant que le modèle de la génération spontanée ne fût rejeté par Pasteur (1822-1895).

• *Les erreurs de communication homme-homme* semblent assez éloignées des préoccupations des concepteurs d’interfaces homme-machine. Néanmoins la communication homme-homme devient de plus en plus médiatisée [Salber 1995]. L’interface joue alors un rôle primordial dans le bon déroulement de cette communication. La première cause d’erreur est le bruitage du canal de communication, dû par exemple à un mauvais support technique. Mais ce n’est pas la seule cause. La communication homme-homme suppose une connaissance partagée entre les interlocuteurs. Par exemple, un certain nombre d’acronymes, ou la langue doivent faire partie des connaissances communes. Si l’un des interlocuteurs ne partage pas, ou ne partage que partiellement cette connaissance, il peut faire une interprétation erronée du message transmis. Le signe PAF par exemple peut signifier selon le contexte Police de l’Air et des Frontières, Patrouille Aérienne de France, Paysage Audiovisuel Français, ou encore Participation Aux Frais !

• *L’absence de prise de décision* en temps voulu est à rapprocher de l’adage qui affirme qu’une mauvaise décision prise à temps vaut mieux qu’une bonne décision prise trop tard. Le facteur temps joue un rôle primordial dans les systèmes dynamiques. Il peut en effet transformer une situation critique en une situation catastrophique. Parmi les exemples, nous pouvons citer celui du pilote d’avion qui se déroute plusieurs fois à cause du mauvais temps, et qui finit par s’écraser à cause d’une panne sèche.

• *Les actions mal séquencées ou mal dosées* sont pour la plupart dues à des erreurs d'autres types survenues en amont. Ce type d'erreur peut prendre la forme de l'entrée d'une mauvaise valeur, ou de l'inversion de deux actions comme par exemple mettre en route une machine à café, avant d'avoir placé en-dessous la tasse que l'on a "oublié" dans l'autre main.

A mi-chemin entre une vision symptomatique et une vision uniquement basée sur les mécanismes internes, cette classification est un bon compromis entre les deux approches. Elle possède assez de théorie pour prédire les erreurs, tout en restant assez proche d'une vision symptomatique pour détecter facilement le type d'erreur auquel on est confronté.

3.2.c. Niveaux d'activité et types d'erreur

Les travaux de Rasmussen [Rasmussen 1986] et de Reason [Reason 1990], traduits en Français par Hoc [Reason 1993], font partie des contributions majeures dans le domaine de l'erreur humaine. Ces travaux s'appuient sur le modèle simplifié des trois niveaux de contrôle des actions humaines définis par Rasmussen. Ce modèle, reproduit figure I-5, distingue trois catégories de comportement humain :

• *Le comportement basé sur les habiletés* ("skill-based behavior"), est aussi appelé niveau des réflexes. Il représente les actions sensori-motrices effectuées par un opérateur suite à une intention mais qui se déroulent sans réel contrôle conscient. C'est le niveau le plus bas, au sens de la complexité de mise en œuvre cognitive, du comportement humain. C'est aussi un niveau qui peut engendrer des comportements extrêmement rapides et efficaces. La conduite automobile utilise de nombreux réflexes, comme le changement de vitesse, si difficile au novice, et si incroyablement facile après quelques centaines d'heures de conduite. Les comportements de ce niveau sont difficiles à apprendre, mais aussi très difficiles à oublier. Changez vous-même l'emplacement d'un objet utilisé couramment comme un réveil, ou bien passez d'une boîte de vitesse manuelle à une boîte automatique. Votre main cherchera le réveil à son ancien emplacement, tout comme votre pied cherchera sans succès la pédale d'embrayage à chaque fois que le régime moteur montera. C'est pourquoi il faut veiller à ne pas faire prendre à des opérateurs de "mauvaise habitudes" et même des habitudes contre nature. Les robinets s'ouvrent en les tournant dans le sens contraire des aiguilles d'une montre. Il est peu prudent d'inverser ce sens

dans une centrale nucléaire, car bon nombre d'opérateurs feront l'erreur. Les comportements de ce niveau sont quelquefois basés sur la réponse à un retour d'information simple, comme les tâches de poursuite. Le suivi d'une route en conduite automobile en est un bon exemple. Mais ce n'est pas le niveau des réflexes qui effectue le choix de l'itinéraire, ou qui choisit de ralentir à cause du risque de verglas, c'est le rôle du niveau des procédés.

• *Le comportement basé sur les règles* ("rule-based behavior"), est aussi appelé niveau des procédés. Selon Rasmussen, traduit par Hoc dans [Reason 1993] page 94, ‹‹ L'activité est dirigée par un but, mais elle est structurée par un contrôle proactif, grâce à une règle stockée. Très souvent, le but n'est même pas formulé explicitement, mais on le trouve implicitement dans la situation, et il libère les règles stockées. Le contrôle est téologique[8], en ce sens que la règle ou le contrôle sont sélectionnés sur la base d'expériences antérieurement réussies. Le contrôle évolue selon le principe de la priorité à la règle la plus adaptée ››. Les comportements de ce niveau utilisent des règles pré-stockées instanciées par la situation présente. Ce type de comportement est conscient, et l'opérateur comme l'environnement fournissent les paramètres de la règle. L'opérateur n'a pas besoin d'avoir déjà vécu une situation identique, mais il utilise une règle qui est valable dans une situation analogue. Il peut avoir vécu cette situation, l'avoir apprise, ou encore l'avoir planifiée au niveau le plus haut que nous allons évoquer ci-après. Le choix de la règle à appliquer utilise le principe de la priorité à la règle la plus adaptée selon les données sur la situation. La limite entre les niveaux des réflexes et des procédés n'est pas nettement définie, et dépend de l'entraînement de l'opérateur tout comme de son attention. En règle générale, il est possible de faire la distinction selon que l'opérateur peut ou non rapporter sans difficulté l'ensemble des actions qu'il a effectuées.

• *Le comportement basé sur les connaissances* ("knowledge-based behavior"), est aussi appelé niveau du savoir. Ce niveau de comportement, le plus haut cognitivement, est utilisé lorsque qu'aucun réflexe ou aucune règle ne convient à la situation rencontrée par l'opérateur. Ce comportement est utilisé lorsque l'opérateur est confronté à des situations non familières ou bien totalement nouvelles. Pour ces situations, le but est clairement exprimé par l'opérateur en fonction de son analyse de

8 Ne pas confondre avec théologique.

l'environnement et par ses objectifs. Les traitements effectués à ce niveau sont conscients, lents, séquentiels, et fastidieux. Ils s'apparentent à un processus de résolution de problème qui aboutit à la génération d'un plan adapté à la situation. À ce niveau de raisonnement, la structure du comportement de l'opérateur s'appuie sur un modèle mental de la situation. Selon Rasmussen, ce type de comportement pourrait aussi être appelé "model-based behavior", c'est-à-dire comportement basé sur les modèles. L'opérateur peut définir son plan d'action soit en utilisant une méthode d'essai-erreur, soit en utilisant une méthode plus conceptuelle par la compréhension des propriétés fonctionnelles de l'environnement.

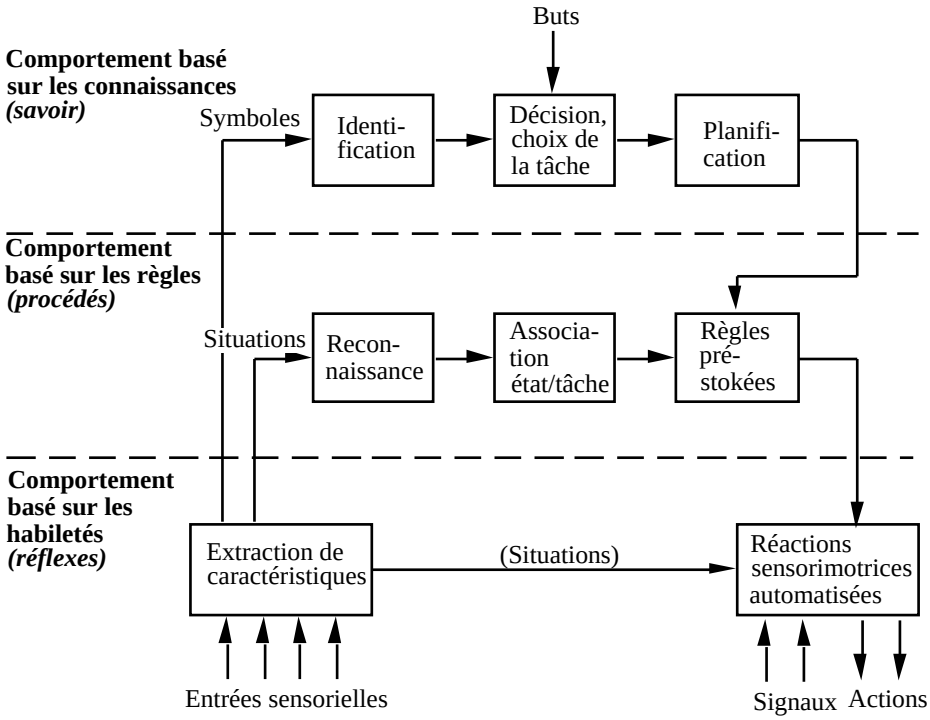

Figure I-5 : Modèle simplifié des trois niveaux de contrôle des comportements humains. Schéma traduit de [Rasmussen 1986].

Selon le modèle de Rasmussen, le comportement humain utilise de manière simultanée ces trois niveaux de comportement. Le niveau du savoir, le plus haut, définit les buts tandis que les deux autres se chargent de les satisfaire en fonction de leurs compétences. Par exemple, lorsque l'on cherche son chemin en voiture dans une ville inconnue, le niveau du savoir lit la carte et essaye de planifier une route. S'il faut faire demi-tour, c'est le niveau des procédés qui déterminera la procédure à utiliser en fonction de l'environnement, tandis que le niveau réflexes s'occupera de changer les vitesses, de contrôler la trajectoire du véhicule, et de surveiller le trafic.

Remarquons certaines similitudes entre les trois niveaux du modèle de l'opérateur humain de Rasmussen et la notion de cerveau triunique définie par MacLean, dont les travaux ont été récemment rassemblés et traduits par Guyot [MacLean & Guyot 1990]. Cette ancienne théorie originaire du domaine de la biologie animale suppose que l'évolution du cerveau humain s'est effectué par la superposition de trois couches successives, qui peuvent par ailleurs se visualiser par coloration. La figure I-6 illustre cette théorie. Bien que très peu d'éléments permettent de justifier une correspondance entre les deux modèles, il est intéressant de tenter de rapprocher ces deux théories en provenance de deux domaines de recherche fort éloignés :

• Le *néo-cortex* est le siège de l'esprit rationnel. Ce niveau peut être comparé au niveau du savoir du modèle de Rasmussen.

• Le *système limbique* est le siège des émotions, de la mémoire et de l'identité personnelle. Il peut être associé au niveau basé sur les règles du modèle de Rasmussen.

• Le *complexe R* (pour reptilien) est le siège des instincts, mais aussi de l'imitation, de la routine. Il peut se voir comme le niveau réflexe du modèle de Rasmussen.

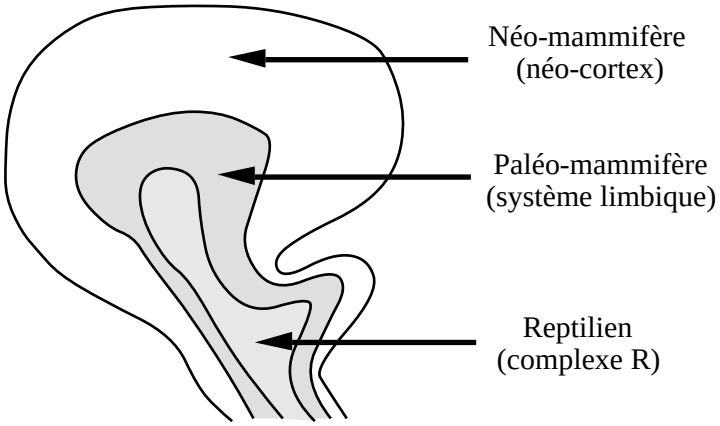

Figure I-6 : Représentation de l'organisation hiérarchique des trois cerveaux de l'homme [MacLean et al. 1990].

À chaque niveau du modèle de Rasmussen sont associés des mécanismes d'erreur. La figure I-7 reprend le modèle de Rasmussen de la figure I-5 en y ajoutant ces mécanismes d'erreur et leurs relations avec le contrôle comportemental. Selon Rasmussen, ils peuvent se classer en deux catégories :

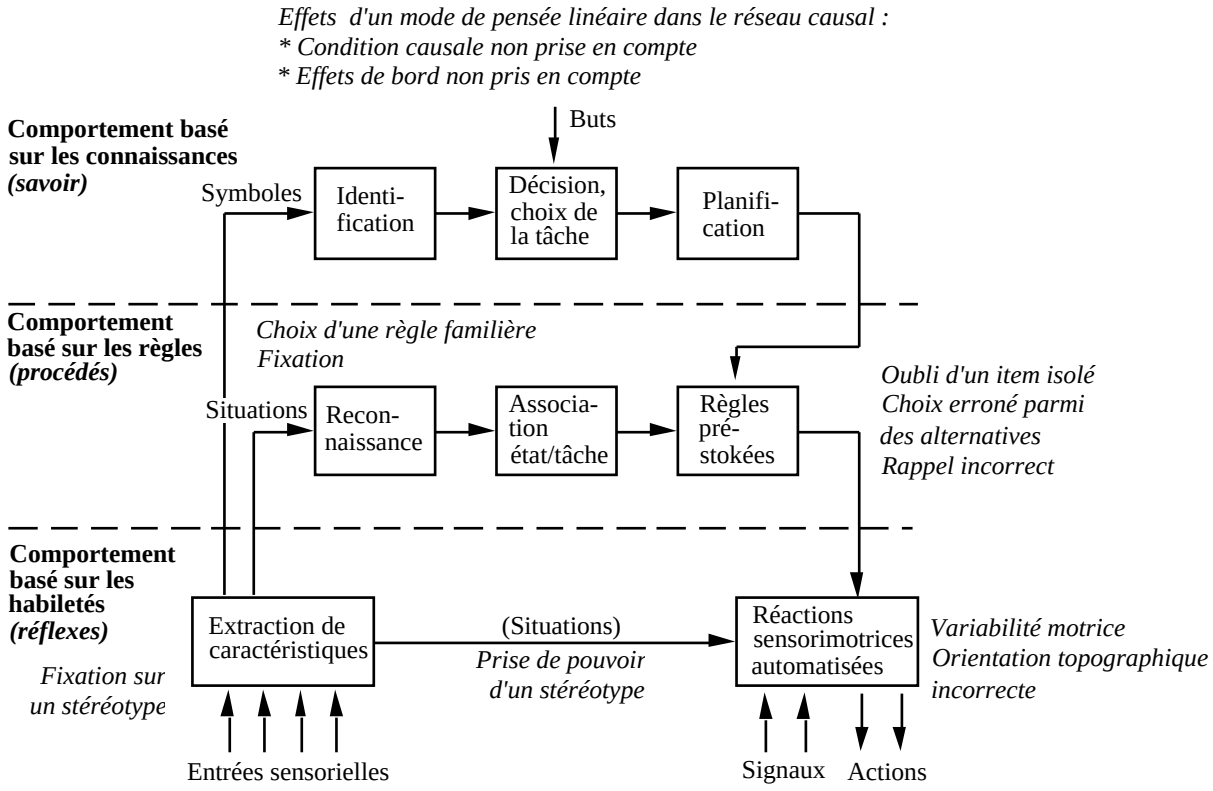

Figure I-7 : Mécanismes typiques des "erreurs" humaines et leurs relations avec le contrôle comportemental. Schéma traduit de [Rasmussen 1986].

‹ La ***variabilité humaine au cours de tâches familières*** comprend six formes :

• La *variabilité motrice*, situé au niveau des réflexes, peut prendre la forme d'un manque de précision lors des actions de l'opérateur. Mais aussi plus simplement d'un mauvais dosage de la force musculaire. Ce mécanisme d'erreur peut conduire, par exemple, à la rupture d'une commande.

• L'*orientation topographique incorrecte* est due à une désynchronisation entre le monde réel et le modèle mental de l'opérateur. Un opérateur peut alors confondre, par exemple, entre les deux sens de déplacement d'un levier.

• La *prise de pouvoir d'un stéréotype* est un mécanisme d'erreur très courant et facile à détecter. Ainsi, il n'est pas rare de voir certains conducteurs, sains de corps et d'esprit, prendre leur voiture un matin pour partir en congé, et, se "retrouver" sur le parking devant leur bureau. Ce type d'erreur se produit fréquemment lorsque le chemin suivi est au départ identique à l'habitude. Le comportement basé sur le niveau réflexe a pris le

pouvoir sur le niveau du savoir, le conducteur s'étant laissé "porté" par l'habitude.

• La variabilité humaine au niveau des règles regroupe trois formes : le *choix erroné parmi des alternatives*, le *rappel incorrect d'une règle*, et l'*oubli d'un item isolé*. Cette dernière est la plus fréquente. Elle se caractérise le plus souvent par l'omission d'une tâche secondaire au cours de l'exécution d'une procédure. Cette tâche secondaire, car non vitale à l'accomplissement du but de la procédure, est néanmoins obligatoire. Par exemple, il est courant de mettre à bouillir de l'eau sur une cuisinière, et, d'oublier d'éteindre le gaz après avoir retiré la bouilloire. La tâche "éteindre le gaz" n'est en effet pas nécessaire à l'accomplissement du but "obtenir de l'eau bouillante".

› La ***mauvaise adaptation humaine aux changements du système*** comprend quatre formes :

• La *fixation sur un stéréotype* est à rapprocher de la prise de pouvoir d'un stéréotype évoquée au paragraphe précédent. Cependant, dans ce cas, c'est le contexte qui a changé, et non pas le but de l'opérateur. Ce mécanisme se situe au niveau des réflexes. Par exemple, un conducteur évitera de freiner brusquement sur une route gelée. Néanmoins, il est fort probable qu'il freinera tout de même brutalement si un animal traverse inopinément devant son véhicule.

• De la même manière, au niveau des procédés, une *fixation* peut avoir lieu sur une règle qui ne correspond pas au contexte de la situation.

• Le *choix d'une règle familière* est due à la sélection d'un niveau de comportement inférieur à celui que la situation réclame. L'opérateur humain a tendance à limiter au maximum sa charge cognitive. Or celle-ci est plus importante au niveau du savoir qu'au niveau des procédés. Ce comportement se remarque le plus souvent au sein des tâches de résolution de panne où l'opérateur pense avoir affaire à une situation connue et applique la procédure associée.

• Les *effets d'un mode de pensée linéaire dans un réseau causal* sont associés au niveau du savoir. Rasmussen identifie deux grands mécanismes d'erreur dus à un manque de prise en compte des conditions causales ou des effets de bords. En d'autres termes, le raisonnement échoue soit par

oubli des causes d'un problème, soit par ignorance des conséquences des actions planifiées.

Activité basée sur les automatismes	
Inattention	***Attention excessive***
Ratés de double capture Omission suivant des interruptions Intentionnalité réduite Confusions perceptives Erreurs d'inférence	Omissions Répétitions Inversions
Activité basée sur les règles	
Mauvaise application de règles correctes	***Application de règles erronées***
Premières exceptions Contresignes et non-signes Surcharge d'information Force de la règle Règles générales Redondance Rigidité	Déficiences d'encodage Déficiences de l'action Règles erronées Règles inélégantes Règles déconseillées
Activité basée sur les connaissances déclaratives	
Sélectivité Limitations de l'espace de travail Hors de la vue, hors de l'esprit Biais de confirmation Confiance excessive Révision biaisée Corrélation illusoire Effets de halo Difficultés avec la causalité Difficulté avec la complexité Difficultés avec la rétroaction différée Prise en considération insuffisante des processus au bon moment Difficultés avec les développements exponentiels Raisonnement sur les séquences causales au lieu de réseaux causaux Vagabondage thématique Fixation	

Tableau I-8 : Résumé des principales catégories de classification des modes de défaillance à chacun des trois niveaux d'activité [Reason 1993].

Nous venons de décrire des mécanismes d'erreur, et non des types d'erreur. Cependant Rasmussen propose également une classification multifacette des types d'erreurs qui reprend en grande partie les mécanismes décrits au paragraphe précédent. De même, Reason s'appuie sur le modèle simplifié des comportements humains de Rasmussen pour définir une classification des erreurs. En outre, cette classification des erreurs est plus précise que celle de Rasmussen. Le tableau I-8 la reproduit intégralement. Autant la classification de Rasmussen est assez aisée à comprendre, autant celle de Reason nécessite

une lecture attentive des définitions de chacun des types d'erreur [Reason 1993]. Cette difficulté limite de fait l'utilisation de telles classifications par les praticiens et rend donc difficile leur intégration au sein du processus de conception.

Le modèle de Rasmussen nous aide peu à comprendre les mécanismes d'erreur du niveau cognitif. Le modèle ICS est plus adapté, il fait l'objet du paragraphe suivant (§ 3.2.d).

3.2.d. Interacting Cognitive Subsystems (ICS)

Le modèle ICS est avant tout un modèle cognitif de l'opérateur humain [Barnard 1985]. ICS peut être vu comme un affinement du modèle du processeur humain. Il est composé de neufs sous-systèmes spécialisés, interconnectées, et possédant chacun une mémoire locale. On distingue deux classes de sous-systèmes :

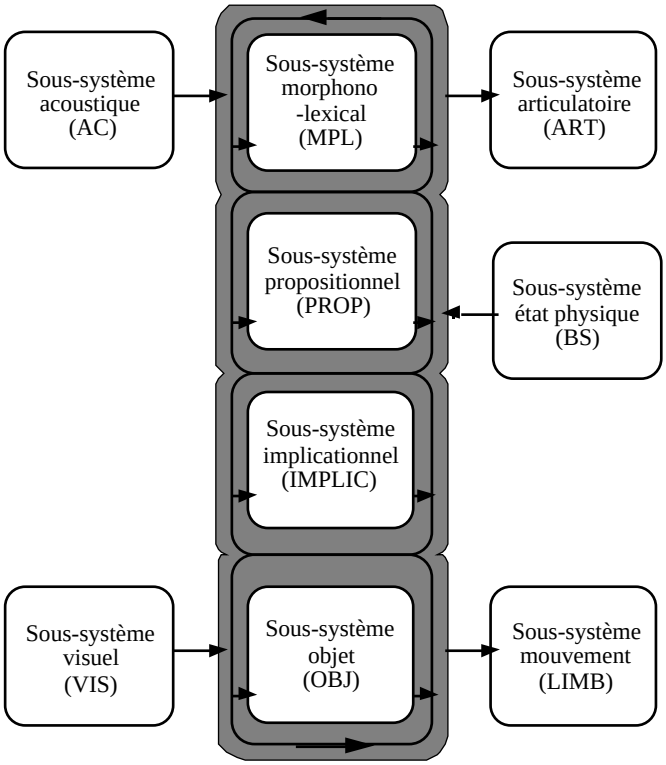

Figure I-9 : Schéma simplifié de ICS [Salber 1995].

› Les sous-systèmes périphériques perceptifs et moteurs qui acquièrent l'information (visuel [VIS], acoustique [AC], état physique [BS]) et agissent sur l'environnement (articulatoire [ART] et mouvement [LIM]).

› Les systèmes centraux effectuent le traitement des informations et modélisent le raisonnement (morphono-lexical [MLP], propositionnel [PROP], implicationnel [IMPLIC], et objet [OBJ]).

Tous les sous-systèmes sont reliés entre eux par un réseau de communication comme le montre la figure I-9. Le fonctionnement de ICS est comparable à celui d'un système multiprocesseur à entités spécialisées. Chaque sous-système est responsable d'un certain nombre de traitements sur les données qu'il reçoit. Il effectue des transformations sur ces données avant

de les communiquer à un autre sous-système. C'est l'ensemble des ces transformations et traitements qui modélise les comportements cognitifs.

Bien que ICS n'ait pas été prévu pour expliquer l'erreur humaine, il peut néanmoins être utilisé dans ce but. Par exemple, Salber a utilisé les règles associées à ICS pour interpréter certaines des erreurs constatées au cours de l'expérience de travail coopératif "Garden Movie" [Salber 1995]. Un sous-système d'ICS peut être amené à combiner deux flots d'information, une séquence vidéo et un espace de travail par exemple. Si ces deux flots ne sont pas cohérents, la vue vidéo était inversée dans une des expérience de "Garden Movie", la fusion risque de s'effectuer en donnant des résultats incorrects. En effet, l'une des règles de fusion de ICS impose la cohérence des flux d'information dans un sous-système. En conséquence, l'interprétation par le sujet d'une instruction vidéo "met cela à droite" peut être inversée. Il existe probablement d'autres types d'erreur explicables par le modèle ICS. Néanmoins ICS est plus explicatif que prédictif car son utilisation est difficile et demande une bonne connaissance des heuristiques associées au modèle.

3.3. Aspect multifacette de l'erreur humaine

La nature de l'erreur pose tout d'abord un problème épineux. La définition de Reason, conditionnant l'erreur à l'intention de l'opérateur, semble être la plus juste du point de vue de l'interaction homme-machine. Cependant, l'intention de l'opérateur est une donnée non disponible lors de la phase de conception d'une interface. Les concepteurs ne disposent dans le meilleur des cas que d'une analyse de tâche. Cette analyse comporte habituellement une série de procédures, correspondant aux buts supposés de l'opérateur. Implicitement, intentions et buts de l'opérateur sont considérés comme identiques. Or, ils ne le sont pas nécessairement. À ce propos, la méthode DIANE fait la distinction entre la tâche prescrite, celle définie par les concepteurs en fonction de l'analyse de tâche avec ses tâches annexes, la tâche minimale qui est un compromis entre sécurité et efficacité, et la tâche effective, celle réalisée effectivement par les opérateurs [Barthet 1988]. Cette dernière n'est définie qu'a posteriori, une fois l'interface ou sa maquette réalisée. Finalement, du point de vue du concepteur, l'erreur ne peut se concevoir que comme une déviation de la tâche normale prévue pour atteindre un but donné. Cette définition est très restrictive. Elle correspond en fait à la première définition de l'erreur donnée au paragraphe 3.1. La tâche du concepteur est alors de définir suffisamment d'alternatives pour la résolution

d'un but afin de couvrir les différentes voies que l'utilisateur est susceptible emprunter.

L'aspect multifacette de l'erreur humaine se remarque, aussi bien en ce qui concerne sa définition ci-dessus, que ses classifications. La meta-classification de Reason, décrite au paragraphe 3.2, effectue un premier tri des différentes classifications de l'erreur en fonction des besoins de leurs auteurs. Malgré un aspect qui semblerait à première vue anarchique, les différents modèles proposés se recoupent au moins en partie. Par exemple, le modèle de Rasmussen évoque "l'oubli d'un item isolé" à la figure I-7 ; il se rapproche de "l'erreur d'omission" de Swain et Guttmann à la figure I-4. Sans nul doute les modèles comme celui de Rasmussen et ICS font progresser notre connaissance en ce domaine. Mais ces modèles ne sont pas exploitables directement. Seules les heuristiques le sont à l'heure actuelle. En effet, comme indiqué dans l'introduction de ce mémoire, l'approche science appliquée est aujourd'hui la plus réaliste en conception d'interfaces homme-machine. C'est pourquoi le but de ces modèles doit être avant tout de fournir aux concepteurs des heuristiques. Ce sont ces heuristiques que nous utiliserons prioritairement dans la suite de notre démarche de recherche, même si les modèles doivent toujours être gardés pour mémoire.

4. Gérer l'erreur humaine

Il est intéressant de constater l'émergence d'un consensus, aussi bien dans le domaine de la recherche que du côté judiciaire, selon lequel l'erreur d'un opérateur n'est en fait que le dernier maillon d'une longue suite d'erreurs. L'étude des erreurs latentes au sein des organisations se retrouve dans de nombreuses contributions comme celle de Reason [Reason 1993], de Nicolet, Carnino, et Wanner [Nicolet et al. 1990], mais aussi dans une moindre mesure, avec les critères de Swain et Guttmann [Swain et al. 1983]. Selon Reason, un accident ne peut avoir lieu que s'il existe une trajectoire traversant l'ensemble des défenses du système. Ces défenses peuvent présenter des lacunes, représentées par les trous, mais un accident ne se produit que si l'ensemble de ces lacunes se font face. Reason schématise figure I-10 la distribution des responsabilités au sein des organisations.

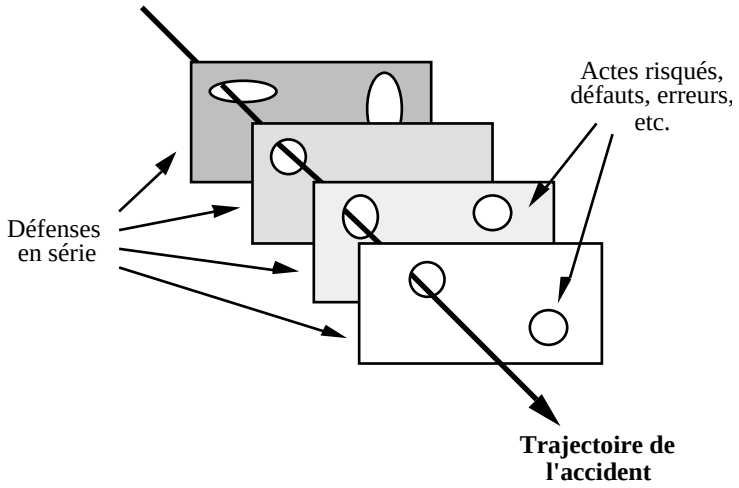

Figure I-10 : Trajectoire d'un accident. Adapté de [Reason 1993].

Ainsi, malgré des manquements aux règlements dans le but d'augmenter l'efficacité, des erreurs de la part d'opérateurs, ou encore des défauts d'organisation de la part de l'encadrement, il n'y a le plus souvent aucune conséquence grave à ces actes. Les nombreuses défenses en série et la robustesse d'un système ont donc un effet pervers : elles renforcent le sentiment des opérateurs et de tous les intervenants du système, que l'erreur commise est sans conséquence. Dès lors, les procédures de sécurité risquent de ne plus être appliquées à tous les niveaux de la hiérarchie. Les manquements aux règles se multipliant, il suffit qu'ils permettent à une trajectoire accidentelle d'apparaître, et c'est la catastrophe.

L'usage quasiment systématique de simulateurs réalistes permet de faire prendre conscience aux opérateurs des risques encourus [Nicolet et al. 1990]. Mais les actions en amont sont aussi très importantes. En effet, une automatisation utilisée à mauvais escient doublée d'une interface homme-machine mal conçue sont en cause dans de nombreuses catastrophes. Ces défauts de conception font partie des erreurs latentes au sein des organisations qui ouvrent la voie à une trajectoire d'erreur. La responsabilité des concepteurs est alors engagée.

Cependant, l'étude de l'erreur humaine est complexe, et jusqu'à présent le processus conception des interfaces homme-machine ne la prenait que peu souvent en compte. L'erreur est ignorée par les concepteurs car elle est méconnue, il semble aussi que l'on considère l'erreur comme une fatalité. Fort heureusement ce point de vue change, malheureusement aussi grâce à des catastrophes, mais l'importance de l'erreur humaine est maintenant admise et

fait doucement son chemin parmi ceux qui conçoivent les interfaces de demain. Les méthodes permettant de prendre en compte l'erreur humaine existent. Même si elles sont limitées, leur application au processus de conception des interfaces homme-machine est envisageable. Elle fait l'objet du chapitre suivant.

5. Bibliographie

[Airbus Industrie 1992] Airbus Industrie. *Flight Crew Operating Manual A320*. 1992.

[Amalberti 1992] Amalberti R. Sécurité des vols et automatisation des cockpits. *Le transpondeur: bulletin de laison*, Avril 1992. n° 7, p. 7-16.

[Balbo 1994] Balbo S. *Un pas vers l'évaluation automatique des interfaces homme-machine*. Thèse en Informatique : Université Joseph Fourier (Grenoble I), 5 Septembre 1994.

[Barnard 1985] Barnard J.P. *Cognitive resources and the learning of computer dialogs*. Interfacing throught, Cognitive aspect of Human-Computer Interaction, MIT Press, 1985. p. 112-158.

[Barthet 1988] Barthet M.-F. *Logiciels interactifs et ergonomie : Modèles et méthodes de conception*. Paris, France : Dunod, 1988.

[Bertaud du Chazaud 1994] Bertaud du Chazaud H. *Le dictionnaire des synonymes et contraires*. Paris, France : Le Robert, 1994.

[Billings 1991] Billings C.E. *Human-centered aircraft automation: A concept and guidlines*. NASA Ames research center, August 1991. Technical memorandum NASA-TM-103885.

[Billings 1996] Billings C.E. *Human-centered aviation automation: Principles and guidlines*. NASA Ames research center, February 1996. NASA Technical Memorandum NASA-TM-110381.

[Boy 1993] Boy G. Knowledge acquisition in dynamic systems: how can logicism and situatedness go together? *EKAW'93*, 1993.

[Boy et al. 1993] Boy G. & Hollnagel E. *Cockpit evolution and human-centered automation*. EURISCO, October 1993. Technical Report T-93-002-GB-VI.

[Chevalier 1992] Chevalier G. Les retombées d'un nuage d'innovations. *Les Cahiers de Science & Vie : Concorde / La puissance du rêve, le rêve de la puissance*, Juin 1992. n° 9, p. 72-76.

[Dorwell et al. 1989] Dorwell J. & Long J. Towards a conception for an engineering discipline of human factors. *Ergonomics*, November 1989. vol. 32, n° 11, p. 1513-1535.

[Dupont 1994] Dupont J. Le dossier des commandes de vol électriques. *Air & Cosmos Aviation International*, 1994. n° 1452/53, p. 61-73.

[Grau et al. 1995] Grau J.-Y. & Doireau P. *Erreur humaine et automatisation des cockpits*. Centre d'Études et de Recherche de Médecine Aérospatiale, Octobre 1995. Rapport de recherche 95-33.

[Grudin 1988] Grudin J. Why CSCW applications fail: problems in the design and evaluation of organisation interface. *Proceedings of the ACM conference on Computer-Supported Cooperative Work, Portland (Oregon), USA*, September 1988. p. 85-93.

[Hollnagel et al. 1993] Hollnagel E., Warren C., & Cacciabue P.C. Automation in aerospace and aviation applications: Where are the limits. *Human-Machine Interaction and Artificial Intelligence in Aerospace (HMI-AI-AS'93) Workshop, Toulouse, France*, September 28-30 1993.

[Laronche 1993] Laronche M. *Les suites de la catastrophe du mont Sainte-Odile*. Le Monde. Paris, France : Samedi 18 décembre 1993. p. 24.

[MacLean et al. 1990] MacLean P.D. & Guyot R. *Les trois cerveaux de l'homme*. Paris, France : Robert Laffont, 1990.

[Mellor 1993] Mellor P. *Review of AMJ 25.1303 and related documents*. Center for Software Reliability, City University, 1993. News posted in "sci.aeronautics.airliners" newsgroup.

[Nicolet et al. 1990] Nicolet J.-L., Carnino A., & Wanner J.-C. *Catastrophes ? Non merci ! La prévention des risques technologiques et humains*. Éditions Masson, 1990.

[Norman 1991] Norman D.A. Cognitive science in the cockpit. *Gateway (Crew System Information Analysis Center - University of Dayton Research Institute)*, Spring 1991. vol. II, n° 2, p. 1-6.

[Rasmussen 1986] Rasmussen J. *Information processing and Human-Machine Interaction : An approach to cognitive engineering*. North-Holland, 1986.

[Reason 1990] Reason J. *Human error*. Cambridge University Press, 1990.

[Reason 1993] Reason J. *L'erreur humaine*. Paris, France : Presses Universitaires de France, 1993.

[Robert 1992] Robert P. *Le petit Robert : Dictionnaire de la langue française*. Paris, France : Le Robert, 1992.

[Salber 1995] Salber D. *De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs : L'exemple de la communication homme-homme médiatisée*. Thèse en informatique : Université Joseph Fourier (Grenoble I), 1995.

[Swain et al. 1983] Swain A.D. & Guttmann H.E. *Handbook of human reliability analysis with emphasis on nuclear power plant applications*. Nuclar Regulatory Commission (NUREG), August 1983. Final report NUREG/CR-1278F.

[Sweet 1995] Sweet W. The glass cockpit. *IEEE Spectrum*, September 1995. p. 30-38.

[Tarby 1993] Tarby J.-C. *Gestion automatique du dialogue homme-machine à partir de spécifiactions conceptuelles*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1993.

[Wanner 1992] Wanner J.-C. Y a-t-il encore un pilote dans l'avion ? *Le transpondeur: bulletin de laison*, Avril 1992. n° 7, p. 51-54.

[Wiener 1989] Wiener E.L. *Human factors of advanced technology (glass cockpits) transport aircraft*. National Aeronautics and Space Administration, June 1989. Report NASA-CR-17752.

[Wiener et al. 1980] Wiener E.L. & Curry R.E. *Flight-deck automation: Promises and problems*. National Aeronautics and Space Administration, June 1980. Report NASA-TM-81206.

CHAPITRE II

ERREUR HUMAINE ET INTERRUPTIONS

1. Introduction

Au chapitre précédent, nous avons mis l'accent sur les risques d'aggravation des conséquences des erreurs humaines, et sur la modification de leur nature en raison de l'automatisation croissante des systèmes. Nous avons également passé en revue les classifications existantes de la notion d'erreur. Cette analyse essentiellement taxinomique et descriptive nous a permis de comprendre les mécanismes provoquant l'occurrence des erreurs. Néanmoins, la compréhension des mécanismes de l'erreur humaine n'est pas suffisante. Il est également nécessaire de s'intéresser au point de vue du concepteur. En effet, celui-ci doit disposer d'heuristiques afin de prévenir l'erreur, comme de modèles destinés à réaliser les fonctions de correction. C'est à ces besoins, exprimés par les concepteurs d'interfaces homme-machine, que nous nous intéressons maintenant.

L'attitude des concepteurs d'interfaces homme-machine face à l'erreur humaine peut être résumée par le schéma figure II-1. En premier lieu, il est nécessaire d'évaluer le risque d'erreur. Une fois ce risque évalué, le concepteur peut se tourner vers deux approches complémentaires :

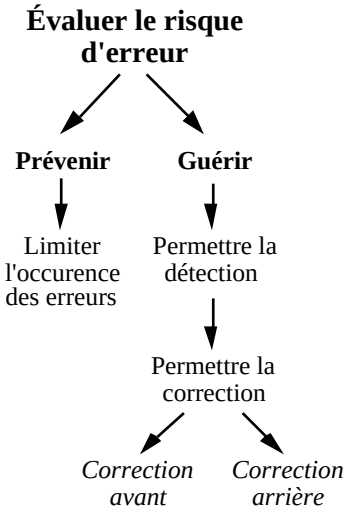

Figure II-1 : Erreurs humaines vis-à-vis de la conception des interfaces homme-machine.

- *Prévenir*, c'est-à-dire éviter l'occurrence de l'erreur par une conception adéquate de l'interface. Cette approche est a priori la meilleure car l'erreur est tout simplement écartée. Cependant les heuristiques à notre disposition sont limitées, et elles ne permettent que de réduire l'occurrence des erreurs.

- Il faut donc également *guérir*, c'est-à-dire minimiser les conséquences d'une erreur. Pour cela, le concepteur doit permettre à l'utilisateur de détecter son erreur, afin de pouvoir ensuite la corriger. Cette correction peut être une correction arrière ou bien une correction avant. Ces deux types de corrections sont détaillées à la section 3.

Le concepteur d'interfaces homme-machine dispose donc de plusieurs voies possibles pour prendre en compte l'erreur humaine au sein du processus

de conception. Nous étudierons à la section 2 successivement l'analyse, la prévention et la détection des erreurs. Les mécanismes de correction des erreurs font l'objet de la section 3. Enfin, les relations entre interruptions et erreurs sont regroupées dans la section 4 de ce chapitre.

2. Analyse, prévention, et détection des erreurs

L'erreur humaine ne doit pas être vue comme un comportement totalement imprévisible. L'analyse de la fiabilité humaine est possible au même titre que l'analyse de la fiabilité d'un système. Même si cette analyse, que nous évoquons au paragraphe 2.1, n'a pas la justesse des calculs de fiabilité utilisés pour les systèmes, elle apporte des informations non négligeables. De même, de nombreuses heuristiques et règles ont été définies dans le but de diminuer l'occurrence des erreurs. Le paragraphe 2.2 en dresse un inventaire non exhaustif. Et quand bien même l'erreur se produit, l'homme est faillible mais il sait détecter ses erreurs. Le paragraphe 2.3 est consacré à l'étude de ces mécanismes et des taux de détection d'erreur.

2.1. Analyse de la fiabilité humaine

L'analyse de la fiabilité humaine, dans le contexte de l'interaction avec un système, est en quelque sorte l'introduction du facteur humain dans les études probabilistes de la fiabilité des systèmes. L'analyse de la fiabilité humaine est encore une science jeune et peu aisée à mettre en œuvre. De plus, les données utilisées prennent peu en compte la capacité humaine à se corriger et donnent ainsi des résultats relativement pessimistes. Malgré toutes ses imperfections et limites, l'analyse de la fiabilité humaine est un passage obligé, car la contribution des facteurs humains dans l'estimation globale des risques peut aller, dans certains cas, jusqu'à 90%. Parmi les nombreuses méthodes disponibles, nous détaillons la méthode THERP (§ 2.1.a) qui est sans conteste la plus connue. En complément, nous évoquons la démarche SHARP (§ 2.1.b), destinée à guider le choix du praticien dans le dédale des méthodes disponibles. Nous évoquerons en complément deux méthodes d'analyse de l'erreur a priori (§ 2.1.c) et a posteriori (§ 2.1.d).

2.1.a. La méthode THERP

La méthode THERP ("Technique for Human Error Rate Prediction") est la plus usitée. Elle est décrite et expliquée dans un volumineux manuel, estimé à environ 600 pages, qui fait référence [Swain & Guttmann 1983]. Selon ses

auteurs, ‹‹ THERP est une méthode destinée à prédire les probabilités des erreurs humaines et à évaluer les dégradations d'un système homme-machine susceptibles d'être causées par les erreurs humaines, prises isolément ou en relation avec le fonctionnement des appareils, par les procédures opérationnelles ou les pratiques, ainsi que par les autres caractéristiques du système ou de l'homme qui influencent le comportement du système ››. La méthode THERP s'appuie sur une décomposition de la séquence incidentelle ou accidentelle en tâches élémentaires quantifiables. Les données associées aux tâches élémentaires permettent alors de calculer la probabilité d'incident ou accident de la séquence. Elle se décompose en quatre étapes dans le cas général, et en cinq si elle est utilisée par des concepteurs [Reason 1993] :

‹ Identification des fonctions du système qui peuvent être affectées par l'erreur humaine.

› Inventaire et analyse des actions humaines qui y sont liées (analyse détaillée de la tâche).

fi Estimation des probabilités pertinentes d'erreur en s'appuyant à la fois sur des jugements d'experts et sur des données disponibles.

fl Estimation des effets des erreurs humaines sur les défaillances du système.

° Si la méthode est utilisée par des concepteurs, on y ajoute une étape itérative qui comporte l'introduction de modifications dans le système, suivie d'un nouveau calcul des probabilités afin d'en mesurer leurs effets.

L'outil principal de la méthode THERP est une sorte d'arbre d'événements appelé diagramme en arbre des probabilités. Un tel arbre se compose de choix binaires représentant pour chaque événement la probabilité de succès ou d'échec. Appliquons cet outil à un distributeur automatique de boissons simplifié. On suppose que l'utilisateur peut faire deux types d'erreur : insérer un nombre trop faible de pièces, ou se tromper sur le choix de sa boisson. Il peut bien entendu détecter et corriger sa première erreur. Le schéma figure II-2 pourrait être le résultat d'une telle analyse. Les probabilités indiquées sur les branches de l'arbre sont données à titre d'exemple et ne sont pas issues d'une analyse rigoureuse du problème.

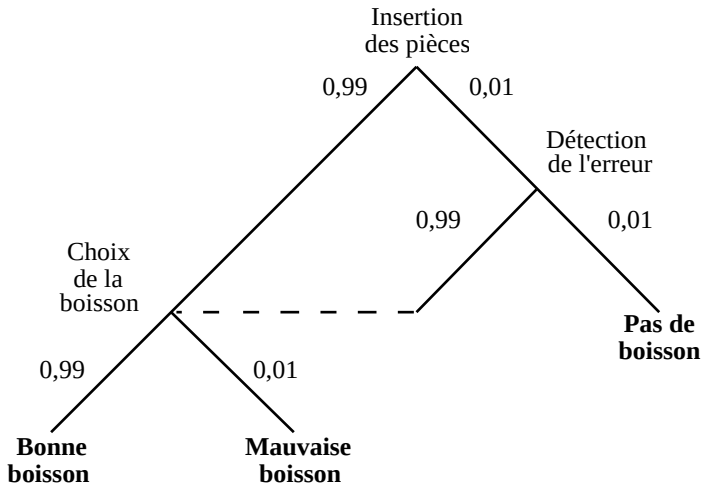

Figure II-2 : Exemple de diagramme en arbre des probabilités pour un distributeur automatique de boisson simplifié. Les branches de gauche indiquent les succès, et les branches de droite les échecs.

Les données sur la probabilité d'erreur sont fournies par une série de vingt-sept tableaux inclus dans le manuel de la méthode. Ces tableaux indiquent le plus souvent la probabilité que se produise au moins une erreur par heure lors de l'exécution d'une action physique ou cognitive ("Human Error Probability"), ainsi que le facteur d'erreur sur cette mesure ("Error Factor"). Celui-ci est calculé comme la racine carrée du rapport entre les limites d'incertitudes hautes et basses. Celles-ci encadrent la mesure de l'HEP dans laquelle 90% des sujets se conforment. L'accès aux informations contenues dans ces tableaux est complexe et demande une certaine expertise du domaine. Les tableaux sont regroupés en sept catégories :

- Balayage visuel
- Diagnostic
- Erreurs d'omission
- Erreurs d'exécution
- Facteurs modificateurs de la performance
- Limites d'incertitude
- Facteurs de correction

Ces catégories ne sont pas homogènes. Parmi celles-ci nous retrouvons les erreurs d'omission et d'exécution indiqués au paragraphe 3.2.a du chapitre I. Notons que le balayage visuel ou le diagnostic font référence à des tâches à part entière. De même, les facteurs de correction indiquent la probabilité de correction, suite à une tâche de vérification, des erreurs commises précédemment. Au contraire, d'autres catégories, et notamment les facteurs

modifiant la performance, ne peuvent être considérées comme des tâches ou des erreurs. Ce sont des facteurs destinés à adapter les données d'autres catégories en fonction du contexte. Par exemple, l'un de ces facteurs est le niveau de stress subit par l'opérateur. Enfin, les limites d'incertitude permettent de prédire la valeur du facteur d'erreur en fonction de l'HEP estimé pour des tâches génériques.

2.1.b. La démarche SHARP

D'autres méthodes, concurrentes ou complémentaires à THERP, peuvent être utilisées pour l'étude de la fiabilité humaine. Le praticien se trouve alors devant un choix difficile : quelle méthode utiliser, pour quels problèmes, et de manière à obtenir quels résultats ? Afin de répondre à ces questions, la démarche SHARP ("Systematic Human Action Reliability Procedure") définit sept étapes qui constituent un guide pour le concepteur [Nicolet, Carnino, & Wanner 1990] :

‹ Recherche et définition des interactions : il s'agit de répertorier toutes les séquences accidentelles conduisant aux différents dommages possibles.

› Sélection des interactions principales : seules les données de l'étape ‹ jugées importantes ou critiques sont retenues, de manière à limiter les coûts de l'étude.

fi Description détaillée de chacune des interactions sélectionnées : il s'agit de collecter le maximum d'informations afin d'être en mesure de décrire avec la plus grande précision chacune des interactions retenues.

fl Recherche des modèles les plus représentatifs : à chaque type d'interaction mis en évidence, l'analyste doit faire correspondre le modèle qui s'avère le plus représentatif. C'est ici que la méthode THERP peut intervenir.

° Étude le l'impact de chaque interaction sélectionnée : il s'agit d'analyser les conséquences éventuelles et l'impact de chaque interaction sélectionnée sur l'ensemble des arbres d'événements et de défaillance construits.

– Quantification de l'impact de chaque interaction retenue : il s'agit d'estimer les incertitudes et les marges associées à chaque interaction humaine.

† La documentation : étape indispensable destinée à justifier l'analyse des interactions retenues.

2.1.c. Analyse d'erreurs a priori

Gray et Johnson ont proposé l'utilisation de formalismes de description de tâches, en particulier UAN [Hartson, Siochi, & Hix 1990], afin de détecter, selon une série d'heuristiques, le risque d'erreur sur une tâche donnée [Gray & Johnson 1996]. Ces travaux, encore récents et donc limités, sont néanmoins prometteurs car ils permettent d'entrevoir la détection automatique, à partir d'une description de tâche, des cas classiques d'erreur. Ils permettraient ainsi de prévenir le concepteur d'un problème très en amont du cycle de conception d'une interface homme-machine.

2.1.d. Analyse d'erreurs a posteriori

L'analyse d'accidents de systèmes critiques temps-réel se heurte souvent à la complexité de la trajectoire accidentelle. C'est pour y remédier que Johnson et Telford proposent l'utilisation des réseaux de Petri pour l'analyse a posteriori des scénarii d'accidents [Johnson & Telford 1996]. En effet, les réseaux de Petri permettent de mettre en lumière l'enchaînement temporel et la combinaison des actions ayant conduit à un accident. Bien que cette analyse a posteriori n'ait que peu d'intérêt lors de la conception d'un système totalement nouveau, elle permet la compréhension des causes d'accidents, leur description formelle, et donc la réalisation plus aisée de modifications. De même, elle peut servir à la réalisation de scénarii d'accidents a priori. Cependant cette méthode étant avant tout descriptive, la réalisation automatique et systématique de ces scénarii semble difficilement envisageable.

2.2. Heuristiques de conception

Une fois la fiabilité humaine évaluée, le concepteur peut être amené à revoir sa conception, c'est la phase ° de la méthode THERP évoquée au paragraphe 2.1.a. Deux grandes classes d'heuristiques destinées à prévenir les erreurs se côtoient : des règles très proches de l'implémentation comme les "forcing functions" de Norman évoquées au paragraphe 2.2.a, ou bien des règles plus générales comme les règles de pilotabilité de Nicolet, Carnino, et Wanner évoquées au paragraphe 2.2.b. Nous comparerons ensuite, au paragraphe 2.2.c, ces règles spécifiques à l'erreur aux règles ergonomiques plus générales.

2.2.a. Les "forcing functions" de Norman

Norman [Norman 1990] propose une heuristique très intéressante destinée à réduire l'occurrence des erreurs : il propose d'étendre la notion de "forcing functions" que l'on peut traduire ici par "détrompeur". Cette idée se retrouve sous le nom de "fonction de contrainte" chez Reason [Reason 1993], et est aussi appelée de manière générale "retour d'information proactif". Cette heuristique de conception est très simple et astucieuse : il s'agit de forcer l'utilisateur à s'apercevoir de son erreur avant même de la commettre. Norman définit trois types de détrompeurs :

• Les interblocages ("Interlock") forcent l'utilisateur à ne pas oublier d'étapes dans une tâche. Par exemple, sur certains avions légers la commande du démarreur est cachée par le sélecteur de commande du robinet d'alimentation en carburant, lorsque celui-ci est en position fermée. Ainsi, le pilote ne peut accéder au démarreur que lorsque le réservoir est ouvert. Ceci évite à un avion de décoller avec le carburant encore présent dans les tuyauteries, pour ensuite tomber en panne sèche juste après le décollage [Avions Pierre Robin 1986].

• Les verrouillages actifs ("Lockin") empêchent un système de s'arrêter prématurément. C'est le cas en particulier des traitements de texte qui s'assurent que le ou les fichiers en cours d'édition sont bien sauvegardés, lorsque l'on quitte le programme.

• Les verrouillages passifs ("Lockout") sont destinés à empêcher l'utilisateur d'effectuer une opération dangereuse. Cette notion de détrompeur est présente en aéronautique sous la forme de boutons protégés par un petit clapet qu'il faut soulever pour y accéder. Ce verrouillage passif prévient le pilote qu'il active une fonction dangereuse, mais ne l'en empêche pas. De plus, ces clapets évitent l'activation de la fonction par inadvertance au cas où un objet quelconque heurterait le bouton. Par exemple, les commandes d'arrêt immédiat des moteurs disposent souvent d'un tel système. Elles sont activées seulement une fois l'aéronef au sol, ou en cas d'incendie.

De même, mais de manière plus autoritaire, cette fonction est utilisée pour les lecteurs de disquettes Macintosh™ : il n'est pas possible d'insérer complètement une disquette dans le lecteur dans le mauvais sens : par un jeu d'ergots, le chariot destiné à recevoir la disquette se bloque avant que

celle-ci ne soit complètement insérée. Ces détrompeurs ont tout de même quelques limites, car l'utilisateur n'est pas toujours averti du problème : il n'est pas rare de voir des disquettes insérées de force dans le lecteur par des débutants !

Appliquons maintenant la notion de détrompeur à un exemple de la vie courante : il n'est pas rare qu'un utilisateur oublie sa Télécarte[9] dans le lecteur de carte d'un Publiphone à la fin d'une conversation téléphonique. Afin de prévenir l'utilisateur, les Publiphones font retentir une série de bips environ deux secondes après que le combiné ait été raccroché. Si l'utilisateur est pressé, ces deux secondes suffisent pour qu'il soit déjà parti en oubliant sa Télécarte. Comme le montre la figure II-3, et en application de la notion de détrompeur, il suffirait de placer le lecteur de carte à l'emplacement de l'écouteur du combiné téléphonique, empêchant ainsi l'utilisateur de raccrocher le combiné si la carte est encore présente dans son lecteur.

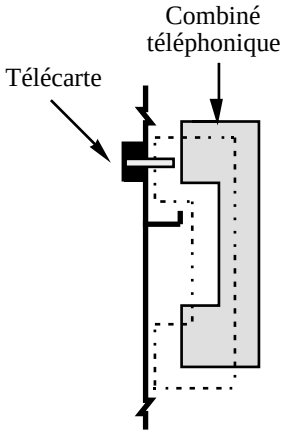

Figure II-3 : Application du principe du détrompeur au cas du Publiphone. 🕮

2.2.b. Règles de pilotabilité

Nicolet, Carnino, et Wanner proposent huit règles de pilotabilité d'un système destinées à diminuer le risque d'accident [Nicolet et al. 1990]. Inspirées de l'automatique, ces règles s'appuient sur plusieurs modèles de l'opérateur basés sur la notion de boucle de contrôle avec rétroaction. Leur utilisation souffre malheureusement des problèmes classiques des règles ergonomiques : elles peuvent être contradictoires, et sont assez difficiles à interpréter et à mettre en œuvre. De plus, de l'aveux même de leurs auteurs, elles sont non exhaustives. Néanmoins, ces règles ont le mérite d'exister et de permettre aux concepteurs de se rendre compte de problèmes potentiels.

• *Première règle* : fournir des informations utiles (utilisables et exactes). ‹‹ Il s'agit de fournir des informations en rapport direct avec la situation réelle du point de fonctionnement par rapport aux limites en tenant compte de l'état du système ››.

[9] Les Publiphones sont les téléphones public mis en place par France Télécom™. Ces téléphones utilisent une carte à puce appelée Télécarte.

• *Deuxième règle* : fournir des informations faciles à interpréter. ‹‹ Il ne faut pas obliger l'opérateur à effectuer des opérations mentales compliquées en vue de déduire la marge qu'il lui reste par rapport aux limites, l'écart par rapport aux consignes, ou l'état réel du système ››.

• *Troisième règle* : ne fournir que des informations utiles. ‹‹ Autrement dit, il est important de supprimer toutes les informations inutiles. Cette remarque est évidente, mais combien de fois oubliée. Il est néfaste d'augmenter la charge de travail par la lecture de paramètres inutiles, mais la tentation est grande pour le concepteur d'ajouter tel ou tel paramètre sous prétexte qu'il est facilement mesurable et qu'il "peut toujours servir" ››.

• *Quatrième règle* : fournir des informations permettant la prévision et la pré-action. ‹‹ Les informations de ce type sont avant tout des informations sur l'état du système, permettant de déterminer le sens et l'amplitude des variations probables. Disposant de ces données, l'opérateur peut alors prévoir l'évolution du système à l'aide de ses divers modèles de représentation ››.

• *Cinquième règle* : fournir de l'information. L'homme a besoin d'information, et il souffre de ne pas en recevoir. Si un système est figé à sa valeur nominale pendant des heures, l'attention de l'opérateur va rapidement se focaliser sur autre chose que ces paramètres fixes.

• *Sixième règle* : fournir des alarmes qui ne passent pas inaperçues. La conception des alarmes est une tâche très délicate. En effet, il faut minimiser les fausses alarmes qui incitent les opérateurs à ne plus prendre en compte les futures vraies alarmes. Pourtant, elles doivent être déclenchées suffisamment loin des limites pour permettre à l'opérateur de réagir. Enfin, il est nécessaire de supprimer les alarmes multiples ou "arbres de Noël" et de faciliter au maximum leur interprétation.

• *Septième règle* : limiter le nombre d'actions et faciliter l'élaboration des tactiques et stratégies. ‹‹ La limitation du nombre d'actions peut s'obtenir […] en automatisant toutes les actions tactiques simples où l'opérateur n'a qu'un rôle de servomécanisme, et en lui laissant les opérations de conduite ou stratégiques qui en général nécessitent peu d'actions sur les commandes ››.

• *Huitième règle* : étudier la localisation et le mode d'action des commandes. Cette règle fait partie du domaine de l'ergonomie au sens

strict. C'est-à-dire qu'elle s'intéresse à la compatibilité des dispositifs de commande avec la biomécanique humaine.

2.2.c. Règles ergonomiques générales

Toutes ces règles de conception et heuristiques sont particulièrement bien adaptées à la prévention des erreurs. Cependant, elles peuvent aussi être vues comme des heuristiques générales de conception. Elles sont en effet très proches de certains principes pratiques ergonomiques que l'on retrouve dans [Abowd, Coutaz, & Nigay 1992] tels l'observabilité, l'honnêteté, et la prédictabilité. De même, Bastien et Scapin définissent, parmi leurs critères ergonomiques pour l'évaluation, la *protection contre les erreurs* qui concerne ‹‹ les moyens mis en place pour détecter et prévenir les erreurs d'entrées de données ou de commandes ou les actions aux conséquences néfastes ›› [Bastien & Scapin 1993]. Nielsen s'intéresse également à la prévention des erreurs sous une forme similaire [Nielsen 1994]. Ces heuristiques de conception ne permettent pas d'éviter à coup sûr tout risque d'erreur. L'utilisateur en commettra donc, et doit en conséquence pouvoir les détecter. Les mécanismes alors mis en jeu font l'objet du paragraphe suivant de cette section.

2.3. Détection des erreurs

Les causes de l'erreur sont très étudiées, cependant ce qui se passe après son occurrence reste une friche de recherche. En effet, les mécanismes permettant la détection des erreurs restent encore mal connus. Il est cependant possible de dégager un certain nombre de résultats, grâce en particulier à la synthèse des travaux sur le domaine effectuée par Reason [Reason 1993].

2.3.a. Mécanismes de détection d'erreur

L'erreur peut être détectée selon trois mécanismes de base :

• *L'autocontrôle* permet à celui qui a commis l'erreur de la détecter par lui-même. Aux plus bas niveaux de comportement, cet autocontrôle prend la forme de boucles de rétroaction internes permettant par exemple le contrôle de la posture. Il semble qu'à tous les niveaux de comportement, l'homme dispose de modèles internes lui permettant de contrôler périodiquement l'adéquation entre son modèle et l'état de l'environnement escompté. En conséquence, l'interface doit permettre ce type de vérification par la mise à

disposition de l'utilisateur d'informations pertinentes. Les checklists en aéronautique sont une forme institutionnalisée d'autocontrôle.

• *La détection par des tiers* semble être l'un des seuls moyens efficaces de détecter les erreurs de diagnostic. Les diagnostics erronés ont tendance à persister même en présence d'informations contradictoires. Ce sont par exemple des erreurs de modèle définies par Nicolet, Carnino, et Wanner vues au paragraphe 3.2.c du chapitre I. C'est pourquoi des "esprits neufs" peuvent plus aisément réaliser un diagnostic correct car ils n'ont pas le même modèle que les équipes en place. Par exemple, la procédure habituellement utilisée au sein des équipages d'aéronefs est le contrôle systématique des actions de l'un des pilotes par l'autre.

• *La détection à partir d'indices de l'environnement* est celle qui nous intéresse plus particulièrement. En effet, ce type de détection est induit par l'environnement de la tâche, c'est-à-dire dans notre cas par l'interface homme-machine. Les retours d'information en provenance de l'interface permettent à l'utilisateur de se rendre compte de son erreur. Par exemple, grâce à l'aide des détrompeurs vus au paragraphe 2.2.a, l'utilisateur peut détecter immédiatement ses erreurs.

2.3.b. Taux de détection d'erreur

Peu de résultats réellement utilisables sont disponibles quant aux taux de détection d'erreur. Reason en fait une synthèse succincte [Reason 1993] page 229 dont il ressort que la détection des erreurs semble plus aisée aux niveaux bas du comportement humain. Le niveau basé sur les réflexes obtient le meilleur taux de détection. Celui-ci s'amenuiserait ensuite au niveau basé sur les règles pour devenir faible au niveau basé sur les connaissances déclaratives. On trouve un résultat inverse lorsque l'on s'intéresse au taux d'occurrence des erreurs. De plus, les actions basées sur les réflexes étant bien plus nombreuses que les actions basées sur les connaissances déclaratives, il en résulte que les erreurs basées sur les niveaux bas du comportement humain sont les plus nombreuses mais les mieux détectées. De même, il semble également que ce soient les plus faciles à corriger. C'est à cette correction des erreurs humaines que nous allons maintenant nous intéresser.

3. Correction des erreurs

Une fois l'erreur détectée, l'utilisateur doit pouvoir la corriger afin d'en minimiser les conséquences. Les définitions actuelles de la correction d'erreur, qui font l'objet paragraphe 3.1, permettent de faire la distinction entre correction avant et correction arrière. Cependant, pour les raisons que nous évoquons au paragraphe 3.2, cette distinction ne nous semble pas suffisante. C'est pourquoi nous proposons au paragraphe 3.3 une aide au choix des mécanismes de correction d'erreur. Enfin, le paragraphe 3.4 conclut cette section par l'étude des relations entre les pannes de la machine et les erreurs de l'homme.

3.1. Mécanismes de correction d'erreur

Selon Dix, Finlay, Abowd, et Beale, il convient de distinguer deux mécanismes de correction d'erreur : la correction arrière ("backward error recovery"), et la correction avant ("forward error recovery") [Dix, Finlay, et al. 1993] :

3.1.a. La correction d'erreur arrière

La correction d'erreur arrière fait revenir le système dans l'état précédant l'erreur. Il y a réellement correction au sens où les effets de l'erreur sont annulés. La grande majorité des interfaces homme-machine actuelles disposent de telles fonctions de correction arrière. Selon Yang, elles peuvent se présenter sous trois formes [Yang 1992] lue dans [Lenman & Robert 1994a] :

• *La fonction défaire* ("undo") est la plus fameuse. Ses effets de bord font revenir l'état présent du système d'une ou de plusieurs étapes en arrière. Elle est très utilisée et appréciée des utilisateurs. Cette fonction pose cependant de nombreux problèmes lors de la phase de programmation du noyau fonctionnel, comme la gestion de l'historique des commandes par exemple. De plus, la réalisation de cette fonction impose des choix de conception parfois délicats : présentation de la fonction à l'utilisateur, portée, granularité des commandes, ou encore aspects d'une commande couverts par la fonction. Selon Lenman et Robert, les utilisateurs n'ont pas a priori le même point de vue que les concepteurs sur ces choix [Lenman & Robert 1994b]. Problème auquel s'ajoute, dans le cas des collecticiels, celui

de choisir si un utilisateur peut ou non défaire ce qu'a fait un autre utilisateur.

• *La fonction arrêter* ("stop") est plus aisée à définir. Elle se contente d'interrompre la commande en cours. Ce type de fonction permet à un utilisateur d'interrompre un traitement qui n'aboutit pas, ou de récupérer une erreur de doigt. Notons que cette fonction ne correspond pas toujours à la définition d'une correction arrière. Citons comme exemple la commande de copie de fichier sur un Macintosh™ ou sous UNIX. Celle-ci peut être interrompue respectivement par les commandes "stop" ou "control-C". Lorsque la copie d'un seul fichier est interrompue, le système reprend son état initial. Par contre, lorsque une copie multiple est en cours, le système peut aboutir à un état intermédiaire où un certains fichiers ont déjà pu être copiés avant l'interruption.

• *La fonction annuler* ("cancel") est plus subtile à définir. Cette fonction permet à l'utilisateur d'abandonner une commande en cours de spécification. Par exemple, un utilisateur peut vouloir abandonner la spécification d'une impression s'il s'aperçoit qu'il ne dispose pas de l'imprimante adéquate. Lenman et Robert n'y font pas allusion, mais on retrouve ici un problème de granularité comparable à celui de la fonction défaire. Il est en effet nécessaire de déterminer la portée de la fonction annuler. L'utilisateur doit pouvoir connaître, au moment où il annule une spécification en cours, l'étendue des spécifications qui vont être perdues par le système.

La tâche de correction d'erreur arrière replace l'utilisateur dans une situation connue, une étape avant d'avoir commis l'erreur. En fait, elle peut être vue comme un moyen de remonter le temps, tout en gardant la connaissance de ce qu'il ne faut pas faire. De ce fait, ces fonctions de correction arrière, en particulier la fonction défaire, encouragent les utilisateurs à explorer les possibilités de l'interface sans risque de se retrouver dans une situation catastrophique. C'est pourquoi malgré les nombreux problèmes de programmation qu'elles engendrent, les fonctions de correction arrière doivent être mises à disposition des utilisateurs lorsque cela est possible.

3.1.b. La correction d'erreur avant

Contrairement à la correction arrière, la correction avant se contente de minimiser les effets de bord dus à l'erreur. Dans une tâche de dessin par exemple, le fait de gommer une esquisse faite au crayon est une correction arrière, par contre, le fait de barrer un mot erroné pour le réécrire plus loin est une correction avant. En effet, l'utilisateur ne se retrouve pas dans l'état précédent l'erreur, car il reste des traces de la correction : le mot barré. L'utilisation de fonctions de correction avant est, du point de vue de l'interface homme-machine, beaucoup moins intéressante que la correction arrière. L'utilisateur ne peut plus utiliser une fonction générique, comme défaire qui annulera les effets de bord non désirés, mais il doit planifier de nouvelles tâches de façon à minimiser les effets de son erreur. Bon nombre de commandes ne peuvent disposer de fonctions de correction arrière, et ce pour deux raisons principales [Lenman et al. 1994a] :

• Lorsque la fonction utilisée possède des effets de bord irréversibles, à l'évidence, aucune fonction de correction arrière n'est applicable. Par exemple lorsque l'on allume un feux d'artifice, il est impossible de revenir en arrière. Ce type de fonction est courant dans de nombreux logiciels pourtant simples : le déplacement des fichiers sur le bureau d'un Macintosh™ ne peut être annulé. Les possibilités restreintes des noyaux fonctionnels sous-jacents sont le plus souvent responsables de ces lacunes.

• Lorsqu'une fenêtre temporelle engendre un effet irréversible. C'est le cas de nombreuses fonctions présentes dans les systèmes dynamiques. La fonction a été corrigible pendant un certain laps de temps, mais ce temps est écoulé et les effets de bords sont irréversibles. Vous pouvez par exemple toujours récupérer un document jeté par erreur dans la corbeille, tant qu'elle n'a pas été vidée.

Selon Lenman et Robert la correction avant est peu utilisée et peu étudiée [Lenman et al. 1994a]. Pourtant, elle est à rapprocher des procédures d'urgence utilisée dans les systèmes critiques comme les aéronefs ou les centrales nucléaires. En effet, la correction arrière n'est en général pas réalisable en cas de panne ou de fausse manœuvre sur ces systèmes très dynamiques. Le pilote ou l'opérateur doivent alors intervenir et effectuer des tâches de correction avant.

3.2. Insuffisance des définitions actuelles

La distinction entre les mécanismes de correction arrière et de correction avant n'est pas aussi aisée que ne le laisse à penser les définitions mentionnées au paragraphe 3.1. En effet, la granularité de la tâche, l'étendue du monde, et le rôle du temps jouent un rôle primordial dans cette distinction. De plus, celles-ci ne prennent pas en compte le coût réel de l'erreur pour l'utilisateur.

3.2.a. Granularité de la tâche considérée

Dix, Finlay, Abowd, et Beale citent l'exemple d'un château de cartes afin de différencier la correction arrière de la correction avant [Dix et al. 1993]. Les auteurs affirment que faire tomber l'ensemble d'un château de cartes, en ajoutant maladroitement une carte sur l'édifice, est l'exemple type d'une erreur qui ne peut pas être corrigée par correction arrière. En fait, il est nécessaire de préciser dans ce cas quelle est exactement la tâche entachée d'erreur. Si la tâche en question est la mise en place d'une seule carte, alors l'erreur commise ne peut être corrigée par correction arrière. Par contre, si la tâche considérée est la construction d'un château de cartes dans sa totalité, cette erreur peut être corrigée par correction arrière en ramassant toutes les cartes tombées. Le joueur se retrouve alors dans l'état précédant la construction du château de cartes, et peut la recommencer.

Utilisons un autre exemple afin de préciser la problématique. Écrivons une lettre ; si par mégarde nous écrivons un mot pour un autre, nous nous trouvons devant plusieurs alternatives :

- si l'encre utilisée est effaçable, ou si nous écrivons au crayon, il suffit de faire disparaître le mot erroné puis de réécrire par-dessus, c'est une correction arrière. Nous nous retrouvons en effet dans l'état précédant l'erreur.
- si l'encre est indélébile, deux possibilités s'ouvrent à nous. La première consiste à barrer le mot erroné puis, à écrire le mot correct à la suite. C'est une correction avant car il reste des traces, le mot barré, ce qui est relativement disgracieux…

Nous pouvons alors être amené à choisir une seconde solution, plus radicale, consistant à déchirer la lettre et à recommencer. Cette solution peut être vue comme une correction avant, car nous ne nous retrouvons pas

dans la situation précédant l'écriture du mot erroné. Nous avons dû planifier de nouvelles tâches, et nous nous retrouvons avec une feuille blanche.

Mais c'est également un cas de correction arrière, car nous nous retrouvons dans l'état où nous étions avant de commencer la lettre. Cependant, la tâche considérée ici est l'écriture d'une lettre dans sa totalité, plutôt que l'écriture d'un seul mot comme auparavant. Nous avons néanmoins perdu une feuille de papier, un peu d'encre, de temps, et de patience. Nous évoquons ce problème au paragraphe 3.2.b.

La distinction entre correction arrière ou correction avant dépend donc de la granularité de la tâche considérée. À l'extrême, toute erreur peut être corrigée par correction arrière, à condition que la granularité de la tâche considérée soit suffisamment importante.

3.2.b. Étendue du monde considéré

Reprenons l'exemple de la tâche d'écriture d'une lettre. Si le rédacteur est quelque peu maladroit, et s'il décide de recommencer sa rédaction au début à chaque erreur, il peut arriver à épuiser son stock de papier. Au moment où il utilise sa dernière feuille de papier, il n'a plus le droit à l'erreur. La possibilité d'utiliser une correction arrière est donc dépendante du monde considéré. Si le stock de papier ne fait pas partie du monde de la tâche, et donc supposé infini, alors l'erreur pourra toujours être corrigée par correction arrière. Dans le cas contraire, la correction arrière ne pourra être disponible que jusqu'à épuisement du stock de papier. Plus généralement, on se réfère ici à la notion d'épuisement des ressources.

De même, si la notion de temps doit être prise en compte, comme dans les systèmes dynamiques, les possibilités de correction peuvent évoluer au cours du temps. Supposons que la lettre écrite précédemment soit destinée à commander un cadeau. Or par mégarde, vous avez indiqué comme adresse d'expédition non pas celle du destinataire du cadeau, mais la votre. La tâche considérée est l'écriture de l'adresse. Si vous vous apercevez de cette erreur avant de mettre la commande sous enveloppe, il vous suffit d'effectuer une correction arrière : effacer votre adresse et inscrire la nouvelle. Si vous vous souvenez de votre erreur une fois la lettre sous enveloppe, il vous faudra effectuer une correction avant : déchirer l'enveloppe. Si vous avez déjà posté la lettre, et bien c'est trop tard, le cadeau n'arrivera pas à temps à l'adresse

voulue. Comme le souligne Woods, le temps peut dégrader l'état des systèmes suite à une erreur et rendre les corrections de plus en plus difficiles [Woods 1990] lue dans [Lenman et al. 1994a]. Les systèmes dynamiques sont particulièrement sensibles à ce type de dégradation. Dans ces systèmes, le temps fait alors partie de l'état du monde et joue un rôle primordial.

3.2.c. Coût réel de l'erreur et de sa correction

Reprenons l'exemple de la lettre du paragraphe 3.2.a. La tâche de correction arrière consistant à recommencer la lettre au début est, du point de vue du rédacteur, beaucoup plus coûteuse en temps que la simple rature d'un mot. En effet, la tâche du rédacteur ne se limite pas à corriger une erreur, il doit aussi atteindre son but, c'est-à-dire écrire la lettre. Défaire ce qui a été fait n'est qu'une partie de la correction d'erreur, l'utilisateur doit aussi refaire ce qu'il a fait faux. Les définitions des mécanismes de correction d'erreur ne prennent pas en compte cet aspect, et peuvent ainsi laisser supposer qu'une correction arrière est toujours préférable à une correction avant.

Or, l'utilisateur peut se trouver devant une situation où il doit choisir entre réaliser une tâche coûteuse pour corriger complètement son erreur par correction arrière, ou bien tolérer un état non optimal et réaliser une tâche moins coûteuse par correction avant. C'est pourquoi il est nécessaire d'aider le concepteur à évaluer les mécanismes de correction d'erreur en fonction de leur coût estimé pour l'utilisateur. Nous retrouvons ici l'importance de la notion de coût pour l'utilisateur déjà introduite au paragraphe 2.1.a du chapitre I. Cette notion est reprise au paragraphe 3.3 ci-après. Ce paragraphe a en outre pour objet principal la description de dimensions d'analyse devant aider le concepteur au choix des mécanismes de correction d'erreur.

3.3. Aide au choix des mécanismes de correction d'erreur 🕮

En réponse à l'insuffisance de la définition des mécanismes de correction d'erreur pour aider aux choix de conception, nous proposons une représentation graphique de la notion de correction d'erreur basée sur trois dimensions d'analyse.

3.3.a. Représentation graphique

Cette représentation permet au concepteur de situer dans un graphe les mécanismes de correction d'erreur afin de les comparer. Ce graphe comporte

au minimum deux axes, indiqués sur la figure II-4, représentant les deux principales dimensions d'analyse :

• Sur l'axe des ordonnées, *le coût de l'erreur pour l'utilisateur* représente la totalité des coûts pour l'utilisateur dus à l'occurrence d'une erreur, comme évoqué au paragraphe 3.2.c. Pour une tâche donnée, cela représente la somme du coût de la tâche erronée, de la correction, et de la poursuite de la tâche jusqu'à son but ; auquel on retranche le coût de la tâche correcte sans erreur. De ce fait, le choix de la granularité de la tâche, évoqué au paragraphe 3.2.a, ne se pose plus dans les mêmes termes. En effet, cette notion de coût s'affranchit de la granularité de la tâche considérée pour ne retenir que le surplus de coût dû à l'erreur. Elle normalise en quelque sorte la notion de coût de l'erreur.

Cette notion de coût est représentée sur un seul axe afin de faciliter la lecture du graphe. C'est au concepteur de choisir la grandeur la mieux représentative de la notion de coût. Ce peut être le coût cognitif, conatif, ou bien plus simplement le temps perdu lors de la correction. Cependant, il peut s'avérer utile d'utiliser plusieurs axes, au détriment de la lisibilité, mais afin de permettre une meilleure distinction entre les différents constituants du coût. Ils ont été évoqués précédemment au paragraphe 2.1.a du chapitre I.

Lorsqu'un seul axe est utilisé, le coût C peut se calculer par exemple comme la somme pondérée et normalisée des n différents constituants C_i du coût. Les coefficients p_i de cette pondération sont à la discrétion de l'évaluateur et dépendent fortement du contexte de l'utilisation du système.

Le coût C peut donc être calculé selon la formule : $$C = \frac{\sum_{i=1}^{n} p_i . C_i}{\sum_{i=1}^{n} p_i}$$

• Sur l'axe des abscisses, *la différence entre l'état souhaité et l'état corrigé* représente l'ensemble des états du monde qui diffèrent entre l'état du système souhaité par l'utilisateur, et l'état obtenu après sa correction complète. De la même manière que le coût, l'évaluation de cette différence ne peut se faire que de manière subjective car deux états du monde ne sont pas toujours comparables.

Comme précédemment, cette différence est représentée sur un seul axe par souci de lisibilité. Mais l'utilisation de plusieurs axes est possible si plusieurs grandeurs états du monde doivent être distinguées. Cet axe permet au concepteur de prendre en compte les problèmes de l'état du monde évoqués au paragraphe 3.2.b.

Comme le montre la figure II-4, les notions précédemment définies de correction arrière et de correction avant peuvent être représentées sur le graphe. Il est également possible d'y représenter l'état du système s'il n'y a pas de correction, ou lorsque aucune erreur ne s'est produite :

• La correction arrière fait revenir l'état du système tel qu'il était avant l'occurrence de l'erreur. Elle permet ainsi à l'utilisateur d'obtenir ensuite l'état souhaité sans séquelles. La différence entre l'état souhaité et l'état corrigé est donc nulle.

• La correction avant est en général plus coûteuse qu'une correction arrière. De plus, elle ne permet pas dans le cas général de faire revenir le système dans l'état précédent l'erreur. Il reste donc des traces de l'erreur qui éloignent l'état du système de l'état souhaité par l'utilisateur.

• Lorsqu'il n'y a pas de correction, le coût de l'erreur pour l'utilisateur est par définition nul. Par contre, l'état du système est très éloigné de ce que souhaite l'utilisateur.

• Le point "0" peut être vu comme l'état du système lorsque aucune erreur ne s'est produite. Le coût pour l'utilisateur, ainsi que la différence entre l'état souhaité et l'état corrigé sont nuls.

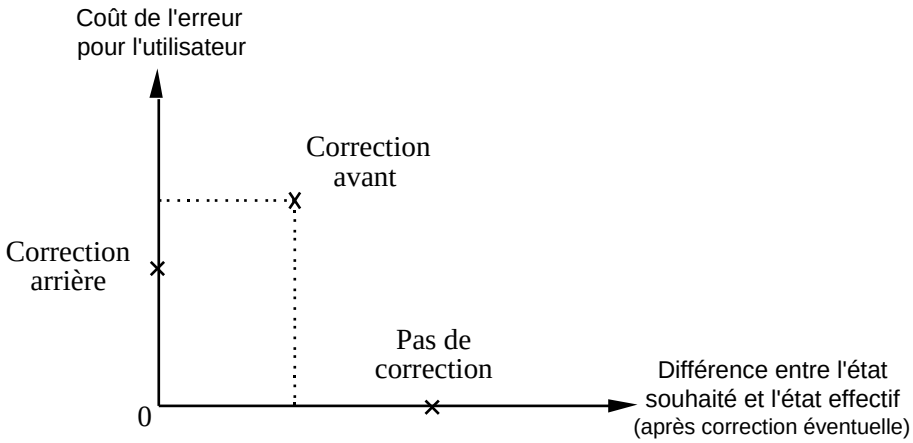

Figure II-4 : Graphe représentant les différents types de correction d'erreur selon deux axes.

3.3.b. Intégration de la notion de temps

La notion de temps évoquée au paragraphe 3.2.b doit impérativement être prise en compte dans le cas des systèmes dynamiques. Nous avons donc ajouté un axe au graphe précédent représentant cette nouvelle dimension d'analyse. Elle représente l'écoulement du temps depuis l'occurrence de l'erreur. Comme le montre la figure II-5, la représentation graphique de cette dimension d'analyse permet de mettre en évidence trois zones :

‹ Une première zone où une *correction arrière* est disponible. Par exemple, un avion en approche finale a oublié de sortir son train d'atterrissage. Le pilote s'est aperçu du problème et sort le train, c'est le cas de *correction arrière* car il se retrouve immédiatement dans l'état souhaité. Il peut aussi choisir de remettre les gaz et recommencer la procédure d'atterrissage. C'est un cas de *correction avant* car un peu de temps et de carburant ont été perdus dans cette tâche supplémentaire.

› Une deuxième zone où seule la *correction avant* est disponible. L'état du système sans correction se dégrade et l'état récupérable est plus éloigné de l'état souhaité que précédemment. Reprenons notre exemple : le pilote de l'avion ne s'est aperçu de rien, mais la tour de contrôle lui signale son erreur et lui demande une remise de gaz. L'avion étant plus bas que précédemment, la manœuvre de remise des gaz est plus longue et plus coûteuse en carburant.

fi Une troisième zone où *aucune correction* de l'erreur initiale n'est plus disponible. L'état du système se dégrade à la fois très rapidement et très fortement. Dans notre exemple ni le pilote, ni la tour de contrôle ne se sont aperçus du problème, l'avion se pose sur le ventre.

Cette représentation des possibilités de correction d'erreur selon trois dimensions d'analyse doit permettre au concepteur de l'aider dans ses choix. Cependant, le principe de ces dimensions n'est pas limité aux mécanismes de correction d'erreur. Il peut être étendu aux trajectoires d'interaction en général. En substituant par exemple au coût de l'erreur le coût d'une tâche, et à l'état corrigé l'état obtenu, on obtient la trajectoire d'interaction d'un utilisateur au cours du temps.

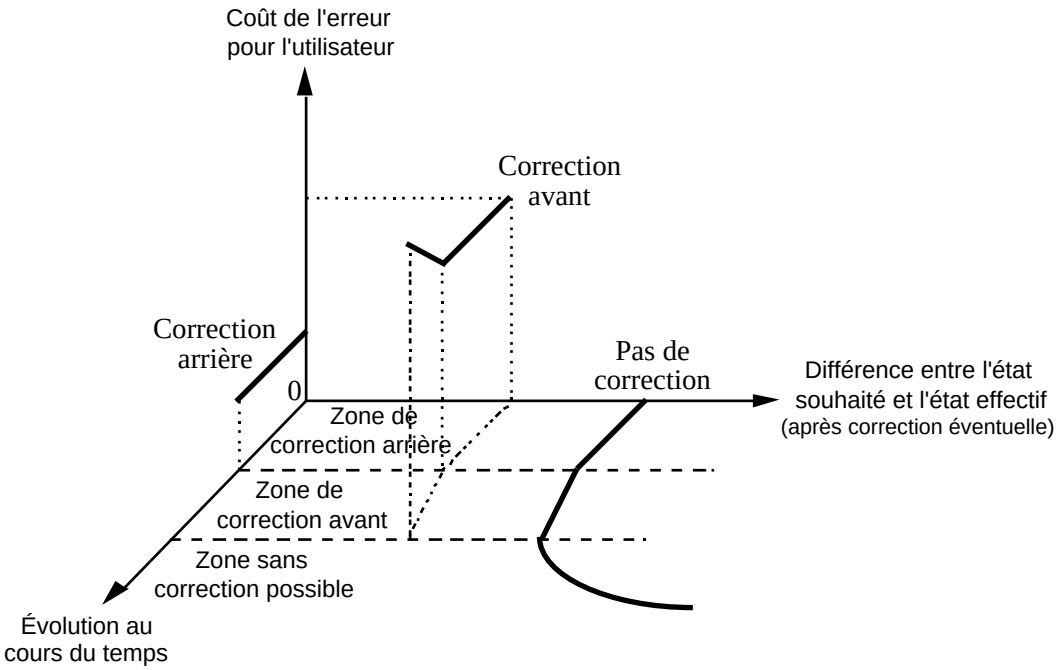

Figure II-5 : Graphe représentant les différents types de correction d'erreur avec leur évolution au cours du temps.

3.4. Les erreurs de l'homme et les pannes de la machine

Les mécanismes de correction des erreurs s'appliquent aux erreurs de l'homme, mais peuvent également être destinés à corriger les pannes de la machine. En effet, la section 2 du chapitre I traitant de l'automatisation nous a montré que les pannes de la machine répondent aux erreurs de l'homme. C'est pourquoi, les mécanismes de traitement d'erreur détaillés au paragraphe 3.3 ne doivent pas être considérés comme s'appliquant uniquement à l'homme. Ils peuvent également s'appliquer aux architectures de machines ou bien au couple homme-machine. Ce point de vue peut aussi être étendu au couple homme-homme dans le cas des collecticiels :

• *La machine, composant d'un système, peut tomber en panne.* Les machines utilisent entre elles des procédures particulièrement efficaces pour traiter les pannes dont elles peuvent être victimes. L'exemple du domaine le plus connu est celui de la sonde spatiale Voyager. Lancée en 1977, elle répond encore aux sollicitations du contrôle au moment où ces lignes sont écrites, et ceci grâce à une architecture STAR ("Self-Test And Repair") tolérant les fautes [Jones 1979]. Une architecture de machines peut donc corriger ses erreurs, comme le font les systèmes de transmission de données grâce aux codes de redondance cyclique. Elle peut aussi se

reconfigurer, comme le permet l'architecture des ordinateurs des commandes de vol électriques de l'A320 [Airbus Industrie 1992].

• *L'homme corrige l'homme*. Cette fonction est mise à profit dans les collecticiels, et en particulier dans les systèmes pilotés en équipage comme les avions de ligne [Racca 1992]. Cette notion justifierait à elle seule une partie complète. Mais elle ne sera pas traitée dans le cadre de ce mémoire.

Deux entités d'un système homogène peuvent donc se corriger. Cependant dans le cadre de ce mémoire, l'aspect le plus intéressant est sans doute les corrections réciproques que peuvent effectuer l'homme et la machine. Comme le souligne l'introduction du chapitre I, les systèmes hétérogènes homme-machine sont porteurs d'un potentiel extraordinaire, car ils peuvent être conçus de façon à ce les erreurs de l'un soient corrigées grâce à l'attention de l'autre :

• *L'homme corrige les pannes de la machine*. Si le train d'atterrissage d'un avion ne sort pas suite à un défaut du système hydraulique, l'équipage doit utiliser une procédure de secours pour récupérer la panne de la machine. Cette procédure consiste le plus souvent à utiliser la gravité pour laisser le train sortir de lui même. Cette procédure est longue et particulièrement peu efficace. Elle peut obliger l'équipage à différer son atterrissage le temps de l'exécuter. Bien qu'ils soient rares, les incidents techniques existent et les opérateurs n'y sont pas toujours correctement préparés. C'est aux concepteurs et aux équipes de formation d'entraîner les opérateurs à de telles situations et de définir les procédures correctives.

• *La machine peut aussi corriger l'homme*, comme le font les correcteurs orthographiques intégrés à la plupart des logiciels de traitement de textes actuels. Sur l'A320 une sécurité empêche l'appareil d'atteindre l'incidence de décrochage et le maintient ainsi pilotable [Airbus Industrie 1992]. Mais les corrections de la machine ne sont pas toujours fort à propos. Les correcteurs orthographiques n'ont pas toujours un vocabulaire complet, et considèrent comme faux bon nombre de termes techniques. C'est pourquoi les corrections de la machine doivent être soit parfaitement sûres[10] et automatiques, soit être négociées avec l'homme.

[10] Le terme "parfaitement sûr" correspond en aéronautique à une probabilité de 10^{-9} pannes par heure de vol.

La correction des erreurs entre deux entités hétérogènes, négociée ou non, nécessite une interruption de l'une des entités par l'autre. Très liées aux erreurs, les interruptions font l'objet de la section 4 ci-après.

4. Erreurs et interruptions

Commettre une erreur suite à une interruption inopinée est une chose courante. Reprenons un exemple déjà cité au chapitre I : vous étiez parti chercher le pain, vous avez rencontré un ami, après avoir bavardé vous êtes rentré chez vous, sans le pain… De même, corriger une erreur nécessite habituellement une interruption de la tâche en cours : vous étiez en train de taper un numéro de téléphone lorsque votre doigt a glissé sur une mauvaise touche, vous avez dû raccrocher et recommencer l'appel. Les interruptions et les erreurs sont deux notions très liées, l'une provoquant souvent l'autre. En outre, les interruptions sont omniprésentes aussi bien dans la vie quotidienne que lors de la conduite de systèmes critiques. Pourtant, contrairement aux erreurs, les interruptions n'ont été à ce jour que peu étudiées du point de vue de l'interaction homme-machine. C'est donc un domaine encore à défricher, où l'on peut s'inspirer en partie des résultats obtenus sur les interruptions au niveau des systèmes d'exploitation, mais aussi des résultats des études ethnographiques. Cette section débute par un bref état de l'art des études sur les interruptions, au paragraphe 4.1. Nous proposons ensuite une modélisation de la notion d'interruption au paragraphe 4.2. Enfin, cette section se conclut par une synthèse des liaisons entre interruptions, erreurs, et corrections d'erreurs.

4.1. Études de l'interruption

Les rares études concernant les interruptions, du point de vue de l'interaction homme-machine, ont été réalisées principalement à but ethnographique dans le cadre du travail de bureau (§ 4.1.a). Les plus nombreuses études évoquant les interruptions sont en fait issues des recherches en systèmes d'exploitation des ordinateurs (§ 4.1.b). Nous proposons à la fin de cette partie une synthèse entre ces deux domaines (§ 4.1.c).

4.1.a. Études ethnographiques

O'Conaill et Frohlich ont cherché à quantifier l'importance des interruptions dans l'activité du travail de bureau [O'Conaill & Frohlich 1995].

Ils ont montré que l'occurrence des interruptions n'est pas un phénomène rare : chaque sujet est interrompu en moyenne quatre fois par heure, et la durée moyenne d'une interruption est d'un peu plus de deux minutes. Il en résulte que le temps consacré aux interruptions est d'environ une dizaine de minutes par heure, ce qui n'a rien de négligeable. Autre résultat : O'Conaill et Frohlich ont montré que suite à une interruption, seulement 55% des sujets retournent à la tâche qui avait été interrompue. Pourtant, être interrompu n'est pas toujours négatif : les auteurs précisent que dans la majorité des cas le sujet qui subit l'interruption en retire un bénéfice, parfois même aux dépends du sujet interrompant. Par exemple, si un étudiant en thèse interrompt son responsable pour lui poser une question, il se peut qu'il reparte avec un article à écrire…

Rouncefield et al. ont effectué une étude ethnographique rapide du travail de bureau dans un centre de conférence [Rouncefield, Hughes, et al. 1994]. Leur étude, même si elle s'intéresse aux interruptions, est essentiellement destinée à l'étude du travail coopératif. Les auteurs soulignent l'existence de "l'état d'interruption permanent" ("constant interruption") qui caractérise ce travail. Ils font également remarquer que les interruptions prévisibles sont les moins disruptives.

4.1.b. Étude des systèmes d'exploitation des ordinateurs

Les études ethnographiques ne nous apportent que peu d'éléments sur la nature exacte de l'interruption. Nous nous sommes donc tourné vers les définitions issues des systèmes d'exploitation [Krakowiak 1985]. Nous constatons en outre que ces définitions peuvent assez facilement s'appliquer au cas de l'interaction homme-machine. Tout d'abord, les systèmes d'exploitation définissent les interruptions comme des ‹‹ événements (excepté les instructions de branchements) qui modifient le déroulement normal de l'exécution d'un programme ›› Cette définition peut convenir aux activités humaines si l'on substitue le mot tâche au mot programme. Cependant, pour les systèmes d'exploitation, comme pour l'interaction homme-machine, il est nécessaire de faire une distinction supplémentaire entre les interruptions internes et externes :

• *Les interruption internes*, également appelés auto-interruptions ou déroutements sont causées par l'exécution d'une instruction. Cette interruption peut être volontaire comme l'appel au superviseur, ou

consécutive à une erreur comme la division par zéro. Du point de vue de l'interaction homme-machine, les déroutements peuvent prendre la forme d'une remise en mémoire d'un fait important, typiquement le rendez-vous urgent oublié. Ce peut être également la détection immédiate par un utilisateur d'une action erronée, par exemple une erreur de doigt.

• *Les interruptions externes* sont causées par un module externe au processeur. L'arrivée d'une information sur un port d'interface, ou le déclenchement d'un chien de garde sont des interruptions externes. Du point de vue de l'interaction homme-machine, de telles interruptions peuvent être dues à une demande en provenance du système, ou à un coup de téléphone.

Lorsqu'une interruption apparaît au cours du traitement d'une autre interruption, deux comportements peuvent être distingués selon Walker et Cragon [Walker & Cragon 1995] : ‹ La seconde interruption est prioritaire et interrompt la première. On parle alors d'interruptions emboîtées ("nested interrupt"). › La seconde interruption est mise en file d'attente ("queued interrupt"). Ces deux comportements se retrouvent dans le cas de l'interaction homme-machine. Lors de l'occurrence d'une interruption l'utilisateur peut l'ignorer, ou bien la juger prioritaire et suspendre sa tâche en cours, même si cette tâche est déjà une interruption. Les actions des standardistes, répondant à plusieurs coups de téléphone simultanés, sont un exemple de ces comportements.

4.1.c. Synthèse

Nous avons successivement évoqué le point de vue ethnographique et le point de vue système sur la nature de l'interruption. L'étude des systèmes d'exploitation nous apporte de nombreuses définitions sur les mécanismes d'interruption qui peuvent être réutilisés pour l'ingénierie de l'interaction homme-machine. Cependant cette réutilisation doit être tempérée par les résultats des études ethnographiques. Notamment, et contrairement au cas général des systèmes d'exploitation, une tâche interrompue n'est pas obligatoirement reprise par l'utilisateur à la fin de l'interruption. En effet, l'étude ethnographique de O'Conaill et Frohlich évoquée au paragraphe 4.1.a montre que seulement une partie des sujets retournent à la tâche qui a été interrompue. Les résultats des systèmes d'exploitation ne sont pas donc réutilisables directement. En outre, définir la nature de l'interruption ne nous semble pas suffisant pour aider le concepteur à les prendre en compte au sein

du processus de conception. C'est pourquoi nous avons cherché à les modéliser. Le paragraphe 4.2 ci-après a pour objet cette modélisation.

4.2. Modélisation des interruptions 🕮

Afin de guider le concepteur lors des étapes de spécification d'une interface homme-machine, nous proposons une modélisation des interruptions. Notre démarche de recherche est purement exploratoire : elle se base sur une compilation non exhaustive d'exemples et d'expériences du quotidien. Sa finalité première est l'ingénierie de l'interaction homme-machine, cependant il est probable que ses résultats peuvent également s'appliquer à un domaine plus large. Conscient des limites du modèle proposé, nous le présentons comme un modèle indicatif et non impératif. Celui-ci permet néanmoins au concepteur de structurer son approche des interruptions et d'en prévoir les principales étapes. A notre connaissance aucun travail n'a été réalisé sur ce thème jusqu'à présent.

4.2.a. Décomposition de l'interruption en trois phases

Une interruption peut se décomposer en trois phases séquentielles, comme le montre la figure II-6 : le prologue, le corps de l'interruption, et enfin l'épilogue. La réunion de ces trois phases est appelée la tâche d'interruption.

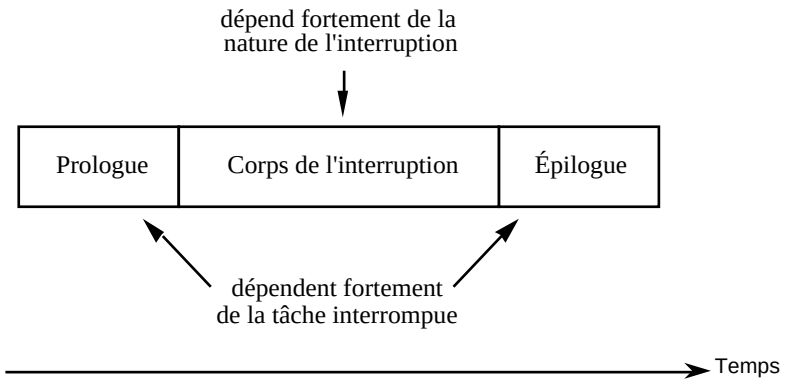

Figure II-6 : Décomposition d'une tâche d'interruption en trois phases séquentielles [11].

• Le *corps de l'interruption* est défini comme la tâche réellement "utile" de l'interruption, c'est elle qui contient les effets de bord ayant motivé l'interruption. Cette tâche est le plus souvent indépendante du contexte, c'est-à-dire de la tâche interrompue. Par exemple, la tâche de répondre au téléphone ne dépend que rarement de la tâche de l'utilisateur avant l'appel.

[11] Structure identique à celle proposée par Pendibidou pour le projet THÉRÈSE [Pendibidou 1982].

• Au contraire, le *prologue* et *l'épilogue* sont des tâches articulatoires fortement dépendantes du contexte de l'interruption. Par exemple, lorsqu'un cuisinier doit répondre au téléphone, il prendra la précaution de couper le feu sous le lait en train de bouillir. Cependant, prologue et épilogue dépendent également, mais dans une moindre mesure, du corps de l'interruption lui-même. Dans notre exemple, le cuisinier peut se contenter de réduire le feu si le téléphone est un modèle sans fil, car il pourra surveiller son lait en parallèle avec sa conversation téléphonique.

Cette décomposition permet au concepteur de concevoir séparément les deux catégories de tâches entrant en compte dans l'interruption. Le corps de l'interruption peut être spécifié comme une tâche classique, par exemple : "répondre au téléphone". Elle sera éventuellement instanciée en fonction du contexte, par exemple : "en environnement bruyant". Le prologue et l'épilogue doivent être spécifiés en même temps que la tâche interruptible, par exemple : "ne pas faire déborder le lait". Ils seront ensuite instanciés en fonction du contexte, par exemple : "réduire le feu si on peut surveiller, couper sinon". Le concepteur dispose alors d'une bibliothèque de prologues & épilogues, et de corps d'interruption pouvant se combiner afin de donner des tâches d'interruption complètes. Il convient cependant de modérer la règle de dépendance du prologue et de l'épilogue de la tâche interrompue. Certaines interruptions particulières, comme les procédures d'extrême urgence imposant à la tâche courante une fin brutale, ne tolèrent pas de longs prologues. Dans ce cas le prologue devient principalement dépendant de la tâche d'interruption.

4.2.b. Interruptions externes

Au paragraphe 4.1.b nous avons montré que l'interaction homme-machine, de la même manière que les systèmes d'exploitation, fait la distinction entre interruptions internes et externes. Or, lorsqu'une interruption est externe, elle est déclenchée explicitement par l'environnement de l'utilisateur. Il est alors nécessaire de spécifier non seulement la tâche d'interruption de l'agent interrompu, mais aussi la tâche de déclenchement effectué par l'agent interrompant. Cette tâche de déclenchement de l'interruption, illustrée figure II-7, peut se décomposer en trois phases successives :

‹ Elle débute par une tâche de prise de *décision* de déclencher l'interruption. Cette tâche est souvent modélisée par des préconditions. Dans notre modèle, elle est considérée comme une tâche à part entière.

› L'agent interrompant effectue ensuite une tâche de *signalisation* en direction de l'autre agent, par l'intermédiaire d'un ou plusieurs canaux de communication. Cette signalisation peut avoir une durée significative qui peut être celle de la durée de la tâche d'interruption, et même au-delà.

fi Une tâche optionnelle de *surveillance* permet à l'agent interrompant de vérifier l'acquittement et éventuellement la bonne exécution de l'interruption. Cette tâche peut être effectuée en parallèle de la tâche de signalisation.

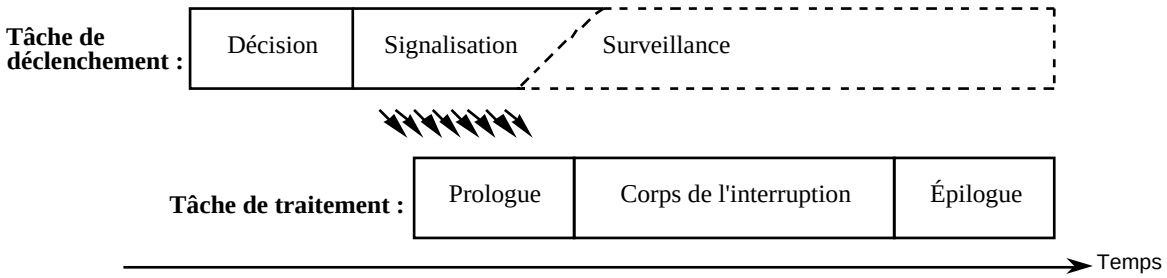

Figure II-7: Tâches de déclenchement et de traitement d'une interruption externe.

Afin d'illustrer notre propos, considérons l'exemple d'un étudiant en train de fixer une affiche sur le mur de sa chambre. C'est alors que son téléphone sonne. Modélisons cet exemple avec deux agents : l'agent interrompu est l'étudiant, tandis que l'agent interrompant réunit l'interlocuteur et le téléphone. Selon notre modèle, la décision de déclencher la sonnerie du téléphone est prise par l'interlocuteur lorsqu'il décide de téléphoner. La signalisation est bien évidemment la sonnerie du téléphone. La surveillance modélise le fait qu'au bout d'un certain nombre de sonneries, l'interlocuteur décidera qu'il n'y a personne à l'autre bout, et raccrochera. Le cas de l'agent interrompu est plus intéressant. Le prologue comprend la prise en compte de la sonnerie, suivie du choix effectué par l'agent pour savoir s'il va répondre ou non. Si l'étudiant décide de répondre, une tâche de sauvegarde de contexte, ici finir rapidement de fixer une punaise pour éviter que l'affiche ne se déchire, doit être exécutée. Enfin l'étudiant peut entreprendre de répondre au téléphone, ce qui constitue le corps de l'interruption. L'épilogue débute dès que l'étudiant a raccroché le combiné du téléphone. Il restaure alors partiellement son contexte (où en étais-je ?), puis choisit ou non de reprendre la tâche de fixer l'affiche. Si oui, il doit finir de restaurer le contexte en retirant la punaise mise rapidement lors du prologue. Il peut enfin finir de fixer l'affiche au mur de sa chambre.

4.2.c. Affinement du prologue et de l'épilogue

L'exemple évoqué au paragraphe 4.2.b suggère que le prologue et l'épilogue d'une tâche d'interruption peuvent se décomposer en sous-tâches aux rôles bien définis. Nous proposons de les modéliser selon deux structures symétriques.

Le prologue est initié par la prise en compte de la signalisation en provenance de l'agent interrompant. Dans le cas d'une interruption interne, cette tâche est remplacée par la prise de décision de s'auto-interrompre. Une fois la prise en compte de l'interruption effectuée, la tâche de sauvegarde et de changement de contexte peut débuter en parallèle. Cette tâche permet de mettre en suspend la tâche interrompue sans conséquence grave, et de préparer la reprise. Cette tâche se poursuit jusqu'à la fin du prologue. À la suite de la prise en compte de la signalisation, l'agent interrompu doit choisir s'il exécute une interruption, et si oui, quel corps d'interruption exécuter. Vient ensuite le corps de l'interruption. Cet affinement du prologue est illustré par la figure II-8 : les tâches pouvant être effectuées en parallèle sont disposées verticalement.

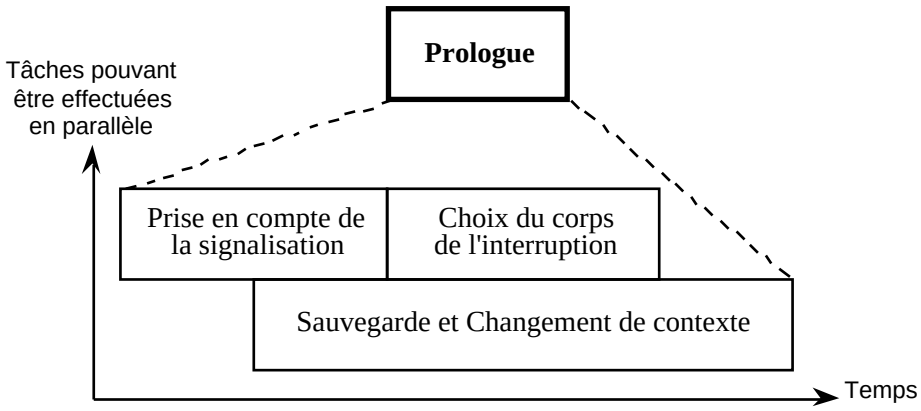

Figure II-8 : Prologue d'une tâche d'interruption.

L'épilogue est construit de manière symétrique. Une restauration partielle du contexte de la tâche interrompue est nécessaire, afin que l'agent interrompu puisse choisir la tâche à effectuer après le corps de l'interruption. Il peut choisir de reprendre la tâche interrompue, ou au contraire de tout abandonner pour faire autre chose. L'agent interrompu peut également abandonner temporairement la tâche interrompue pour la reprendre plus tard. Ce cas peut se modéliser comme une prolongation de l'interruption, ou encore comme une seconde interruption se produisant juste après la première. Ensuite, les actions de sauvegarde du contexte ayant mis en suspend la tâche interrompue peuvent être défaites, et enfin l'agent peut commencer la tâche post-interruption choisie. Cette tâche de reprise est le symétrique de la prise en compte de la signalisation. Elle peut aussi être constituée d'une signalisation en direction de l'autre agent. Cette tâche était absente de l'exemple du paragraphe 4.2.c. Dans le cas d'un système de multifenêtrage par exemple, elle peut être associée au clic dans une nouvelle fenêtre.

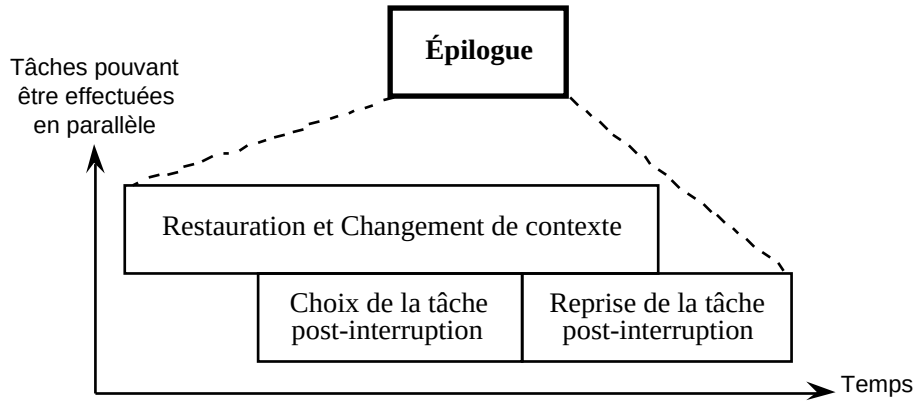

Figure II-9 : Épilogue d'une tâche d'interruption.

4.2.d. Synthèse

La figure II-10 ci-dessous représente la synthèse de notre proposition de modélisation des interruptions.

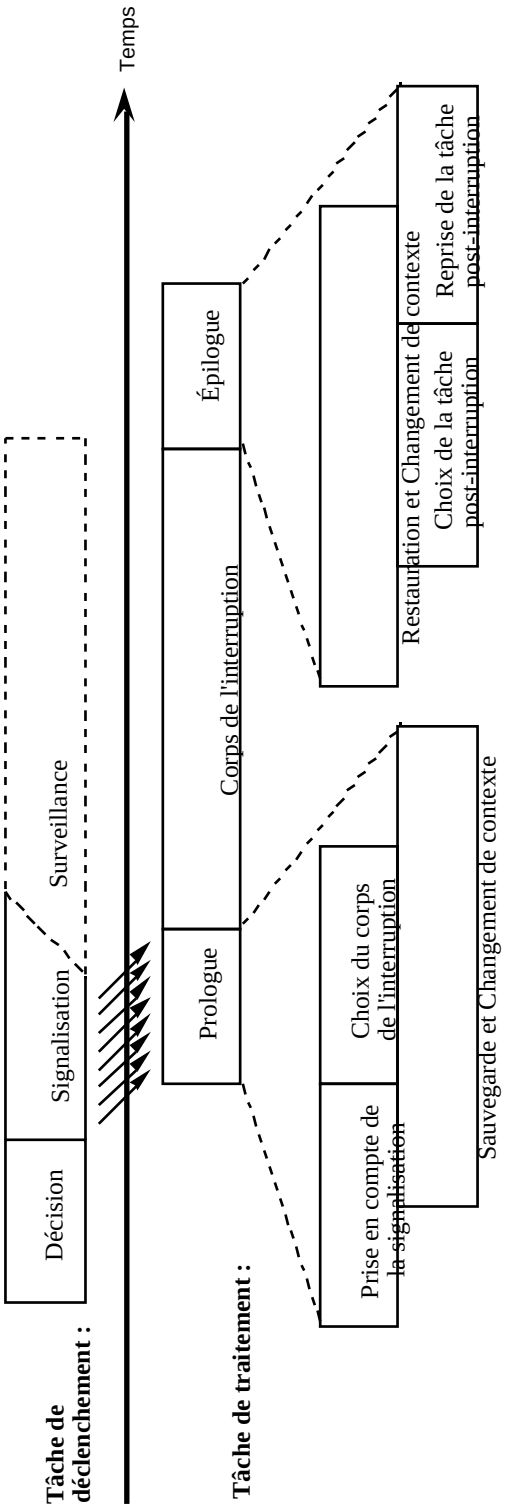

Figure II-10 : schéma synthétique de la modélisation des interruptions.

4.3. Liaisons entre interruption, erreur, et correction d'erreur

Nous venons de définir et modéliser la notion d'interruption. Nous allons maintenant nous intéresser aux relations entre les interruptions, les erreurs, et leurs corrections. Afin d'illustrer notre propos, nous nous intéressons tout d'abord à un exemple de formalisation de la négociation homme-machine, puis nous mettons en lumière les conséquences des interruptions sur l'occurrence des erreurs, et de manière symétrique, les liens entre la correction d'erreur et les interruptions. Enfin, nous définissons la notion fédératrice de singularité.

4.3.a. Négociation homme-machine

Perez-Quiñones et Sibert ont défini et modélisé formellement les interactions entre l'opérateur et le système, dans le cas de demandes d'interruptions et d'annulations de la part de l'utilisateur sur le système [Pérez-Quiñones & Sibert 1996]. Selon ces auteurs, l'interruption n'est qu'une étape préliminaire à une demande d'annulation. Prenons l'exemple d'un processus UNIX : le "Control-Z" est une simple interruption du processus, peut ensuite suivre une annulation par la commande "kill", ou bien une reprise de celui-ci grâce à la commande "fg".

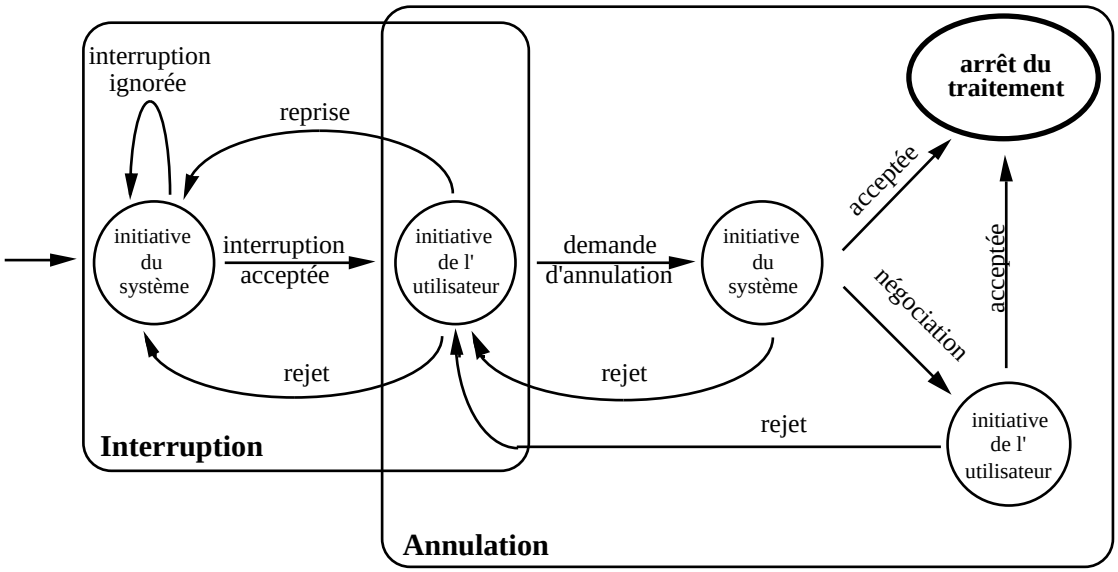

Figure II-11 : Diagramme complet d'une interruption suivie d'une annulation. Adapté et traduit de [Pérez-Quiñones et al. 1996].

Les auteurs ont modélisé le cheminement de l'interaction en prenant successivement en compte le moment où l'utilisateur a l'initiative, et le moment où c'est le système qui a l'initiative. Cette modélisation prend la

forme d'un graphe étape-transition, comme nous le montre la figure II-11. Bien que le point de vue pris par les auteurs soit résolument du côté des outils de spécification, sujet du chapitre III de ce mémoire, le fait même que la notion d'interruption soit liée à la notion d'annulation est intéressante. En effet, pour modéliser l'annulation, c'est-à-dire une correction d'erreur, les auteurs ont dû prendre en compte une interruption préalable du système. Cette relation entre interruption et correction d'erreur est explicitée au paragraphe 4.3.c.

4.3.b. De l'interruption à l'erreur

Qui n'a jamais oublié quelque chose après avoir été interrompu ? Reason cite les "omissions suivant les interruptions" parmi les erreurs du niveau réflexe. L'importance de ce type d'erreur est corroborée par la lecture de nombreux rapports d'accidents aériens, où l'on retrouve parmi les causes la phrase générique : ‹‹ je faisais ceci, puis j'ai été interrompu par cela, puis après avoir traité cela, j'ai oublié de revenir à ceci ››. Reason cite également plusieurs exemples dont un requiert l'attention [Reason 1993] : ‹‹ En faisant le thé, je me suis rendu compte que la boîte [à thé] était vide. J'ai pris un paquet de thé sur l'étagère et j'ai rempli la boîte. Mais j'ai omis de mettre du thé dans la théière et j'ai versé de l'eau bouillante dans une théière[12] vide ››. La procédure réalisée par ce sujet britannique a été interrompue par une situation exceptionnelle : la boîte à thé était vide. Elle lui a fait omettre un certain nombre de tâches de son plan d'action, car la reprise de l'interruption ne s'est pas effectuée au bon endroit. Cet exemple montre s'il en était besoin le rôle non négligeable des interruptions dans la génération d'erreur. Cependant les interruptions ne possèdent pas que cet effet négatif sur l'erreur. Elles permettent également de les corriger, c'est l'objet du paragraphe suivant.

4.3.c. De l'erreur à l'interruption

Reprenons l'exemple précédent. L'interruption qui a induit en erreur notre amateur de thé n'est pas anodine. C'est une exception ou condition anormale de l'état du système qu'il utilisait. En somme, une "panne" de sa boîte à thé. La procédure de correction de cette panne, remettre du thé, est considérée par

[12] Le mot théière a été reproduit ici à la place du mot bouilloire de l'édition originale [Reason 1993] page 112 compte tenu du contexte. Le traducteur, Jean-Michel Hoc, ayant probalement été victime d'une erreur d'inférence...

Reason comme une interruption. L'utilisateur a donc dû déclencher une interruption pour corriger une panne de la machine.

Nous pouvons donc considérer que certaines tâches de correction d'erreur sont des instances particulières d'interruptions. En effet, ces tâches interrompent l'exécution normale d'un plan d'action pour apporter une correction. Une autocorrection est vue comme une instance d'une interruption interne, alors que la correction de panne est vue comme une instance d'une interruption externe. En conséquence, nous pouvons utiliser la modélisation des interruptions proposée au paragraphe 4.2 pour construire certaines des tâches de correction d'erreur.

4.3.d. Notion de singularité 🕮

Nous venons de montrer, au paragraphe 4.3.c, que les notions d'interruption et de correction d'erreur disposent d'un sous-ensemble commun. En effet, en fonction du contexte de la tâche, une interruption peut être vue comme une correction d'erreur. Une voiture qui refuse de démarrer oblige le conducteur à générer une interruption pour réparer. Et cette réparation est une correction d'erreur (en fait de panne) du système véhicule. De même, la correction d'une erreur, comme l'utilisation de la touche "BACKSPACE" sur un clavier est une interruption, pour correction, de la tâche en cours.

Cependant la correction d'une erreur peut avoir lieu en parallèle avec la tâche erronée, et n'interrompt donc pas cette tâche. Les corrections d'erreur ne sont donc pas toutes des interruptions. De même, toutes les interruptions, comme un simple coup de téléphone, ne sont pas des corrections d'erreur.

En résumé nous savons que les interruptions et les corrections d'erreur partagent des caractéristiques communes qu'il convient de traiter ensemble. Cependant ces deux notions ne sont pas identiques et disposent d'éléments n'appartenant qu'à leur ensemble, comme l'illustre la figure II-12 ci-contre.

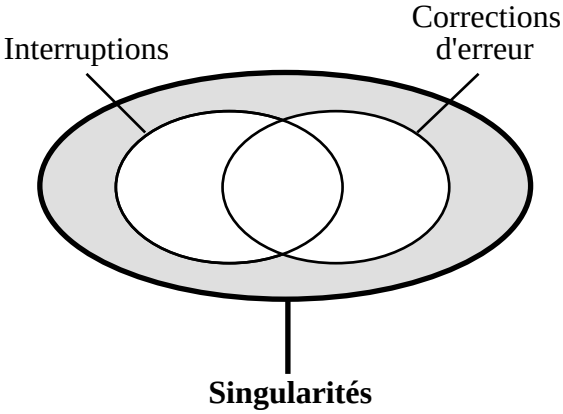

Figure II-12 : Notion de singularité.

La difficulté rencontrée à distinguer ces notions nous a amené à proposer la notion fédératrice de “singularité”. Cette notion s’inspire de la notion d’exception ou de condition anormale utilisée par exemple dans les blocs de connaissance de Boy [Boy 1989]. Cependant, la notion de singularité ne se limite pas aux seuls cas exceptionnels ou anormaux. Cette notion englobe toutes les tâches qui “singularisent” la tâche en cours, en quelque sorte tout ce qui s’éloigne de la trajectoire d’interaction normale sans erreurs ni interruptions d’aucune sorte. De plus, toutes les relations temporelles entre la singularité et la tâche singularisée sont prises en compte, contrairement aux blocs de connaissance.

En effet, la notion de singularité ne se base pas sur le type de tâche mais sur la relation temporelle entre la singularité et la tâche qu’elle singularise. Comme le montre la figure II-13, une singularité peut avoir trois relations de base avec sa tâche mère : *séquence*, *parallélisme*, ou *entrelacement*. Lorsqu’il y a séquence, c’est-à-dire vraiment interruption, deux cas sont possibles : la tâche interrompue peut reprendre ou non. Lorsqu’il y a reprise, il est alors possible de distinguer différents points de reprise : au début, en cours, à la fin, qui sont des cas particuliers utilisés couramment, ou bien en un point quelconque qui est le cas général. Cette notion de singularité basée sur les relations temporelles nous permettra d’étendre le pouvoir d’expression des outils de description de tâche au chapitre IV.

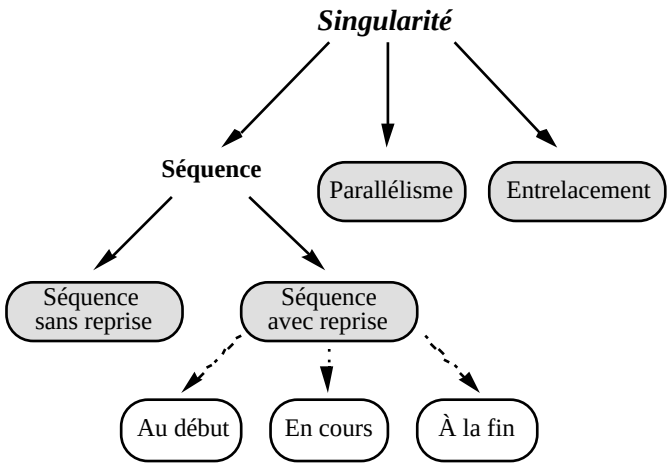

Figure II-13 : Classification des singularités.

5. Concevoir pour l’erreur [13]

Concevoir pour l’erreur est une nécessité. En effet, pour une tâche équivalente, une mauvaise réalisation de l’interface homme-machine peut augmenter considérablement le taux d’erreur. Il peut par exemple doubler,

[13] Inspiré de “Design for error” de Norman [Norman 1990].

dans une simple tâche de sélection, du fait de la modification du bouton de la souris utilisée [Jambon & Coutaz 1994]. L'importance du rôle des interruptions n'est pas à négliger, car elles peuvent être la cause d'erreurs. De plus, elles sont associées à la notion de correction d'erreur grâce à la notion fédératrice de singularité. Afin de concevoir pour l'erreur, il s'avère nécessaire d'intégrer les notions d'erreur, de correction d'erreur, et d'interruption au sein du processus de conception des interfaces homme-machine. Pour cela, il est en premier lieu nécessaire de faire un bilan des résultats disponibles :

• Les ouvrages sur l'erreur humaine sont nombreux et ses classifications multiples. Malgré cette diversité, les principaux types d'erreur se retrouvent dans presque toutes les classifications. Peu importe donc celle qui est utilisée. Ces classifications ont le mérite de sensibiliser les concepteurs à l'erreur humaine, première étape nécessaire à leur prise en compte. Mais l'utilisation de ces ouvrages est complexe, car ils sont le plus souvent uniquement descriptifs. En effet, les classifications de l'erreur permettent de comprendre a posteriori l'erreur, mais elles ne permettent pas directement de la prévoir et donc de la prévenir.

• Les méthodes d'analyse de la fiabilité humaine, comme THERP, sont un premier pas vers l'évaluation systématique du risque d'erreur. Néanmoins, elles sont encore très complexes à utiliser et réservées aux spécialistes de la fiabilité humaine. Elles ne peuvent donc être utilisées que pour des cas particuliers et des applications très critiques.

• Les heuristiques destinées à prévenir l'occurrence des erreurs sont une application indirecte des classifications de l'erreur. Immédiatement utilisables par les concepteurs, elles permettent une amélioration sensible des réalisations. Elles souffrent cependant des problèmes génériques liées aux heuristiques : elles sont nombreuses, difficiles d'accès, quelquefois contradictoires et soumises à de nombreuses interprétations divergentes. Citons à ce propos l'ouvrage de Bodart et Vanderdonckt qui a pour but de faciliter l'accès aux nombreuses règles ergonomiques présentes dans la littérature [Bodart & Vanderdonckt 1994].

• Peu de résultats sont disponibles quant à la détection des erreurs, ce domaine de recherche est encore à explorer. Il est tout de même acquis qu'une bonne conception, à l'aide d'heuristiques, facilite cette détection.

• Les mécanismes de correction des erreurs sont relativement bien cernés. Le choix des mécanismes à réaliser bénéficie maintenant de dimensions d'analyse permettant aux concepteurs de visualiser les alternatives possibles. En outre, le travail réalisé sur la modélisation des interruptions permet de constituer des bibliothèques de tâches de correction d'erreur.

La prise en compte de l'erreur humaine lors de la conception d'interfaces homme-machine est difficile. La figure II-14 ci-contre, reprenant la figure II-1 du début de ce chapitre, nous montre les voies d'actions possibles. S'il est difficile, voire même probablement impossible, de prévenir totalement l'erreur, il est néanmoins possible de limiter ses effets par la correction, indiquée en grisé sur la figure II-14. S'intéresser à l'erreur par sa correction revient en quelque sorte à prendre l'erreur à revers. C'est vers cette voie, bien adaptée à la pratique en ingénierie, que nous nous tournons maintenant.

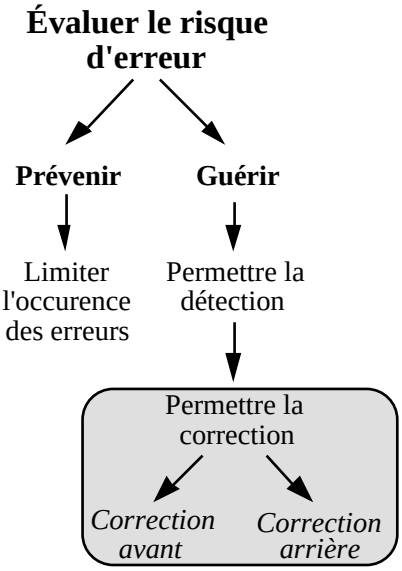

Figure II-14 : Erreurs humaines vis-à-vis de la conception des interfaces homme-machine.

Tous les résultats mis à jour doivent pouvoir s'exprimer lors des spécifications d'une interface homme-machine. Ils ont pour cela besoin d'un support. Celui-ci doit permettre, de la même manière que le dessin industriel permet d'exprimer les interactions entre les pièces mécaniques, l'expression des interactions entre l'homme et la machine. Or les formalismes disponibles ignorent pour la plupart le concept même d'erreur. C'est pourquoi, la seconde partie de ce mémoire est consacrée à l'étude des formalismes actuels, puis à leur extension aux concepts de correction d'erreur et d'interruption par l'intermédiaire de la notion de singularité.

6. Bibliographie

[Abowd et al. 1992] Abowd G.D., Coutaz J., & Nigay L. *Structuring the space of interacting system properties*. IFIP transactions / Engineering for human-computer interaction, North-Holland : Elsevier Science Publishers, 1992. p. 113-129.

[Airbus Industrie 1992] Airbus Industrie. *Flight Crew Operating Manual A320*. 1992.

[Avions Pierre Robin 1986] Avions Pierre Robin. *Manuel de vol DR400*. Avions Pierre Robin, 1986.

[Bastien et al. 1993] Bastien C.J. & Scapin D.L. *Ergonomic criteria for the evaluation of human-computer interfaces*. INRIA Rocquencourt, Juin 1993. Rapport technique N°156.

[Bodart et al. 1994] Bodart F. & Vanderdonckt J. *Guide ergonomique de la présentation des applications hautement interactives*. Namur, Belgique : Presses Universitaires de Namur, 1994.

[Boy 1989] Boy G. The block representation in knowledge acquisition for computer integrated documentation. *4th AIAA-sponsored knowledge acquisition integrated documentation for knowledge-based systems workshop, Banff, Canada*, October 1989.

[Dix et al. 1993] Dix A., Finlay J., Abowd G., & Beale R. *Human-computer interaction*. Cambridge, U.K. : Prentice Hall, 1993.

[Gray et al. 1996] Gray P.D. & Johnson C.W. Supporting error-driven design. *Third International Eurographics Workshop on Design, Specification, and Verification of Interactive systems (DSV-IS'96), Namur, Belgium*, 5-7 June 1996.

[Hartson et al. 1990] 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, p. 181-203.

[Jambon et al. 1994] Jambon F. & Coutaz J. Retours d'information sensori-moteurs des dispositifs physiques : Tâche de sélection avec une souris sans "clic". *12th Triennal Congress of the International Ergonomics Association (IEA'94), Toronto, Canada*, Août 1994. vol. 4(6), p. 464.

[Johnson et al. 1996] Johnson C.W. & Telford A.J. Using formal methods to analyse human error and system failure during accident investigations. *Human-Computer Interaction Journal*, November 1996.

[Jones 1979] Jones C.P. *Automatic fault protection in the Voyager spacecraft*. American Institute of Aeronautics and Astronautics, 1979. Paper n°79-1919.

[Krakowiak 1985] Krakowiak S. *Principes des systèmes d'exploitation des ordinateurs*. Dunod informatique, 1985.

[Lenman et al. 1994a] Lenman S. & Robert J.-M. A framework for error recovery. *International Ergonomics Association (IEA'94), Toronto, Canada*, August 15-19 1994a. vol. 6(6), p. 374-376.

[Lenman et al. 1994b] Lenman S. & Robert J.-M. Investigating the granularity of the Undo function in human-computer interfaces. *Applied Psychology: An international review*, 1994b. vol. 43, n° 4 (special issue on Human Errors), p. 543-564.

[Nicolet et al. 1990] Nicolet J.-L., Carnino A., & Wanner J.-C. *Catastrophes ? Non merci ! La prévention des risques technologiques et humains*. Éditions Masson, 1990.

[Nielsen 1994] Nielsen J. Enhancing the explonatory power of usability heuristics. *Human Factors in Computing Systems (CHI'94), Boston (MA), USA*, 24-29 April 1994. vol. conference proceedings(2), p. 152-153.

[Norman 1990] Norman D.A. *The design of every day things*. New York, USA : Doubleday Currency, 1990.

[O'Conaill et al. 1995] O'Conaill B. & Frohlich D. Timespace in the workplace: Dealing with interruptions. *Human factors in computing systems (CHI'95), Denver (Colorado), USA*, May 7-11 1995. vol. Conference companion(2), p. 262-263.

[Pendibidou 1982] Pendibidou J.-M. Présentation du projet THÉRÈSE. *Technique et Science Informatiques*, 1982. vol. 1, n° 3, p. 253-257.

[Pérez-Quiñones et al. 1996] Pérez-Quiñones M.A. & Sibert J.L. Negaciating user-initiated cancellation and interruption requests. *Conference on Human Factors in Computing Systems (CHI'96), Vancouver (British Columbia), Canada*, April 14-18 1996. vol. Conference companion(2), p. 267-268.

[Racca 1992] Racca E. Le concept de CRM / Crew Ressource Management. *Le transpondeur / bulletin de laison*, Avril 1992. n° 7, p. 55-59.

[Reason 1993] Reason J. *L'erreur humaine*. Paris, France : Presses Universitaires de France, 1993.

[Rouncefield et al. 1994] Rouncefield M., Hughes J.A., Rodden T., & Viller S. Working with "constant interruption": CSCW and the small office. *Conference on Computer Supported Cooperative Work (CSCW'94), Chapel Hill (North California), USA*, October 22-26 1994. p. 275-286.

[Swain et al. 1983] Swain A.D. & Guttmann H.E. *Handbook of human reliability analysis with emphasis on nuclear power plant applications*. Nuclar Regulatory Commission (NUREG), August 1983. Final report NUREG/CR-1278F.

[Walker et al. 1995] Walker W. & Cragon H.G. Interrupt processing in concurent processors. *Computer*, June 1995. vol. 28, n° 6, p. 36-45.

[Woods 1990] Woods D.D. *Modeling and predicting human error*. Human performance models for computer-aided engineering, Washington : Academic Press, 1990. p. 248-274.

[Yang 1992] Yang Y. Anatomy of the design of an undo support facility. *International Journal of Man-Machine Studies*, 1992. vol. 36, n° 1, p. 81-95.

Seconde partie

FORMALISMES

Ce que l'on conçoit bien, s'énonce clairement,
et les mots pour le dire viennent aisément.

Nicolas Boileau (1630-1711)

CHAPITRE III

SPÉCIFICATIONS FORMELLES

1. Introduction

Nous utilisons tous, sans même en être toujours conscients, de nombreux formalismes. Ils nous permettent par exemple de dessiner un plan, de rédiger une lettre, ou encore de communiquer verbalement avec nos semblables. Ces formalismes prennent l'apparence d'une norme, de formalités administratives, ou encore de traits de langage. Ces formalismes ont pour but de donner à l'information, c'est-à-dire au fond, la forme ‹‹ dont la précision et la netteté excluent toute équivoque ›› [Robert 1992]. À cet aspect positif des formalismes s'oppose leur aspect négatif : créés de toute pièce, ils imposent également un codage de l'information. Un document juridique par exemple, peut être rejeté s'il ne respecte pas le formalisme imposé. De même, la non-connaissance du formalisme utilisé pour rédiger un document, en quelque sorte la clé, peut rendre difficile la compréhension du fond.

Malgré ces travers, l'usage des formalismes s'est répandu dans de nombreux domaines car ils répondent à plusieurs besoins. Ils permettent notamment de supprimer autant que possible les ambiguïtés et les erreurs d'interprétation. Ainsi, grâce à l'usage des formalismes, l'information peut se transmettre sans perte entre celui qui l'a créée, et celui qui l'utilise. La mécanique par exemple, fait appel au dessin industriel pour transmettre l'information entre celui qui conçoit une pièce et celui qui la réalise. De même, l'informatique utilise les langages de programmation afin permettre au programmeur de faire fonctionner la machine. Elle fait également appel aux formalismes de spécification afin de faciliter la communication entre celui qui analyse le problème, celui qui écrit le code informatique, et celui qui utilise l'application. C'est à cette sorte de dessin industriel adapté à l'informatique, et plus précisément à l'ingénierie des interfaces homme-machine, que nous nous intéressons maintenant.

Au terme formalisme sont associées les notions de spécification formelle et de méthode formelle. En résumant à l'extrême, un formalisme peut être vu comme une notation permettant de rédiger des spécifications formelles lesquelles sont utilisées au sein d'une méthode formelle. Une méthode formelle peut couvrir tout ou partie du cycle de développement d'un logiciel, et y apporter des outils permettant la vérification, ou encore la génération automatique de code. Les possibilités et l'usage pratique des méthodes formelles font l'objet de nombreux mythes. Sept d'entre-eux ont été recensés

et dénoncés par Hall [Hall 1990], puis sept autres par Bowen et Hinchey [Bowen & Hinchey 1995]. Énumérer ces quatorze mythes nous éloignerait du sujet de ce mémoire. Cependant, sept arguments relatifs à l'usage et à l'importance des méthodes formelles ont été mis en lumière au cours d'une discussion du forum électronique[14] "comp.human-factors". Ces arguments nous intéressent plus particulièrement car ils s'appliquent directement au domaine de l'ingénierie des interfaces homme-machine :

• Les méthodes formelles sont très utiles lors de la conception de systèmes critiques, car les conséquences d'une conception erronée de l'interface homme-machine peut dépasser de plusieurs ordres de grandeur le surplus de coût dû à l'usage d'une spécification formelle lors de la conception.

• Les méthodes formelles sont un élément de base du cycle de développement d'un logiciel car elles permettent de poser les problèmes de l'interface dès les premiers stades de la conception. Cependant leur utilisation dans le milieu industriel reste limitée.

• La spécification formelle permet aux concepteurs de disposer d'une description explicite de l'interface à réaliser. Il leur est ainsi possible de séparer la partie conception de la partie réalisation : ils ne sont plus contraints d'avoir recours systématiquement à l'utilisation de maquettes, de prototypes, ou à la réalisation complète pour illustrer le fonctionnement visible du système.

• Les spécifications formelles sont souvent jugées illisibles. Cet argument s'effondre si le concepteur consacre un minimum de temps à leur apprentissage. De plus, les difficultés rencontrées lors de cet apprentissage sont souvent moindres que celles des langages de programmation. Cependant, si malgré une période suffisante d'apprentissage les programmeurs n'arrivent ni à les apprendre ni à les lire, il y a effectivement problème.

• L'usage des notations pose le problème des artefacts de spécification, c'est-à-dire que la notation utilisée peut influencer le concepteur dans ses choix de conception. La notation biaise la spécification. Un problème similaire se pose avec les langages de programmation : les programmeurs renâclent parfois à réaliser ce qui est complexe à décrire avec le langage qu'ils utilisent ; les fonctionnalités du logiciel sont alors légèrement

[14] Newsgroup

modifiées pour “convenir” au langage. Ce problème est insidieux car il n’est pas dû à un pouvoir d’expression limité du formalisme, mais plutôt à l’utilisabilité de ce pouvoir d’expression.

• Une spécification formelle n’a pas besoin d’être réellement formelle, c’est-à-dire définie rigoureusement, pour être utile aux concepteurs. On parle alors plutôt de notation semi-formelle. Même ambiguë, une notation par son pouvoir d’abstraction permet de faciliter la communication entre les membres d’une équipe de conception. La notation UAN que nous décrivons dans ce chapitre en est un exemple.

• Le pouvoir d’expression d’un formalisme doit s’évaluer selon deux facettes : une interface doit être aisée à spécifier, mais sa spécification doit également être facile à interpréter. Une interface ne doit pas seulement être facile à spécifier par celui dont c’est le rôle. Il est en effet nécessaire que celui qui doit l’interpréter puisse le faire avec un effort comparable. Dans le cas contraire, une spécification formelle risque d’être inutilisée par les programmeurs à cause d’une interprétation trop difficile.

Les méthodes formelles appliquées au processus de conception des interfaces homme-machine offrent aux concepteurs de nombreux services : notation formelle pour l’écriture du cahier des charges, vérifications formelles de la spécification, évaluation automatisée de l’interface, génération automatique de code, de l’aide en ligne, etc. Dans le cadre de ce mémoire nous nous intéressons principalement à l’utilisation des formalismes comme notations formelles (ou semi-formelles). Celles-ci ont pour finalité l’écriture du cahier des charges, l’échange de spécifications entre les membres d’une équipe de conception, ou encore la constitution d’une mémoire d’entreprise. Cependant, les notations étudiées et les extensions proposées ne sont pas limitées à ces services.

Ce chapitre débute par un tour d’horizon des classifications des formalismes adaptés à la spécification des interfaces. C’est l’objet de la section 2 qui introduit également une proposition de méta-classification. À l’issue de ce tour d’horizon, la section 3 se consacre à l’utilisation pratique de ces classifications dans le but de choisir un formalisme adapté aux besoins des concepteurs. Cette section retranscrit également les choix faits dans le cadre de ce mémoire. La section 4 conclut ce chapitre en s’intéressant aux

problèmes des classifications actuelles, ainsi qu'aux conséquences des choix effectués dans les sections précédentes de ce chapitre.

2. Classifications

L'ingénierie des interfaces homme-machine dispose de quelques dizaines de formalismes aptes à spécifier l'interface d'un système. Ces formalismes possèdent des qualités et des finalités souvent très différentes. Se pose alors, pour le concepteur, le problème du choix d'un formalisme adapté à ses besoins. Ce choix peut être guidé par l'étude des classifications, publiées dans la littérature, évaluant et comparant les formalismes entre-eux. À l'analyse de ces classifications, nous constatons que leurs auteurs font principalement appel à deux approches :

• *L'approche théorique ou bibliographique* consiste à rechercher dans les publications disponibles les capacités des formalismes étudiés. Elle se heurte bien souvent à l'absence de documents de synthèse décrivant précisément les formalismes étudiés, en quelque sorte leur notice d'utilisation. Cette approche est de loin la plus rapide et la plus utilisée. Elle peut être effectuée à moindre effort car elle ne nécessite pas une parfaite maîtrise des formalismes. Cette facilité permet en conséquence la constitution de classifications portant sur de nombreux formalismes, permettant ainsi d'obtenir une vue d'ensemble du problème. Cependant, ces évaluations sont dépendantes de ce que diffusent leurs auteurs. En conséquence, ce ne sont que les capacités publiées des formalismes qui sont évaluées, et non pas forcément les capacités réelles.

• *L'approche pratique,* au contraire, consiste à choisir un cas d'étude, un ensemble de caractéristiques, puis à évaluer la capacité des formalismes étudiés à mettre en lumière ces caractéristiques. Cette approche nécessite une parfaite connaissance des formalismes utilisés, voire même une bonne expérience de leur utilisation. Il est souvent préférable que cette étude soit réalisée par les experts de chacun des formalismes étudiés. En conséquence, compte tenu de la quantité importante de travail à fournir, cette approche est peu utilisée. Elle s'effectue cependant lors d'ateliers de travail comme ACM-CHI'96 [Rouff 1996] et nous l'avons choisie lors de l'évaluation des possibilités d'expression des interruptions [Jambon 1995]. L'intérêt de cette approche est de révéler les capacités démontrées des formalismes choisis.

L'approche pratique rend mieux compte des capacités réelles des formalismes, mais son coût est souvent rédhibitoire. C'est pourquoi, lors du choix d'un formalisme, il est souvent plus intéressant de se tourner dans un premier temps vers une approche bibliographique. Celle-ci permet de réaliser un premier tri en élaguant un certain nombre de formalismes, selon des critères que nous détaillerons à la section 3. Une fois le nombre de formalismes à étudier restreint, l'approche pratique est alors envisageable de manière à affiner le choix. Afin d'illustrer les différentes approches, nous allons maintenant nous intéresser aux différents types de classification existants.

Cette section propose un tour d'horizon des classifications des formalismes proposées dans la littérature sous la forme d'une méta-classification. Les paragraphes 2.1 à 2.4 décrivent respectivement les classifications fondées sur le pouvoir d'expression, l'utilisabilité, l'origine, et la finalité des formalismes. À l'issue de cette méta-classification, nous évoquons les Principes de Namur. Détaillées au paragraphe 2.5, ils constituent une sorte de cahier des charges que devraient respecter les formalismes destinés à l'ingénierie des interfaces.

2.1. Classifications selon le pouvoir d'expression

Le pouvoir d'expression d'un formalisme, appelé aussi puissance d'expression, représente sa capacité à décrire les propriétés d'un système. Coutaz parle de déterminer le domaine représentable du système [Coutaz 1990] page 349. L'adéquation entre la capacité d'expression d'un formalisme et les besoins de la spécification étant la condition sine qua non à son utilisation, ce critère est de loin le plus utilisé pour comparer les formalismes.

Parmi les nombreuses analyses sur le sujet, nous avons retenus trois études représentatives. Les deux premières adoptent une approche bibliographique : l'une, proposée par Brun et Beaudoin-Lafon, couvre un grand nombre de formalismes selon des critères généraux (§ 2.1.a), tandis que l'autre, produite par Balbo, utilise des critères plus précis, mais est moins étendue (§ 2.1.b). La troisième étude, proposée par Coutaz, Paterno, Faconti, et Nigay, adopte une approche pratique et limitée : elle explicite seulement deux formalismes, et s'intéresse uniquement à la spécification de la multimodalité (§ 2.1.c).

2.1.a. Classification de Brun et Beaudoin-Lafon

Brun et Beaudoin-Lafon proposent une taxonomie et une évaluation des formalismes destinés à la spécification des systèmes interactifs [Brun & Beaudoin-Lafon 1995]. Cette taxonomie s'appuie sur l'origine des formalismes. Nous la décrivons en détail au paragraphe 2.3. L'évaluation, quant à elle, prend en compte douze critères, dont six ont trait au pouvoir d'expression :

- Description des actions et des tâches utilisateur
- Description de l'état de l'interface et du feedback
- Description de la synchronisation des actions et des contraintes de temps
- Description du parallélisme des actions
- Description de la présentation de l'interface
- Prise en compte des erreurs de manipulation

Cette étude basée sur une approche bibliographique est, de l'aveu même de ses auteurs, entachée de subjectivité. C'est pourquoi les auteurs ont choisi d'évaluer chaque critère selon une échelle discrète : bon ("good"), moyen ("OK"), faible ("poor"), mauvais ("bad"). Ainsi, plutôt que de donner un ordre strict entre les formalismes, cette évaluation permet de faire émerger leurs qualités principales. Couvrant un grand nombre de formalismes, dix-sept, cette classification offre un tour d'horizon assez exhaustif des formalismes aptes à la spécification des systèmes interactifs. Présentés sous forme arborescente selon leur origine comme le montre la figure III-1, les formalismes étudiés peuvent se distinguer visuellement par leur capacité à couvrir tel ou tel critère en fonction du remplissage du carré représentant le critère recherché. Notons qu'aucun des formalismes étudiés ne prend en compte les erreurs de l'utilisateur. Après ce tour d'horizon illustré par l'étude de Brun et Beaudoin-Lafon, nous allons maintenant nous intéresser à une classification plus précise mais moins étendue : celle proposée par Balbo.

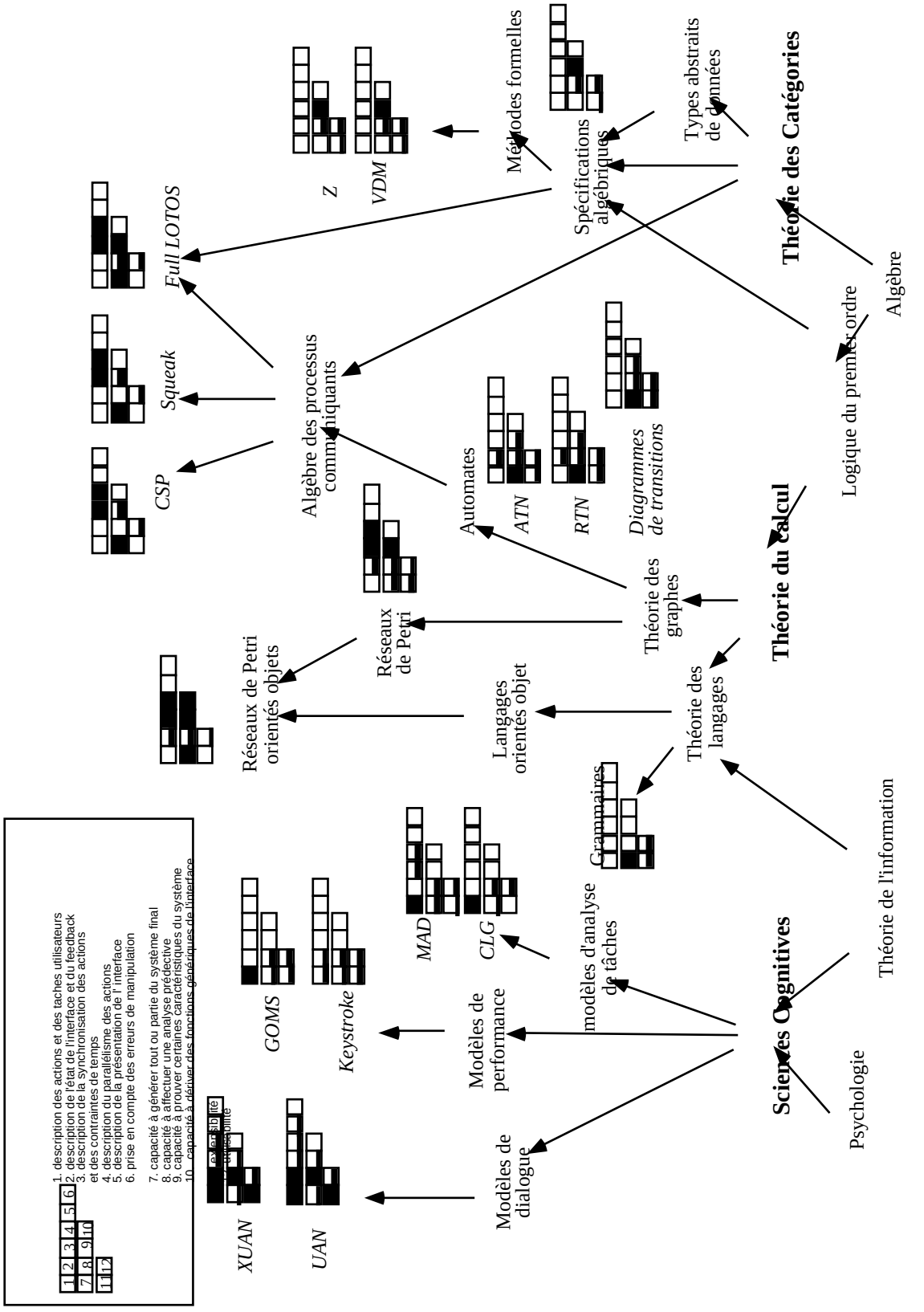

Figure III-1 : Taxonomie et évaluation des formalismes existants [Brun et al. 1995]. Le remplissage du carré est fonction de la capacité du formalisme étudié à couvrir le critère évalué. Traduction et schéma des auteurs.

2.1.b. Classification de Balbo

Balbo considère quatre critères permettant d'évaluer la puissance d'expression des formalismes adaptés à la description des tâches et des spécifications externes : le nombre de niveaux d'abstraction, l'expression des relations temporelles entre les tâches, leurs attributs, et l'expression des entrées/sorties [Balbo 1994]. Cette étude, menée selon une approche bibliographique, porte sur dix formalismes. Tout comme Brun et Beaudoin-Lafon, les critères choisis sont évalués sur une échelle discrète : caractère souligné ou explicite (+), caractère peu souligné ou implicite (+-), caractère pas ou peu souligné (-), et caractère non évalué (?). Notons que ces jugements portent sur la puissance d'expression mais aussi implicitement sur l'utilisabilité des formalismes. Le bilan de l'évaluation est résumé sous la forme de deux tableaux figures III-2 et III-3 regroupant les quatre critères évoqués :

• *Le nombre de niveaux disponibles* (colonne 3 figure III-3) peut être laissé à l'initiative du concepteur (qqs) ou bien être fixé, comme pour GOMS [Card, Moran, & Newell 1983] à quatre. Selon les formalismes, les niveaux de description ont un usage différent, ce qui rend cette comparaison quelque peu abstraite. Pour GOMS les niveaux correspondent à des niveaux d'abstraction alors que pour UAN [Hartson, Siochi, & Hix 1990] ou MAD [Scapin & Pierret-Golbreich 1989] ils représentent une décomposition en tâches et sous-tâches. Ce critère permet néanmoins de faire la distinction entre les formalismes ayant un nombre de niveaux fixés, donc à signification précise, et ceux à nombre de niveaux laissé à l'appréciation du concepteur, destinés à exprimer la différence de granularité entre les descriptions.

• *Les relations temporelles* entre les tâches (colonnes 1-7 figure III-2) évaluées sont au nombre de sept :

- La séquence (seq) : A puis B
- L'alternative (alter) : A ou B
- La composition (com) : A et B dans un ordre quelconque
- L'itération (itér) : A autant de fois que l'on désire
- L'entrelacement (entr) : A en même temps que B, de manière entrelacée
- Le parallélisme vrai (//) : A et B en même temps
- L'expression de délais entre l'exécution de plusieurs tâches (dél)

• *Les attributs des tâches* (colonnes 8-9 figure III-2) pris en compte pour l'évaluation sont :

- La capacité de préciser le caractère obligatoire ou facultatif d'une tâche (obli)
- La capacité de préciser le déclencheur de la tâche (décl) : utilisateur ou système

• *L'expression des entrées/sorties* (colonne 5 figure III-3) dont est capable le formalisme peut soit se limiter aux entrées du système (E), soit permettre la spécification des entrées et des sorties (E+S).

	FORMALISME								
	Relations entre tâches							Attributs	
Modèles	seq	alter	com	itér	entr	//	dél	obli	décl
ETAG	+	-	?	?	-	-	-	-	-
UAN	+	+	+	+	+	+	+	-	-
DIANE	+	+	+	+	+-	+	+	+	+
JSD*	+	+	+	+	+-	-	-	+-	+
MAD	+	+	+	+	+-	+	-	-	-
CLG	+	+	+	+	-	-	-	+	+-
GOMS	+	+	+	+	-	-	-	-	-
ETIT	?	?	?	?	?	?	?	?	?
SIROCO	+	+	+	+-	+	+	-	-	-
ADEPT	+	+	+	+	-	-	-	-	-

Figure III-2 : Tableau récapitulatif des relations/attributs offerts par le formalisme du modèle en regard. Adapté de [Balbo 1994].

Comparés aux critères de Brun et Beaudoin-Lafon, les critères choisis par Balbo sont plus précis. Par exemple, Balbo étudie sept relations temporelles alors que Brun et Beaudoin-Lafon se contentent du parallélisme vrai. D'autres critères sont comparables, comme l'expression des entrées/sorties chez Balbo et l'expression de l'état de l'interface et des retours d'information du système chez Brun et Beaudoin-Lafon. Notons cependant que la capacité à exprimer les tâches est vue comme un élément de la puissance d'expression d'un formalisme par Brun et Beaudoin-Lafon, alors que Balbo le considère comme un élément directeur, sujet du paragraphe 2.3. En résumé, ces deux études suivant l'approche bibliographique sont assez comparables, avec une étude volontairement plus axée sur les tâches chez Balbo. Nous allons maintenant

détailler l'étude pratique proposée par Coutaz, Paterno, Faconti, et Nigay, s'intéressant au problème de la multimodalité.

2.1.c. Spécification de la multimodalité

Coutaz, Paterno, Faconti, et Nigay ont appliqué deux formalismes, UAN [Hartson et al. 1990] et LOTOS [Garavel 1989], à la spécification de la multimodalité [Coutaz, Paterno, et al. 1993b]. Cette démarche, suivant une approche pratique, s'est donnée comme application sujet MATIS, un système multimodal de réservation et de renseignements sur les horaires d'avion [Nigay 1994]. Notons que les deux formalismes choisis par les auteurs, UAN et LOTOS, ont une origine très éloignée : UAN fut créé à l'origine pour décrire l'interaction homme-machine à manipulation directe, tandis que LOTOS est l'objet d'une norme ISO et est destiné à la spécification formelle des protocoles de communication [ISO-8807:1989 1995].

Cette comparaison de deux formalismes met en lumière les limites de leur pouvoir d'expression, mais aussi leur complémentarité. Sans entrer dans les détails, notons que les auteurs soulignent l'absence d'opérateur terminant la tâche en cours dans UAN, alors que LOTOS en dispose. Au contraire, UAN permet le parallélisme vrai, contrairement à LOTOS. De même, les auteurs font remarquer la qualité de l'expression formelle de LOTOS, ce qui n'est pas une surprise compte tenu de son origine. Cependant, malgré son intérêt certain, on peut regretter que cette évaluation manque quelque peu de profondeur. En effet, les auteurs ne se sont pas définis préalablement un sous-ensemble des caractéristiques de l'interface à évaluer. Cette définition aurait alors permis de conduire l'évaluation de manière plus formelle, en établissant une grille d'évaluation permettant de comparer point par point les deux formalismes étudiés.

2.1.d. Évaluer le pouvoir d'expression

Ces trois exemples de classifications sont comparables sur quelques critères, comme par exemple l'expression du parallélisme qui est repris par les trois classifications évoquées. Cependant, ces classifications sont le plus souvent complémentaires car le choix des critères d'évaluation dépend principalement de la finalité des classifications.

De même, la définition d'un critère peut varier selon les auteurs. Il est ainsi souvent difficile de comparer les résultats obtenus. Par exemple, le critère "relations entre les entrées et les sorties" utilisé par Balbo n'est comparable

aux deux critères évoqués par Brun et Beaudoin-Lafon “description de l’état de l’interface et des retours d’information du système” et “expression de la présentation de l’interface” que pour un même niveau d’abstraction. En effet, l’une des difficultés inhérente à la réalisation de classifications est la définition formelle et précise des critères évalués.

En corollaire, l’évaluation des critères, même définis formellement, est subjective. Par exemple, Brun et Beaudoin-Lafon suggèrent que les notations UAN et XUAN satisfont bien au critère “description de l’état de l’interface et du feedback”. Or, ces notations ne sont pas bien adaptés à la description de la multimodalité. Il est donc nécessaire de définir formellement non seulement les critères à évaluer, mais aussi une échelle de mesure précise associée à ces critères.

La variabilité des critères et la subjectivité de leur évaluation ont pour conséquence de rendre difficile la reproductibilité des résultats de ces études. C’est pourquoi l’étude du pouvoir d’expression des formalismes n’est pas discriminante. Il est nécessaire de se tourner vers d’autres types de classifications, comme celles relatives à l’utilisabilité auxquelles nous nous intéressons maintenant.

2.2. Classifications selon l’utilisabilité

Au-delà de la puissance d’expression d’un formalisme, il est impératif de s’intéresser aussi à sa facilité d’utilisation. Appelée le plus souvent utilisabilité, elle apparaît comme primordiale lorsque la taille de l’application spécifiée sort des exemples d’école. Par exemple, la description complète de l’application MATIS à l’aide de la notation UAN [Coutaz, Faconti, et al. 1993a] est volumineuse et très difficile à lire, même lorsque le lecteur connaît déjà l’application.

Ce critère d’évaluation, cité dans l’introduction de ce chapitre parmi les sept arguments relatifs à l’usage des méthodes formelles, est repris par de nombreux auteurs sous différentes formes. Par exemple, Brun et Beaudoin-Lafon incluent l’utilisabilité parmi leurs critères d’évaluation (Cf. figure III-1 critère n°12) [Brun et al. 1995]. Balbo, quant à elle, décline l’utilisabilité d’un formalisme selon deux catégories : lisibilité et facilité d’utilisation (Cf. colonnes 6, et 7 figure III-3) [Balbo 1994]. De même, Abowd, Bowen, Dix,

Harrison, et Took y font allusion sous le non de lisibilité (“readability”) [Abowd, Bowen, et al. 1989].

	FORMALISME						
	Elément	Rigueur	Puiss. d'expression			Lisibilité	Facilité
Modèles	directeur	sémantique	Niv.	Rel.	E/S		d'utilisat°
ETAG	Concept	?	4		E	-	?
UAN	Tâche	-	qqs		E+S	+-	+-
DIANE	Tâche	+	qqs		E+S	+	+
JSD*	Tâche	?	qqs	*cf.*	E+S	+-	?
MAD	Tâche	?	qqs	*fig.*	E	+-	?
CLG	Tâche	-	6	*III-2*	E+S	+-	?
GOMS	Tâche	-	4		E	+	+
ETIT	Tâche	-	2		E	+	?
SIROCO	Concept	+	1		E+S	+-	+-
ADEPT	Tâche	+	qqs		E	+	?

Figure III-3 : Tableau récapitulatif des systèmes étudiés en fonction du formalisme. Adapté de [Balbo 1994].

L’évaluation de l’utilisabilité d’un formalisme est un critère particulièrement subjectif. Cette évaluation est en effet fortement influencée par les connaissances préalables et la formation de l’évaluateur. Afin de minimiser cet aspect subjectif, Johnson s’est tourné vers une approche pratique de l’étude de l’utilisabilité [Johnson 1996a]. Le protocole de l’étude menée par Johnson consiste à présenter à des concepteurs et des utilisateurs la description de deux interfaces selon trois formalismes : UAN, les diagrammes étapes-transitions, et la logique temporelle. Les deux interfaces étudiées implémentent un gestionnaire de fenêtres et un système de contrôle de turbine. Les six descriptions, représentant tous les cas possibles, sont présentées aux sujets. Ceux-ci répondent alors aux questions posées par l’évaluateur portant sur la compréhension de la description et sur l’utilisabilité du formalisme sous-jacent. Cette étude apporte de nombreuses informations. Parmi celles-ci, l’auteur fait remarquer que la logique temporelle a provoqué le moins d’erreurs de compréhension. Cependant, c’est cette même logique temporelle qui fut jugée la moins agréable à utiliser par les sujets. L’étude de Johnson est intéressante à la fois par son approche pratique, et par le choix des formalismes et des exemples. En effet, les trois formalismes utilisés ont des origines très éloignées. Quant aux exemples d’interfaces, ils sont représentatifs des interfaces du monde réel appartenant à deux domaines

complémentaires : un gestionnaire de fenêtres représentatif des interfaces communes des applications informatiques, et un système de contrôle de turbine représentatif des interfaces à fortes contraintes temporelles. L'intérêt d'une telle étude est de s'être affranchie d'une grande partie des subjectivités en migrant la tâche d'évaluation vers des sujets indépendants. Cependant les résultats sont presque décevants. En effet, aucun formalisme ne se distingue véritablement les autres, ils ont en fait des capacités et des lacunes qui se compensent.

L'utilisabilité d'un formalisme peut se regarder selon deux facettes : celle de l'écriture, c'est-à-dire la tâche de celui qui spécifie l'interface, et celle de la lecture, c'est-à-dire la tâche de celui qui la réalise. En règle générale, l'équipe de conception est différente de l'équipe chargée de réaliser l'interface. Cependant, les difficultés d'utilisation d'un formalisme se doivent d'être partagées entre les deux équipes. Il est en effet peu judicieux de concevoir un formalisme facile à écrire, mais très difficile à interpréter. Dans ce cas, l'équipe chargée de réaliser le système risque de se détourner de l'utilisation trop coûteuse des spécifications formelles, les rendant alors inutiles. Il est même préférable de privilégier une lecture facile, de manière à rendre plus aisé le travail de l'équipe qui réalise. En effet, l'équipe de conception est souvent aidée dans sa tâche de spécification par une vue d'ensemble de l'interface, ce que l'équipe chargée de réaliser le système doit acquérir au cours de l'interprétation des spécifications. De plus, il ne faut pas négliger certains facteurs sociaux : pour l'équipe de conception, la description formelle de l'interface est l'aboutissement de leur travail, alors que ce n'en est que l'ordre d'exécution pour l'équipe chargée de la réaliser .

Le critère de l'utilisabilité ne peut justifier à lui seul le choix d'un formalisme. L'utilisabilité d'un formalisme doit être vue plutôt comme un catalyseur de son pouvoir d'expression, c'est-à-dire que son utilisabilité ne peut vraiment s'exprimer que si son pouvoir d'expression est déjà suffisant. Nous allons maintenant nous intéresser à une approche radicalement différente, c'est-à-dire à l'étude des formalismes selon leur origine.

2.3. Classifications selon l'origine

L'étude de l'origine d'un formalisme est très révélatrice des qualités que l'on peut en attendre. Ce type de classification ne peut évidemment pas s'effectuer selon une approche pratique.

Balbo, par exemple, s'est intéressée aux éléments directeurs des formalismes. Selon elle, ‹‹ le formalisme utilisé pour l'expression d'un modèle reflète l'approche dans l'analyse du problème ›› [Balbo 1994] page 72. Balbo distingue les formalismes dont l'objet central de la description est la tâche, et ceux dont les objets directeurs sont les concepts du domaine. Dans ce cas, le formalisme s'apparente à une approche objet. Les tâches sont alors vues comme des méthodes appliquées aux objets. La classification de Balbo s'intéresse également à la rigueur sémantique des formalismes. Ce critère est difficilement associable au pouvoir d'expression ou à l'utilisabilité, c'est pourquoi nous avons préféré l'associer à l'origine des formalismes, car il en est totalement dépendant. Balbo résume les résultats de l'évaluation de ces critères dans les colonnes 1 et 2 de la figure III-3.

De manière générale les auteurs utilisent l'origine ou le type des formalismes comme squelette de leurs classifications. Harrisson et Duke utilisent trois catégories [Harrison & Duke 1994], alors que Abowd, Bowen, Dix, Harrison, et Took huit [Abowd et al. 1989]. Comme le soulignent Brun et Beaudoin-Lafon, les formalismes provenant d'une origine commune ont le plus souvent des qualités proches [Brun et al. 1995]. Leur classification est l'une des plus complètes et des plus détaillées. Celle-ci est basée sur trois origines principales : les sciences cognitives, la théorie du calcul, et la théorie des catégories. La représentation graphique utilisée par les auteurs afin de présenter leur classification facilite en outre grandement son interprétation. En effet comme le montre la figure III-1, en regard de chaque formalisme est indiquée son évaluation selon les douze critères utilisés. Le lecteur peut ainsi facilement associer les qualités d'un formalisme, par exemple la description des tâches, avec son origine, ici les sciences cognitives.

L'origine d'un formalisme est une donnée très intéressante car elle permet d'en révéler les qualités a priori. Il est ainsi possible de distinguer les ensembles de formalismes ayant des qualités proches. Un premier choix peut alors être fait sur cette base. Cependant au premier abord, l'origine d'un formalisme est une information souvent assez difficile à obtenir sans une certaine expertise du domaine, expertise que ne possède généralement pas le concepteur à la recherche d'un formalisme correspondant à ses besoins. C'est pourquoi nous nous intéressons maintenant aux classifications basées sur la finalité. Plus aisée à obtenir et non moins utile, ces classifications permettent d'orienter le choix d'un formalisme selon le but que le concepteur s'est fixé.

2.4. Classifications selon la finalité

Exception faite des formalismes “détournés” de leur finalité première comme LOTOS ou Z, les formalismes utilisés pour la spécification des interfaces homme-machine ont été créés, ou étendus à partir de formalismes existants, afin de répondre à des objectifs bien précis. Ce sont ces objectifs que nous définissons comme finalités. Elles représentent les étapes de conception d’une interface comme l’analyse de tâche, mais également les capacités de génération des formalismes, comme par exemple la génération de l’aide en ligne.

Balbo nous donne un bon aperçu des différentes finalités possibles des formalismes. Elle s’y intéresse en effet selon cinq catégories : analyse des besoins, conception de l’interface homme-machine, conception logicielle, réalisation, et évaluation [Balbo 1994]. L’analyse de ces critères est rassemblée sous la forme d’un tableau figure III-4. Remarquons que les trois premiers critères s’intéressent principalement à la facette notation formelle des formalismes. Ils sont ici utilisés comme moyen de communication, ce qui n’exclut pas l’utilisation de leurs capacités de génération. C’est justement à ces capacités que s’intéressent plus particulièrement Brun et Beaudoin-Lafon [Brun et al. 1995]. Elles font appel à l’aspect langage de programmation fonctionnel des formalismes. Les auteurs distinguent quatre aptitudes des formalismes, dont les deux premières sont similaires à deux des finalités de Balbo : génération de tout ou partie du système final, évaluation prédictive, preuves sur les propriétés du système, et dérivation de fonctions génériques propres aux interfaces.

L’évaluation des formalismes selon leur finalité peut être vue comme une synthèse des précédentes classifications. Ce n’en est en fait que le point de départ. En effet, deux formalismes capables de participer à l’analyse des besoins comme UAN et GOMS selon Balbo, recouvrent en fait des réalités très distinctes : le premier détaille précisément les actions sur l’interface alors que le second s’intéresse plus particulièrement à l’aspect cognitif. Cependant, l’intérêt d’une classification basée sur la finalité est d’être directement utilisable par le concepteur. Celui-ci peut en effet facilement déduire de ses besoins la finalité du formalisme qui peut lui convenir.

Modèles	FINALITE				
	Analyse des besoins	Concept° IHM	Concept° Logicielle	Réalisat°	Evaluat°
ETAG	-	+	?	en projet	?
UAN	+	+	-	-	-
DIANE	-	+	+	-	-
JSD*	+	+	?	?	-
MAD	+	+	en projet	en projet	-
CLG	+	+	-	-	+
GOMS	+	-	-	-	+
ETIT	+	-	-	-	+
SIROCO	-	-	+	+	-
ADEPT	+	+	+	+	-

Figure III-4 : Tableau récapitulatif des modèles étudiés en fonction de leur finalité dans le processus de développement des logiciels. Adapté de [Balbo 1994].

Ce tour d'horizon des classifications étant terminé, et avant de passer au choix d'une classification, il est nécessaire d'évoquer les Principes de Namur. En effet ces principes, détaillés ci-après, apportent une contribution importante au domaine.

2.5. Les Principes de Namur

Les Principes de Namur ("The Namur Principles") ont été établis récemment par un groupe de chercheurs lors de l'atelier d'Eurographics DSV-IS'96 [Johnson 1996b]. Ces principes constituent une base de critères pour l'évaluation des notations aptes à la spécification des interfaces homme-machine. Selon leurs auteurs, l'utilité des notations peut s'évaluer selon les onze critères suivants :

. *Capacité à abstraire* ("Support for abstraction") : une spécification doit pouvoir exprimer différents niveaux de détails ou d'abstraction, de manière à s'adapter aux différentes phases de conception. Un exemple de cette capacité d'abstraction est la décomposition en tâche et sous-tâche.

. *Capacité à structurer* ("Support for structuring") : les différents aspects d'une interface doivent pouvoir se spécifier séparément, afin de pouvoir être réalisés par des équipes différentes, puis être rassemblés sans perte d'information notable.

- *Esthétique* ("Visual appeal") : une notation doit être agréable à utiliser visuellement. De ce point de vue, les notations graphiques sont souvent

plus agréables que les notations purement textuelles. En outre, l'esthétique d'une notation participe à sa lisibilité.

- *Apprentissage progressif* ("Incremental acquisition") : une notation doit pouvoir s'apprendre aisément et progressivement comme un langage de programmation. En corollaire, la disponibilité d'une documentation de référence la concernant est nécessaire. On peut regretter que la rédaction d'une telle documentation soit aujourd'hui souvent négligée par les auteurs. Elle est pourtant l'une des conditions nécessaires à l'utilisation des formalismes en milieu industriel.

+ *Existence d'outils d'aide à la spécification* ("Tools support") : l'existence d'un environnement informatisé d'aide au concepteur devient nécessaire lorsque l'interface spécifiée sort des exemples d'école peu volumineux. Ces outils peuvent prendre la forme d'éditeurs, de vérificateurs de syntaxe, de générateurs automatiques de code, ou même d'évaluateurs automatiques de l'utilisabilité de l'interface spécifiée.

- *Capacité à faciliter la détection des erreurs* ("Support for error identification") : tout concepteur commet des erreurs, et leur détection lors des premières phases de conception permet des économies substantielles. En effet, le coût de leur correction est alors peu élevé comme l'illustre la figure 3 du chapitre d'introduction de ce mémoire. Ce principe est à rapprocher d'une remarque issue du forum électronique "comp.human-factors" déjà évoqué dans l'introduction de ce chapitre : c'est dans la facilité de détection des problèmes que l'on peut distinguer les notations les unes des autres ("detectable problems distinguish a notation from another").

- *Sémantique claire* ("Clear semantics") : il est assez évident qu'une notation doit éviter les ambiguïtés et les difficultés d'interprétation. Ce critère est particulièrement important lorsque la notation est utilisée par plusieurs équipes.

- *Étude de cas* ("Case study") : une notation doit comporter une série d'exemples de spécification portant sur des cas d'études classiques. Ainsi il est possible de la comparer, selon une approche pratique, avec d'autres notations.

+ *Finalité clairement définie* ("Clear scoping") : les buts, avantages, et limitations d'une notation doivent être exprimés clairement. Aucune

notation n'est parfaite ni ne convient à tous. Cependant, l'utilisateur potentiel doit pouvoir identifier aisément les notations qui conviennent à ses besoins.

- *Adéquation à la tâche du concepteur* ("Task fit") : la distance entre la notation utilisée et les concepts du domaine ne doit pas être trop importante. Dans le cas contraire, les avantages de la spécification seront minimisés par le surcoût de travail dû aux nécessaires transformations.

- *Utilité sociale* ("Social value") : l'utilisation des notations doit être replacée dans le contexte industriel de la conception d'interfaces en équipe. En effet, la majorité des utilisateurs actuels de ces notations sont des scientifiques motivés, et certaines données ignorées jusqu'à présent doivent maintenant être prises en compte. Dans le contexte industriel, la notation doit aider l'équipe de conception à atteindre des objectifs communs. Citons par exemple le maintien de la connaissance lors de l'arrivée d'une personne dans une équipe, ou lors de son départ.

Comparés aux différents types de classifications précédemment évoqués, les Principes de Namur s'intéressent tous, au moins en partie, à l'utilisabilité des formalismes. Quelques uns (indiquées par le symbole -) s'y intéressent exclusivement. D'autres prennent également en considération la finalité (+) ou le pouvoir d'expression (.) des formalismes, mais ils n'évoquent jamais leur origine. Cet absence n'est pas surprenante, car les Principes de Namur n'ont pas pour objet la constitution de classifications, mais plutôt la constitution d'un cahier des charges auquel les notations doivent adhérer.

Les auteurs des Principes de Namur ont défini, pour chacun des onze principes retenus, une méthode d'évaluation. Notons que ces principes sont destinés non seulement à l'évaluation des formalismes existants, mais aussi à l'évaluation des formalismes à venir. Nous utiliserons d'ailleurs les Principes de Namur pour évaluer les extensions aux formalismes que nous proposons au chapitre IV de ce mémoire.

Ces principes ont pour but de nous aider dans le choix d'un formalisme adapté à nos besoins. Remarquons que certains des Principes de Namur comme "l'étude de cas" ont pour intention première non pas exactement l'évaluation de l'utilisabilité des notations, mais plutôt l'évaluation de l'utilisabilité de leur évaluation. Cette nouvelle approche est très intéressante, elle met en lumière la nécessité impérative de pouvoir choisir aisément une

notation en fonction des besoins. Nous allons maintenant nous intéresser aux moyens de ce choix. Ils font l'objet de la section suivante.

3. Choix d'un formalisme

Nous venons de distinguer quatre principaux types de classifications destinés aux formalismes : selon leur pouvoir d'expression, selon leur utilisabilité, selon leur origine, et selon leur finalité. En outre, nous avons montré que les Principes de Namur reprennent certains des critères évoquées afin de nous offrir une sorte de cahier des charges du "bon" formalisme. Cependant, dans le but de choisir un formalisme, aucune de ces classifications ne peut être écartée ou favorisée de manière justifiée. De même, les Principes ne permettent pas à eux seuls la sélection d'un formalisme. Sélection d'autant plus difficile que, de l'aveu même de leurs auteurs, ces classifications, tout comme les Principes, engendrent des critères non orthogonaux.

Cependant, ces classifications et les Principes de Namur doivent pouvoir aider le concepteur à choisir un formalisme adapté à ses besoins. Plutôt que de choisir arbitrairement l'une des classifications identifiées section 2, nous proposons de les utiliser de manière complémentaire. L'usage simultané de plusieurs classifications risque cependant de provoquer une surcharge d'informations dont la synthèse, le choix d'un formalisme, est difficile à réaliser. C'est pourquoi, afin de structurer la démarche de choix d'un formalisme, nous suggérons un ordre concernant usage de ces classifications.

En premier lieu, il convient avant tout de définir le problème posé, c'est l'objet du paragraphe 3.1. Il a pour but la définition précise des besoins du concepteur. Nous illustrons ce paragraphe par l'exemple de nos propres besoins. Les quatre paragraphes suivants de cette section (§ 3.2-3.5) décrivent et justifient l'ordre d'usage des classifications proposé. Ils donnent également en regard l'exemple des choix faits dans le cadre de ce mémoire. Nous n'avons pas utilisé de manière explicite les Principes de Namur car, comme le souligne le paragraphe 2.5, ils sont inclus dans trois des quatre types de classifications déjà évoqués.

3.1. Problème posé

Ce paragraphe a pour objet de définir les requis en terme de pouvoir d'expression, d'utilisabilité et de finalité d'un formalisme adapté à nos

besoins. Ces besoins sont exprimés de manière volontairement informelle, comme pourrait le faire une équipe de conception. L'affinement de ces besoins s'effectuera au fur et à mesure de l'utilisation des classifications. Dans un premier temps, ce sont les éléments relatifs au pouvoir d'expression qui sont généralement exprimés par les concepteurs. Puis, le rôle du formalisme au sein de l'équipe et des phases de conception est défini, il a alors trait plus particulièrement à la finalité et à l'utilisabilité des formalismes.

Dans le cadre de ce mémoire, nous désirons disposer d'un formalisme capable de spécifier l'interface d'un système à la fois critique et temps-réel. Dans de tels systèmes, les interruptions sont nombreuses et leurs conséquences peuvent être graves. C'est pourquoi elles doivent être prises en considération dès les premières étapes du processus de conception. De même, les corrections d'erreurs aussi bien en provenance du système que de l'opérateur doivent pouvoir être spécifiées.

Comme l'illustre la figure III-5 ci-contre, le formalisme utilisé doit permettre la communication entre les différents intervenants en charge de chacune des phases de conception : analyse de tâche, description de l'interface homme-machine, et spécifications logicielles. Ceci dans le but de garder la mémoire des choix effectués aux premières phases de conception, jusqu'à la réalisation finale. A priori rien n'impose que les intervenants soient des personnes différentes. Cependant cette éventualité est probable car, comme l'illustre la figure III-5, chaque phase de conception correspond à une spécialité distincte, donc à un praticien distinct. Il convient maintenant de choisir le ou les formalismes adéquats.

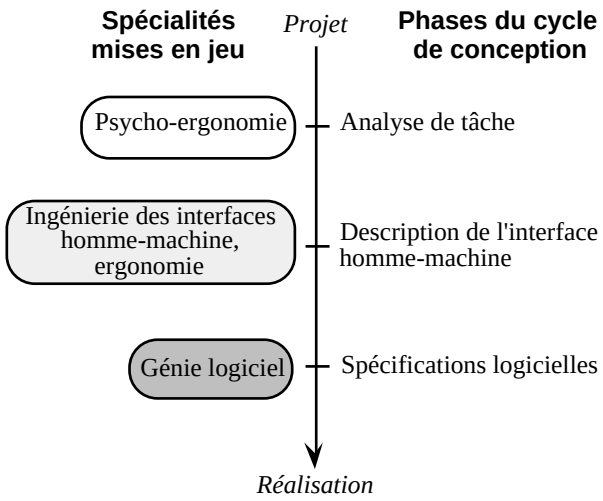

Figure III-5 : Phases de conception et spécialités mises en jeu lors de la conception d'une interface homme-machine.

3.2. Choix selon la finalité

¶ *En premier lieu, la recherche d'un formalisme doit débuter par une définition précise des besoins vis-à-vis de la démarche de conception*

envisagée. Identifiés, ces besoins définiront ainsi les finalités du formalisme adéquat. Un ensemble de formalismes répondants aux besoins peut alors être extrait par l'usage d'une classification basée sur la finalité. La classification proposée par Balbo [Balbo 1994] peut convenir à cette première tâche.

Notre besoin est de disposer d'un formalisme capable de couvrir trois étapes du cycle de conception d'une interface homme-machine : l'analyse de tâche, la description de l'interface homme-machine, et les spécifications logicielles. Ces étapes coïncident avec trois des finalités définies par Balbo : analyse des besoins, conception de l'interface, et conception logicielle. Nous nous servirons donc de sa classification pour ce premier choix.

Ces trois finalités identifiées, nous nous trouvons maintenant devant plusieurs alternatives : utiliser un seul formalisme pour couvrir toutes les finalités, ou bien plusieurs. Un examen rapide de la classification de Balbo, figure III-4, montre que peu de formalismes conviennent à toutes les finalités. Ce sont essentiellement ADEPT [Johnson & Johnson 1993], MUSE [Lim & Long 1994] basée sur la méthode JSD [Jackson 1983], auxquels nous ajouterons Mastermind [Szekely, Sukaviriya, et al. 1996]. Ce dernier étant récent, il n'est cité dans aucune des classifications précédemment étudiées, c'est pourquoi nous le détaillons ci-après.

Mastermind est basé sur les idées de Humanoid [Szekely, Luo, & Neches 1993] et UIDE [Frank & Foley 1993], et n'est encore qu'à l'état de démonstrateur au moment où ces lignes sont écrites. Mastermind permet de générer complètement l'interface d'une application informatique à l'aide de nombreux outils de conception interconnectés. Cependant, Mastermind doit plus être considéré comme un générateur d'interface extrêmement évolué, plutôt que comme un outil basé sur des concepts de l'ergonomie cognitive. En effet, Mastermind ne fournit aucune aide, autre que des outils de dessin graphique, à l'analyse de tâche qui doit être faite manuellement par le concepteur.

L'évaluation des formalismes à large spectre, c'est-à-dire couvrant de nombreuses étapes du cycle de conception comme MUSE/JSD*, ADEPT, ou encore Mastermind pose un épineux problème. Ces formalismes, qui devraient plutôt être appelés multi-formalismes, font appel à plusieurs types de formalismes internes. Doit-on les rejeter en bloc dès la découverte d'une limitation de l'un de leurs formalismes internes, où doit-on les considérer

comme un ensemble de formalismes indépendants aux qualités distinctes ? Cette seconde solution nous semble préférable, cependant la classification de Balbo ne permet pas de distinguer les différentes composantes de ces formalismes à large spectre. Nous sommes alors tenu de considérer ces multi-formalismes comme un tout. C'est là une limitation problématique de la classification.

Nous allons maintenant procéder au recensement des formalismes convenant aux finalités identifiées. Notons que nous avons éliminé d'office Mastermind car il souffre d'un manque critique de documentation, le rendant inexploitable. De plus, il n'est à ce jour qu'à l'état de démonstrateur.

3.2.a. Analyse de tâche

Selon la classification de Balbo, les formalismes adaptés à l'analyse de tâche sont UAN, MUSE/JSD*, MAD, CLG, GOMS, ETIT, et ADEPT.

3.2.b. Description de l'interface homme-machine

Encore selon la classification de Balbo, les formalismes adaptés à la description de l'interface homme-machine sont ETAG, UAN, DIANE/MERISE, MUSE/JSD*, MAD, CLG, et ADEPT.

3.2.c. Spécifications logicielles

Toujours selon la classification de Balbo, les formalismes adaptés aux spécifications logicielles sont SIROCO, DIANE/MERISE, et ADEPT auxquels nous avons ajouté MUSE/JSD*.

3.3. Choix selon l'origine

• *Le premier tri basé sur la finalité effectué, il peut s'avérer judicieux de sélectionner l'origine du formalisme recherché selon ses qualités attendues. Ici la classification proposée par Brun et Beaudoin-Lafon [Brun et al. 1995] convient parfaitement.*

Nous nous intéressons maintenant à la sélection des formalismes convenant à notre problème.

3.3.a. Analyse de tâche

Le formalisme support à l'analyse de tâche doit avant tout intégrer la notion de tâche. Selon Brun et Beaudoin-Lafon, seuls les formalismes issus des sciences cognitives conviennent ici, c'est-à-dire UAN, XUAN, Keystroke, GOMS, CLG, et MAD. Balbo s'intéresse aussi à l'élément directeur des

formalismes et différencie ceux basés sur des concepts, et ceux basés sur la notion de tâche. Sa classification permet de compléter celle de Brun et Beaudoin-Lafon par DIANE/MERISE, MUSE/JSD*, ETIT, et ADEPT.

Seule DIANE/MERISE peut être écartée ici, car elle ne convient pas à l'analyse de tâche. Ce tri est limité car selon la classification de Balbo tous les formalismes adaptés à l'analyse des besoins, à l'exception de DIANE/MERISE, ont la notion de tâche comme élément directeur ou ne sont pas cités. Ce n'est pas surprenant car la notion de tâche est très liée à l'analyse des besoins. Remarquons ici un exemple de la non-orthogonalité des critères utilisés par les classifications.

Remarquons également la complémentarité des classifications, complémentarité synonyme ici de disjonction car plutôt que de restreindre les choix possibles, l'usage des deux classifications augmente le nombre de formalismes candidats.

3.3.b. Description de l'interface homme-machine

Le formalisme support à la description de l'interface homme-machine doit obligatoirement intégrer les retours d'information en provenance de l'interface. Il est difficile de conclure ici, car la notion de retour d'information n'est pas discriminante vis-à-vis d'une origine. Certains des formalismes issus des sciences cognitives et de la théorie du calcul conviennent.

3.3.c. Spécifications logicielles

Le formalisme support aux spécifications logicielles doit permettre d'établir des preuves sur les caractéristiques du système, et doit prendre en compte finement le séquencement des actions et les contraintes temporelles. Nous nous tournons donc vers les formalismes issus de la théorie du calcul qui offrent tous ces possibilités. Selon Brun et Beaudoin-Lafon, les formalismes candidats sont les Réseaux de Petri orientés objet ou non, CSP, Squeak, et full LOTOS.

Les formalismes déjà sélectionnés précédemment, SIROCO et ADEPT, ne sont pas évalués dans la classification de Brun et Beaudoin-Lafon, tout comme les formalismes cités dans cette dernière classification ne sont pas évalués par Balbo. Nous nous trouvons devant deux classifications disjointes. Aucun tri n'est possible, il faut donc se contenter de fusionner les deux

sélections. Ce qui donne : SIROCO, ADEPT, DIANE/MERISE, MUSE/JSD*, les réseaux de Petri orientés objet ou non, CSP, Squeak, et full LOTOS.

3.4. Choix selon le pouvoir d'expression

, *Il est maintenant nécessaire de vérifier que le pouvoir d'expression des formalismes sélectionnés convient aux besoins exprimés. De nombreuses classifications et évaluations s'intéressent au pouvoir d'expression des formalismes. L'usage d'une classification plutôt qu'une autre peut être guidé par les besoins spécifiques exprimés, comme par exemple l'expression du parallélisme. Cependant, l'usage de plusieurs classifications complémentaires peut également s'avérer nécessaire. A ce moment, une évaluation pratique de la sélection peut être envisagée.*

En ce qui nous concerne, tous les formalismes dont nous avons besoin doivent prendre en compte les interruptions et les erreurs. La notion d'interruption ne fait pas partie des critères sélectionnés ni par Brun et Beaudoin-Lafon, ni par Balbo. Et selon Brun et Beaudoin-Lafon aucun formalisme ne traite les erreurs. Cette affirmation est cependant à nuancer. En effet aucun formalisme ne traite les erreurs, mais la notion de comportement erroné est présente dans un formalisme pour la représentation des scénarii dans les didacticiels [Bordon 1994], qui est à rapprocher des conditions anormales des blocs de connaissance de Boy [Boy 1989] que nous évoquerons au paragraphe 3.5. Ces deux formalismes n'ont été étudiés par aucun des auteurs précités, probablement parce qu'ils n'ont pas comme finalité première la spécification des interfaces.

Le critère du pouvoir d'expression des interruptions et des erreurs ne pouvant être utilisé faute de classification, nous nous tournons vers d'autres critères de choix.

3.4.a. Analyse de tâche

Le formalisme support à l'analyse de tâche doit avant tout disposer d'une méthode facilitant cette analyse. En conséquence, un affinement en tâche et sous-tâche doit être possible avec le formalisme utilisé. Selon la classification de Balbo, les formalismes précédemment candidats et ne permettant pas le raffinement en tâche et sous-tâche sont CLG, GOMS et ETIT. Il nous reste donc après ce tri UAN, MUSE/JSD*, MAD, et ADEPT. A notre connaissance seuls UAN et MAD permettent de traiter les interruptions, c'est pourquoi nous les avons retenus. En outre, seul MAD dispose d'une méthode rigoureuse

permettant l'analyse de tâche. Nous avons donc choisi MAD pour l'analyse des besoins.

3.4.b. Description de l'interface homme-machine

Le formalisme support à la description de l'interface homme-machine doit permettre l'expression des retours d'information en provenance de l'interface. Selon Balbo, les formalismes UAN, DIANE/MERISE, MUSE/JSD*, CLG, et SIROCO conviennent. Selon Brun et Beaudoin-Lafon UAN, XUAN, Keystroke, réseaux de Petri conviennent également, avec des résultats en faveur de UAN. Notons que CLG est jugé par Balbo apte et inapte par Brun et Beaudoin-Lafon. Nous remarquons ici un exemple de critère subjectif des évaluations en fonction des auteurs. UAN [Hartson et al. 1990] semble ici le formalisme le plus apte à la spécification de l'interface, ce formalisme et son extension XUAN [Gray, England, & McGowan 1994] sont en effet cités dans de nombreuses études dont celle de Harrison et Duke [Harrison et al. 1994]. De plus, UAN est le seul permettant d'exprimer explicitement les interruptions, c'est pourquoi nous l'avons retenu.

3.4.c. Spécifications logicielles

Tous les formalismes de notre sélection permettent au concepteur de décrire les contraintes temporelles de l'interface car nous les avons choisis selon leur origine. Ces formalismes doivent également pouvoir exprimer le parallélisme. Selon la classification de Brun et Beaudoin-Lafon tous conviennent. Nous nous tournons donc vers d'autres critères pour affiner notre choix. La capacité des réseaux de Petri d'exprimer en partie les retours d'informations de l'interface nous les a fait choisir. En outre, une description UAN peut être traduite selon les réseaux de Petri, ce qui est un point intéressant [Palanque, Bastide, & Senges 1995b].

3.5. Choix selon l'utilisabilité

„ *Afin que le pouvoir d'expression puisse réellement être utilisé, l'utilisabilité du formalisme doit convenir à ses futurs utilisateurs. Compte tenu de l'aspect subjectif de l'étude de l'utilisabilité, il semble qu'une évaluation pratique sur un exemple représentatif, est une solution préférable à l'usage d'une classification.*

Il convient maintenant d'évaluer l'utilisabilité des formalismes que nous avons choisis. Nous nous intéressons donc à la facilité avec laquelle les interruptions sont décrites dans MAD [Scapin et al. 1989], UAN [Hartson et al.

1990] et les réseaux de Petri [Peterson 1981], que nous avons sélectionnés. Nous avons ajouté à notre étude le formalisme des Blocs de Connaissance qui est l'un des rares formalismes à avoir pris en compte les interruptions, sous la forme des "conditions anormales", dès sa création [Boy 1989]. Nous illustrons notre analyse au moyen d'un exemple simple mais réel : le Publiphone, c'est-à-dire le téléphone public à carte de France Télécom™.

Notons que cette étude pratique évalue non seulement l'utilisabilité des formalismes, mais aussi leur pouvoir d'expression. En effet, il est difficile de les évaluer indépendamment. Cette étude combinée permet en conséquence d'apporter un complément d'évaluation au choix n°, effectué précédemment selon le pouvoir d'expression des formalismes (§ 3.3).

3.5.a. Scénario : le Publiphone

Le scénario que nous avons défini comporte deux interruptions. La première est causée par l'utilisateur qui commet une erreur en composant le numéro de son correspondant. La seconde est du fait de la machine lorsqu'en cours de communication, la Télécarte de l'utilisateur arrive en fin de crédit.

Décrivons ce scénario plus précisément : Après avoir décroché le combiné, l'utilisateur enfiche sa Télécarte puis compose le numéro de son correspondant. Au cours de cette tâche, il commet une erreur et s'en aperçoit. Il raccroche puis décroche à nouveau le combiné et, sans erreur cette fois-ci, compose une nouvelle fois le numéro de son correspondant. Au cours de la communication, le Publiphone signale à l'utilisateur que sa Télécarte arrive en fin de crédit, et qu'il doit en changer. L'utilisateur retire la Télécarte vide, puis insère une nouvelle Télécarte tout en continuant la conversation. À la fin de la communication il raccroche le combiné puis retire sa Télécarte, ou bien effectue l'inverse.

Ce scénario, bien que fondé sur l'expérience, limite volontairement les possibilités d'interaction avec un Publiphone afin de centrer la formalisation de l'interface sur la modélisation des interruptions. Le scénario retenu couvre deux sources d'interruptions : Les interruptions provoquées par le système (la Télécarte est vide) et celles décidées par l'utilisateur (qui s'aperçoit d'une erreur de saisie de numéro). Les cas où l'utilisateur interrompt le système et où le système s'interrompt lui-même peuvent être traités de manière symétrique.

Ayant présenté le scénario, nous allons successivement étudier les réponses de MAD, des Blocs de Connaissance, de UAN, et des réseaux de Petri.

3.5.b. Méthode Analytique de Description (MAD)

MAD [Scapin et al. 1989] répond à une méthodologie rigoureuse fondée sur une approche psycho-ergonomique [Sebilotte 1991]. Comme le montre la figure III-6, ce formalisme s'appuie sur une décomposition hiérarchique des tâches à l'aide de quatre constructeurs :

- SEQ (séquence) exprime l'enchaînement séquentiel de plusieurs tâches
- ALT (alternative) traduit la possibilité de choix entre plusieurs tâches
- PAR (parallélisme) désigne l'exécution entrelacée de plusieurs tâches par un même agent
- SIM (simultanéité) dénote l'exécution parallèle de plusieurs tâches par des agents distincts

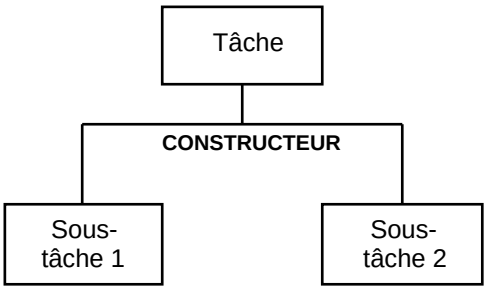

Figure III-6 : Structure hiérarchique des tâches MAD.

Dans une note récente, et pour répondre à des problèmes d'interprétation et d'utilisation de MAD, les auteurs clarifient leur formalisme et introduisent la notion d'interruption [Sebillotte, Alonso, et al. 1994].

Dans MAD, la description d'une tâche comprend :

- Une identification de tâche (numéro, nom de la tâche)
- Des éléments structurants (but, état initial, préconditions, corps de la tâche, postconditions, état final)
- Des attributs : tâche facultative (FAC), boucle (@), tâche prioritaire (PRIOR) et/ou interruptible (INTER), un niveau de priorité

La figure III-7 retranscrit le résultat de la modélisation de notre scénario au moyen de MAD. On constate d'emblée que la représentation graphique d'un scénario simple comme le Publiphone nécessite une page entière de description. Nous verrons que les formalismes évoqués dans les paragraphes suivants sont plus compacts. Mais en contrepartie, MAD est très aisément lisible et une formalisation MAD ne demande que peu d'effort d'interprétation.

Dans notre description du scénario, une interruption se traduit directement au moyen du constructeur PAR: La tâche “Composer Nº”, décorée de l'attribut

INTER, peut être interrompue par la tâche “Reprise erreur Nº” dès que la précondition “Erreur_Nº” est satisfaite. En l'absence de l'attribut INTER, la tâche interrompante s'exécute jusqu'au bout. Alors “Composer Nº” peut reprendre et, d'après la spécification, la reprise se fera en début de tâche. Si cette expression traduit correctement le scénario, il convient néanmoins de s'interroger sur le cas où plusieurs tâches pourraient interrompre simultanément “Composer Nº” (leurs préconditions d'activation deviendraient *vraies* en même temps). La notion de priorité répond au problème du choix d'exécution parmi les tâches activables, mais conduit à des situations de famine de tâche (une tâche activable moins prioritaire perdrait en permanence son tour d'exécution). La sémantique du formalisme MAD n'offre pas de réponse sur ce point. De même, si plusieurs tâches activables sont de même priorité, on ne peut déterminer laquelle sera choisie.

La seconde interruption du scénario (Télécarte en fin de crédit) n'est pas représentable correctement dans le formalisme MAD : L'opérateur PAR suppose une exécution entrelacée entre les tâches qu'il relie. Dans notre cas, il s'agit d'un parallélisme vrai : La conversation peut continuer pendant le changement de carte. Le constructeur SIM conviendrait à la situation mais suppose que les tâches soient exécutées par des agents distincts. MAD rejette explicitement le parallélisme vrai pour un agent donné. Or un utilisateur habile (au sens de Rasmussen [Rasmussen 1986]), et notamment l'habitué du Publiphone, est capable d'exécuter plusieurs tâches simultanément, et peut donc continuer la conversation pendant le changement de carte.

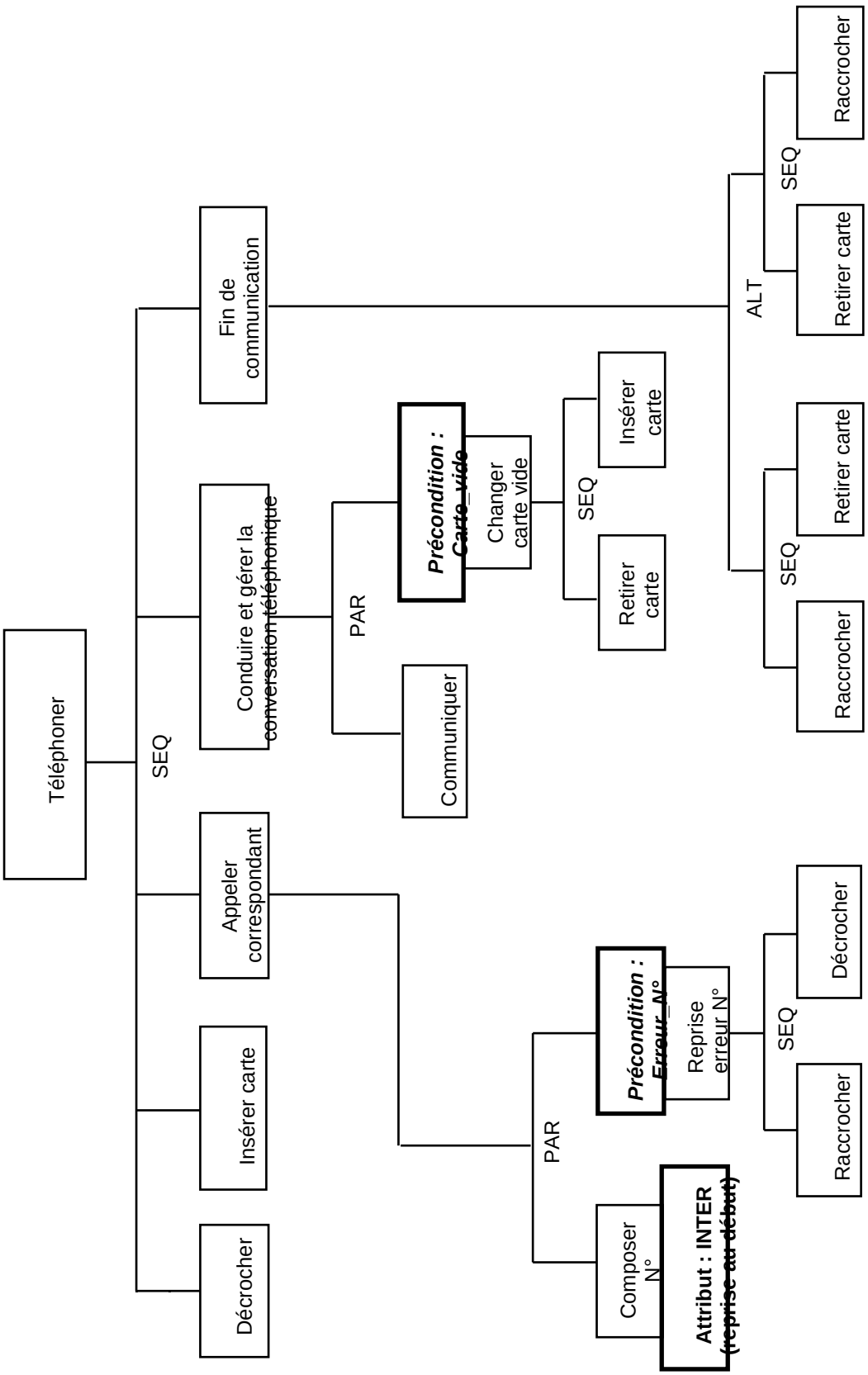

Figure III-7 : Description de la tâche téléphoner à l'aide de MAD.

Une seconde limitation de la couverture d'expression de MAD est l'absence de contraintes temporelles. Dans l'exemple du scénario, les tâches séquentielles “Retirer carte” et “Insérer carte” qui composent la tâche “Changer carte vide” doivent être exécutées dans les limites d'un délai de quelques secondes avant coupure de la ligne. Dans le cas d'un Publiphone, le non respect de cette contrainte par l'utilisateur ne devrait pas mettre en jeu la sécurité. Il n'en va pas de même dans les systèmes critiques où l'expression des contraintes temporelles doit être explicite.

MAD ne couvre pas la description des retours d'information en provenance du système. Cette propriété est cohérente avec les intentions initiales des auteurs qui cherchaient en MAD un outil de conception. Cependant par affinements successifs, la conception doit se préoccuper du comportement perceptible du système, par exemple pour prévenir l'utilisateur du Publiphone que sa carte est vide. Dans notre cas, le mode de notification des conditions anormales a un pouvoir déterminant sur le caractère disruptif du système.

En résumé, dans sa forme graphique, MAD est peu compacte, mais lisible, et reconnaît le concept d'interruption. Nous relevons toutefois un vide sémantique concernant l'exécution des tâches d'interruption. Certes, l’ambiguïté vaut pour des cas extrêmes mais néanmoins possibles dans les systèmes critiques. En outre, MAD n'autorise pas l'expression du parallélisme vrai pour un agent donné, ni ne permet l'expression des retours d'information du système.

3.5.c. Les Blocs de Connaissance (KB)

Les Blocs de Connaissance sont issus de recherches en représentation des connaissances [Boy 1989]. Ils ont été formalisés et appliqués à un système de télémanipulation spatiale [Mathé 1990], puis enrichis d'une capacité de parallélisme et utilisés pour la mise en œuvre d'un système d'aide à la conduite automobile [Pleczon 1992].

Les blocs représentent la connaissance opérationnelle de l'utilisateur d'un système sous forme de tâches. Comme le montre la figure III-8, un bloc comprend cinq parties : des *préconditions*, une *procédure*, un *contexte* ou conditions contextuelles, un *but* et des *conditions anormales*. Ces dernières rendent explicites les cas d'interruptions et d'erreurs. Les flèches renvoient ou proviennent d'autres blocs. Contrairement aux modèles de tâche usuels, la structure des tâches est un graphe orienté.

Le mécanisme d'exécution des blocs fait appel à un moteur d'inférence qui évalue les conditions des blocs focalisés, puis si elles sont vérifiées, exécute les procédures de ces blocs. Une procédure contient des actions élémentaires ou d'autres blocs. Ces derniers sont focalisés lorsque la procédure du bloc englobant est exécutée. Les préconditions d'un bloc déterminent l'exécution de la procédure de ce bloc. Les conditions anormales et le but d'un bloc déterminent la focalisation des blocs référencés respectivement dans ces conditions et dans ce but.

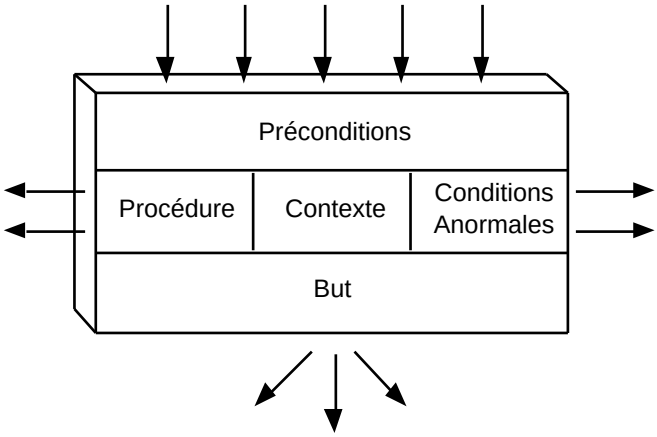

Figure III-8 : Structure d'un Bloc de Connaissance.

Pour traiter l'exemple du scénario, nous nous sommes appuyés sur la dernière version des Blocs de Connaissance qui inclut l'expression du parallélisme [Pleczon 1992]. Comme le montre la figure III-9, le formalisme des Blocs de Connaissance décrit de manière très efficace l'interruption due à l'erreur de saisie du numéro : cette erreur est une condition anormale et le formalisme offre un statut explicite aux situations anormales. La condition anormale “Erreur_Nº” devenant vraie, la procédure du bloc “Composer Nº” est interrompue et le bloc “Raccrocher-Décrocher” est exécuté suivi d'un retour au début du bloc “Composer Nº”. L'interruption système due à la carte vide est modélisée comme une précondition : elle perd son statut d'exception.

Comme MAD, les Blocs de Connaissance ne permettent pas de décrire les retours d'information issus du système. On ignore donc comment l'utilisateur s'aperçoit que sa carte est vide et qu'il doit la changer. Si, contrairement à MAD, les blocs admettent le parallélisme entre plusieurs tâches exécutées par un même agent, ils ne couvrent pas l'expression des contraintes temporelles. Enfin, une interruption qui doit être traitée en parallèle avec la tâche en cours

perd son statut de condition anormale. UAN répond partiellement à ces limitations.

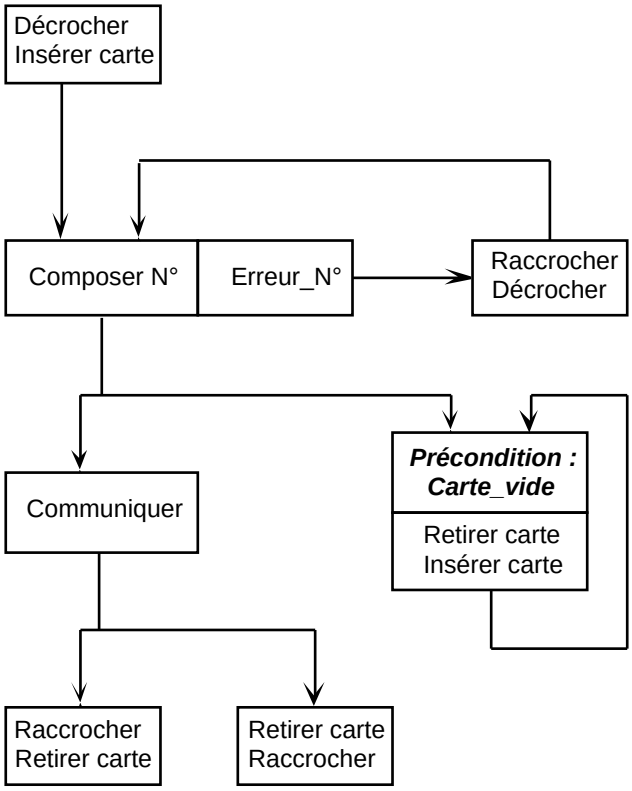

Figure III-9 : Description de la tâche téléphoner à l'aide des Blocs de Connaissance.

3.5.d. User Action Notation (UAN)

A l'origine, UAN a été conçu pour servir de support à l'expression des spécifications des interfaces à manipulation directe. À l'aide de UAN, le concepteur spécifie les actions de l'utilisateur sur les dispositifs physiques, d'où son nom "User Action Notation", avec en regard les réactions du système et ses changements d'état interne pertinents. La figure III-10 illustre le format tabulaire d'une description de tâche élémentaire en UAN. La notation s'est ensuite enrichie d'opérateurs de composition et de relations temporelles, dans le but de modéliser les tâches complexes qu'un utilisateur peut accomplir avec un système. Parmi ces opérateurs, nous relevons :

- Séquence (A.B ou A *placé au-dessus de* B) signifie que les tâches A et B seront exécutées dans l'ordre A puis B
- Choix (A | B) indique que l'utilisateur peut accomplir la tâche A ou la tâche B
- Répétition (A*) dénote l'accomplissement multiple de la tâche A

- Attente (t>x secondes ou t<y secondes) signifie que l'utilisateur doit attendre x secondes, ou au contraire ne doit pas dépasser y secondes avant d'exécuter la tâche suivante
- Entrelacement bidirectionnel (A<->B) exprime que les tâches A et B s'effectuent durant le même intervalle de temps, mais une seule des tâches est active à la fois
- Entrelacement monodirectionnel (A->B) signifie que la tâche A peut interrompre la tâche B puis s'exécuter jusqu'à sa fin, et non l'inverse
- Parallélisme (A || B) dénote l'exécution simultanée des tâches A et B
- Ordre indifférent (A&B) : l'utilisateur peut accomplir A puis B ou B puis A

Tâche : ABCD		
Action utilisateur	Retour d'information de l'interface	État interne
Action 1	Retour 1	État 1
Action 2	Retour 2	État 2
Action 3	Retour 3	État 3

Figure III-10: Structure d'une tâche UAN.

Au sein de notre spécification, nous n'avons pas toujours utilisé les trois colonnes du format UAN. Nous ne les avons explicitées que lorsqu'elles permettaient d'alimenter la discussion. Les quatre tableaux de la figure III-11 répondent à l'expression du scénario. Nous remarquons la concision de la notation, à peine plus gourmande en espace que les Blocs de Connaissance, mais qui en contrepartie rend explicites les retours d'information du système.

Le tableau en haut à gauche figure III-11 montre la structure de la tâche de plus haut niveau "Téléphoner" : en séquence, l'utilisateur exécute les tâches "Décrocher" et "Insérer Carte" puis saisit le numéro du correspondant avant d'entrer en communication. Tout en maintenant la conversation avec son interlocuteur (opérateur ||), l'utilisateur peut changer de Télécarte autant de fois qu'il le faut (symbole *). Puis il retire sa carte et raccroche le combiné, dans l'ordre qui lui convient (opérateur &). L'expression du scénario est directe et simple mais le statut de condition anormale du changement de carte vide ou d'erreur de saisie reste implicite.

Tâche : Téléphoner
Décrocher
Insérer carte
Composer N°
(Communiquer **||** (Changer carte vide)*)
(Raccrocher & Retirer carte)

Tâche : Changer carte vide	
Action Utilisateur	Retour d'information
	Affiche (RETIRER CARTE VIDE)
Retirer carte	Affiche (NOUVELLE CARTE)
(t < 10 secondes)	
Insérer carte	

Tâche : Composer N°
(Taper touche clavier | Reprise erreur N°)*
Taper touche clavier
Taper touche clavier

Tâche : Reprise erreur N°	
Action Utilisateur	Retour d'information
Raccrocher	Affiche (DÉCROCHER)
(t < 2 secondes)	
Décrocher	

Figure III-11 : Description de la tâche téléphoner à l'aide de UAN.

Le tableau en bas à gauche figure III-11 démontre l'inadéquation des opérateurs d'entrelacement, tel <->, cité plus haut, pour exprimer la terminaison brutale d'une tâche par une autre. Les opérateurs d'entrelacement de UAN expriment l'interruptibilité mais supposent une reprise de la tâche interrompue, sans pour autant préciser le lieu de reprise. Dans notre cas, il s'agit de tuer la tâche de saisie afin de l'exécuter de nouveau en son début. Dans UAN, il n'existe pas d'opérateur "tuer tâche". C'est pourquoi, afin de décrire l'erreur de saisie, nous avons dû modéliser la tâche au niveau des actions physiques, et spécifier que la tâche de saisie ne se termine que si au moins un numéro à deux chiffres a été composé[15]. Cette description est lourde et peu lisible, et l'opérateur | ne rend pas compte du caractère disruptif de l'erreur de saisie.

Les tâches "Changer carte vide" et "Reprise erreur N°" reproduites au sein des deux tableaux de droite figure III-11 démontrent les capacités de UAN à exprimer les contraintes temporelles d'exécution entre deux tâches séquentielles. Par exemple, dans la tâche de reprise sur erreur de numérotation, il convient pour l'utilisateur de raccrocher le combiné et de ne

[15] En France, les numéros de téléphones du réseau public ont au moins deux chiffres (17, 18, etc.).

pas attendre plus de deux secondes avant de le décrocher à nouveau. Le facteur temps, qui joue un rôle central dans les systèmes critiques, est donc explicité dans UAN. En revanche, UAN n'offre pas de matériau formel dédié à l'expression des anomalies.

3.5.e. Les réseaux de Petri

Les réseaux de Petri [Peterson 1981] sont un formalisme graphique (Cf. figure III-12) basé sur de solides outils mathématiques. Ils sont avant tout dédiés à la modélisation des systèmes à événements discrets. Selon Palanque et Bastide [Palanque & Bastide 1995a], ‹‹ un réseau de Petri modélise un système par un ensemble de variables d'état (appelées *places* et représentées graphiquement par des ellipses) et par un ensemble d'opérateurs de changement d'état (appelés *transitions* et représentées par des rectangles). L'état d'un système est modélisé à tout instant par une distribution de *jetons* dans les places du réseau. A un instant donné, plusieurs places du réseau peuvent être *marquées* par un nombre quelconque de jetons. Les places et les transitions sont reliées par des *arcs d'entrée* qui définissent les conditions pour qu'un opérateur de changement d'état soit autorisé. Les arcs d'entrée et les *arcs de sortie* (reliant une transition à une place) spécifient l'effet de l'opérateur de changement d'état sur le marquage du réseau. On appelle *occurrence* d'une transition l'activation d'un tel opérateur de changement d'état. Une transition est *franchissable* si chacune de ses places d'entrée contient au moins un jeton. L'occurrence d'une transition retire un jeton de chacune des places d'entrée et dépose un jeton dans chacune des places de sortie. De plus, un *poids* (qui est un entier n) est associé à chacun des arcs permettant de prendre ou de déposer n jetons à la fois. De manière générale, une transition peut consommer ou produire des jetons : le nombre de jetons pris dans les places d'entrée peut être différent du nombre de jetons déposé dans les places de sortie. ››

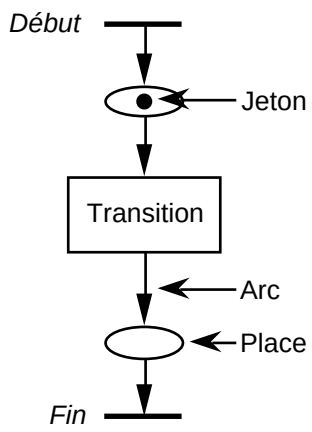

Figure III-12 : Structure d'un réseau de Petri.

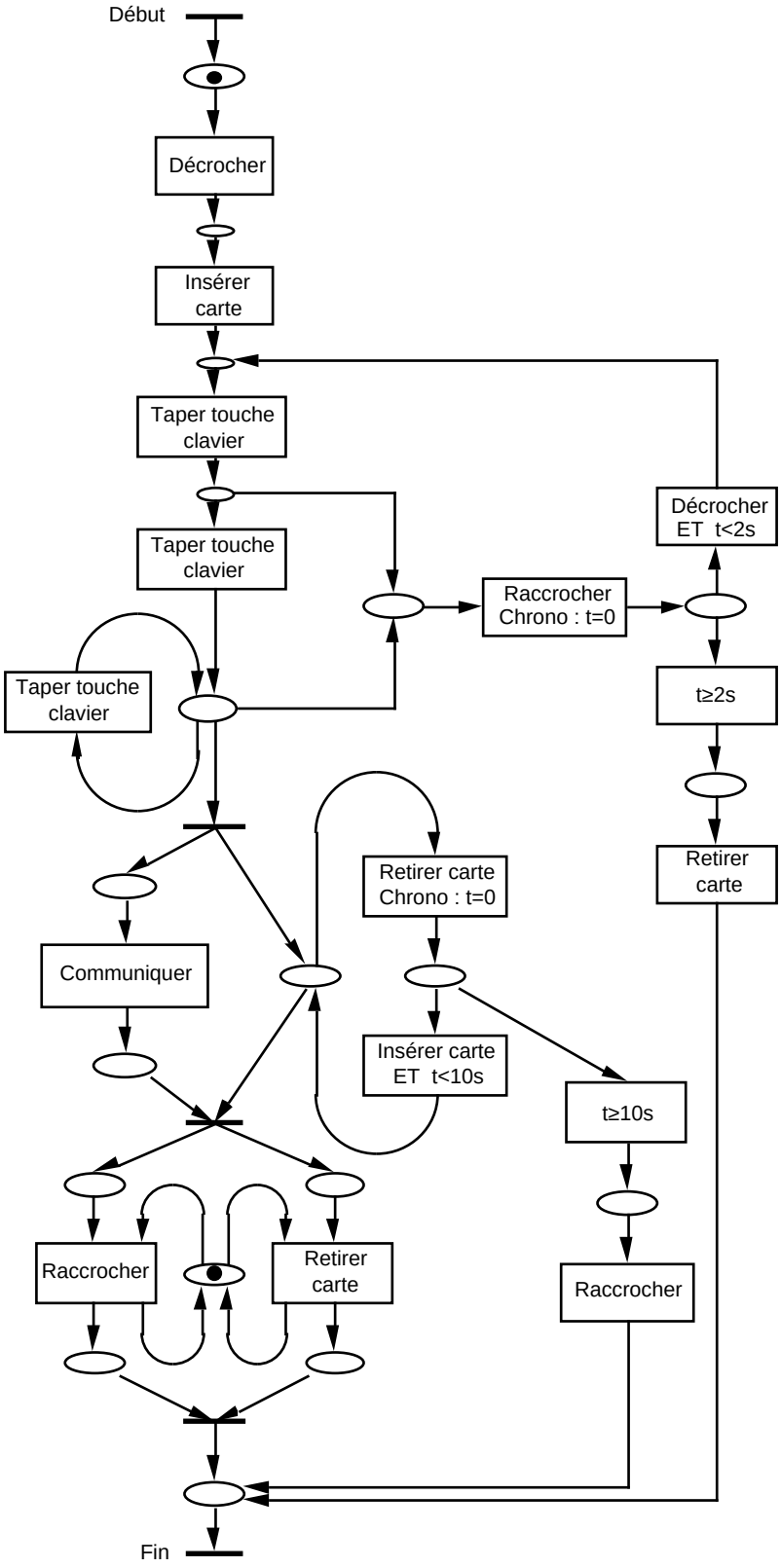

Figure III-13 : Description de la tâche téléphoner à l'aide des réseaux de Petri.

La modélisation du Publiphone à l'aide des réseaux de Petri, reproduite figure III-13, a été obtenue à partir de la modélisation UAN du Publiphone du paragraphe précédent (§ 3.5.d). Nous avons utilisé les équivalences proposées par Palanque, Bastide, et Senges [Palanque et al. 1995b] afin de générer le squelette de l'interaction. Le cas des interruptions à été spécifié "à la main" a posteriori. La description finale est a peu près aussi imposante que celle utilisant MAD. Du fait du faible pouvoir d'abstraction des réseaux de Petri, la lisibilité de la description est faible. Les réseaux de Petri ne permettent pas d'extraire d'un premier regard l'organisation globale de l'interaction. L'usage de macro-transitions permettrait de disposer de plusieurs niveaux d'abstraction. Cependant l'usage de telles macro-transitions est ici difficile du fait des interruptions.

Les problèmes rencontrés avec les réseaux de Petri sont similaires à ceux détectés avec UAN. Il leur est par exemple impossible de "tuer" une tâche. En conséquence, la tâche "composer le numéro" est formalisée d'une manière comparable à UAN, en descendant d'un niveau d'abstraction et en considérant plusieurs tâches "taper touche clavier". De plus, les réseaux de Petri ne modélisent pas explicitement la notion d'interruption, ceci impose de les considérer comme des tâches optionnelles.

Par contre, les contraintes temporelles sont aisées à spécifier à l'aide des réseaux de Petri. C'est le cas pour le délai minimum de dix secondes lors du changement de Télécarte, de même que l'existence du chien de garde de deux secondes lors de l'erreur de numérotation. Notons que le formalisme des réseaux de Petri est le seul à permettre de décrire ce qu'il advient si le délai expire : l'utilisateur doit alors retirer sa Télécarte et recommencer son appel au début.

Les réseaux de Petri ont une définition formelle et une capacité à décrire les contraintes temporelles très intéressantes. Pourtant ils sont complexes à utiliser et à relire. En outre, ils ne permettent que difficilement l'expression des retours d'information de l'interface. C'est pourquoi ils sont plus adaptés à la formalisation des tâches de bas niveau, comme par exemple la définition du double-clic d'une souris [Accot, Chatty, & Palanque 1996] qu'à l'analyse de tâche.

3.5.f. La spécification des interruptions

Ayant fait l'état des lieux, nous constatons qu'aucun des formalismes précédemment utilisés n'apporte de solution réellement satisfaisante. Leur pouvoir d'expression est en effet trop limité. Ils doivent donc être étendus. Les blocs de connaissance ne sont pas utilisés dans la suite de ce mémoire car ils conviennent plus à la description de la connaissance opérationnelle qu'à la spécification d'une interface. Cependant les idées introduites par la notion de condition anormale sont reprises par les extensions proposées au chapitre IV.

Avant de conclure sur les conséquences des choix réalisés, citons une méthode de choix d'un formalisme plus anecdotique que scientifique : "un formalisme conviendra à celui qui l'utilise si ce dernier et celui qui l'a créé partagent une formation et/ou un domaine de recherche commun". Cette méthode ne doit cependant pas être considérée comme farfelue, car les problèmes rencontrés par un spécialiste d'un domaine sont probablement les mêmes que ceux rencontrés par ses collègues.

4. En conclusion

4.1. Choix d'un formalisme

Nous avons effectué, au début de ce chapitre, un tour d'horizon des classifications proposées par différents auteurs. Nous les avons classées en quatre catégories : selon leur pouvoir d'expression, leur utilisabilité, leur origine, et leur finalité. Nous avons également évoqué les Principes de Namur qui, sans vraiment faire partie d'un type de classification, représentent une sorte de cahier des charges d'un "bon" formalisme adapté à la spécification des interfaces homme-machine.

Nous avons également montré que le choix d'un formalisme adapté aux besoins définis par une équipe de conception peut s'effectuer en utilisant les classifications déjà réalisées. Un ordre d'utilisation de ces classifications a été suggéré et illustré par notre propre recherche d'un formalisme adapté à nos besoins. La méthode de choix proposée n'est cependant pas exempte de critiques. Désirant aller du général, la finalité, au particulier, l'utilisabilité, elle se heurte à l'insuffisance des classifications disponibles. Néanmoins, celle-ci a permis de monter qu'une grande partie du choix d'un formalisme peut s'effectuer sans réaliser une étude complète, et fort coûteuse, de chacun des formalismes disponibles.

4.2. Limites des classifications

Nous avons mis en lumière bon nombre d'insuffisances et de limitations des classifications proposées. Elles sont incomplètes autant du point de vue des critères évalués que du nombre de formalismes pris en compte. Il est évident qu'une classification prenant en compte tous les critères et toutes les classifications publiées n'est pas envisageable. Cependant, il faut encourager la constitution de classifications portant sur un grand nombre de formalismes, même au détriment du nombre de critères évalués. En effet, la fusion des classifications permet alors l'évaluation multi-critères selon l'approche proposée précédemment.

La méthode inverse est moins intéressante car il est préférable que la même personne évalue le même critère sur plusieurs formalismes. En effet, on se heurte ici à un autre problème des classifications, la subjectivité des évaluations. Cette subjectivité dans l'évaluation d'une classification peut difficilement être enrayée. Cependant, lorsque le nombre de formalismes étudié est important, cette subjectivité peut être "étalonnée" par l'utilisateur de l'évaluation en comparant les résultats publiés à ses connaissances sur l'un des formalismes cités.

Même si nous n'avons pas utilisé les Principes de Namur lors de notre choix, ils constituent une vision intéressante des formalismes. Ils doivent maintenant entrer dans une phase pratique pour être réellement utilisés. C'est-à-dire que des évaluations doivent être réalisées sur ces principes. Compte tenu de leur établissement récent, ils ne sont pas encore publiés au moment où ces lignes sont écrites, l'absence d'évaluation se basant sur ces principes n'est pas surprenante.

4.3. Conséquences du choix des formalismes

Nous avons choisi d'utiliser plusieurs formalismes, MAD, UAN, et les réseaux de Petri, chacun adapté et optimisé pour une phase du cycle de conception clairement définie. Ce choix nous oblige à réaliser des passerelles entre ces formalismes, de manière à garder une mémoire des choix effectués entre les différentes phases du cycle de conception. Ces passerelles peuvent par ailleurs prendre l'aspect de traducteurs, de manière à présenter à chaque intervenant le travail réalisé par d'autres dans un formalisme compatible avec ses aspirations. La définition de ces passerelles fait l'objet du chapitre IV.

En outre, nous avons constaté que les formalismes choisis n'ont pas la puissance d'expression requise pour spécifier erreurs et interruptions. En conséquence, ils doivent être étendus, et les extensions proposées doivent être compatibles avec les passerelles à définir entre les formalismes. De plus, ces extensions doivent respecter les bases des formalismes étendus de manière à conserver les caractéristiques qui nous les ont fait choisir. Comme les passerelles, les extensions proposées sont détaillées au chapitre IV.

5. Bibliographie

[Abowd et al. 1989] Abowd G., Bowen J., Dix A., Harrison M., & Took R. *User interface languages: A survey of existing methods*. Programming Research Group at Oxford University Computing Laboratory, October 1989. Report PRG-TR-5-89.

[Accot et al. 1996] Accot J., Chatty S., & Palanque P. A formal description of low level interaction and its application to multimodal interactive systems. *Third International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'96), Namur, Belgium*, 5-7 June 1996.

[Balbo 1994] Balbo S. *Un pas vers l'évaluation automatique des interfaces homme-machine*. Thèse en Informatique : Université Joseph Fourier (Grenoble I), 5 Septembre 1994.

[Bordon 1994] Bordon S. *Formalisme pour la représentation des scénarios dans les didacticiels*. DEA Système d'information : Université Joseph Fourier (Grenoble I), Juin 1994.

[Bowen et al. 1995] Bowen J.P. & Hinchey M.G. Seven more myths on formal methods. *IEEE Software*, 1995.

[Boy 1989] Boy G. The block representation in knowledge acquisition for computer integrated documentation. *4th AIAA-sponsored knowledge acquisition integrated documentation for knowledge-based systems workshop, Banff, Canada*, October 1989.

[Brun et al. 1995] Brun P. & Beaudoin-Lafon M. A taxonomy and evaluation of formalisms for the specification of interactive systems. *10th annual conference of the British Human-Computer Interaction Group (HCI'95), University of Huddersfeild (UK)*, August-September 1995.

[Card et al. 1983] Card S.K., Moran T.P., & Newell A. *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, 1983.

[Coutaz 1990] Coutaz J. *Interfaces homme-ordinateur : Conception et réalisation*. Paris : Dunod informatique, 1990.

[Coutaz et al. 1993a] Coutaz J., Faconti G., Paterno F., Nigay L., & Salber D. *MATIS: A UAN description and lesson learned*. ESPRIT BRA 7040 Amodeus-2, June 1993a. Working paper SM/WP14.

[Coutaz et al. 1993b] Coutaz J., Paterno F., Giorgio F., & Nigay L. A comparison of approaches for specifying multimodal interactive systems. *European Research Consortium for Informatics and Mathematics (ERCIM'93), Nancy, France*, 2-4 November 1993b. p. 165-174.

[Frank et al. 1993] Frank M. & Foley J. Model-based User Interface Design by Example and by Interview. *ACM symprosium on User Interface Software and Technology (UIST'93), Atlanta (Georgia), USA*, November 3-5 1993.

[Garavel 1989] Garavel H. *Compilation et vérification de programmes LOTOS*. Thèse en informatique : Université Joseph Fourier (Grenoble I), Novembre 1989.

[Gray et al. 1994] Gray P., England D., & McGowan S. *XUAN: Enhancing the UAN to capture temporal relation among actions*. Department of Computing Science, University of Glasgow, February 1994. Department research report IS-94-02.

[Hall 1990] Hall A. Seven myths on formal methods. *IEEE Software*, September 1990. p. 11-19.

[Harrison et al. 1994] Harrison M.D. & Duke D.J. *A review of the formalisms for describing interactive behaviour*. ESPRIT BRA 7040 Amodeus-2, January 1994. Working paper SM/WP28.

[Hartson et al. 1990] 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, p. 181-203.

[ISO-8807:1989 1995] ISO-8807:1989. *Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour*. Internation Standard Organisation (ISO), Last update 1995. Standard ISO 8807:1989.

[Jackson 1983] Jackson M.A. *System developement*. Englewood Cliffs (NJ), USA : Prentice Hall, 1983.

[Jambon 1995] Jambon F. Interruptions et formalismes de scripts de tâches. *Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95), Toulouse, France*, 11-13 Octobre 1995. p. 161-168.

[Johnson 1996a] Johnson C. The evaluation of user interface design notations. *Proceedings of the Design, Specification and Verification of Interactive Systems (DSV-IS'96), Namur, Belgique*, 1996a.

[Johnson 1996b] Johnson C. The Namur principles: criteria for the evaluation of user interface notations. *Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'96), Namur, Belgique*, 1996b.

[Johnson et al. 1993] Johnson H. & Johnson P. ADEPT-advanced design environment for prototyping with task models. *Human Factors in Computing Systems (InterCHI'93), Amsterdam, The Netherlands*, 24-29 April 1993. vol. adjunct Proceedings(2), p. 56.

[Lim et al. 1994] Lim K.Y. & Long J.B. *The MUSE method for usability engineering*. United Kingdom : Cambridge University Press, 1994.

[Mathé 1990] Mathé N. *Assistance intelligente au contrôle de processus : Application à la télémanipulation Spatiale*. Thèse en Intelligence Artificielle : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1990.

[Nigay 1994] Nigay L. *Conception et modélisation logicielles des systèmes interactifs : Application aux interfaces multimodales*. Thèse en informatique : Université Joseph Fourier (Grenoble I), 1994.

[Palanque et al. 1995a] Palanque P. & Bastide R. Spécifications formelles pour l'ingénierie des interfaces homme machine. *Techniques et Science Informatiques*, 1995a. vol. 14, n° 4, p. 473-500.

[Palanque et al. 1995b] Palanque P., Bastide R., & Senges V. Task model - system model: towards an unifying formalism. *HCI International conference, Yokoohama, Japan*, 9-14 July 1995b. p. 489-494.

[Peterson 1981] Peterson J.L. *Petri net theory and modelling of systems*. Prentice Hall, 1981.

[Pleczon 1992] Pleczon P. *Éléments de méthodologie et outils pour l'assistance à l'opérateur : Application à la conduite automobile*. Thèse en Informatique : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1992.

[Rasmussen 1986] Rasmussen J. *Information processing and Human-Machine Interaction : An approach to cognitive engineering*. North-Holland, 1986.

[Robert 1992] Robert P. *Le petit Robert : Dictionnaire de la langue française*. Paris, France : Le Robert, 1992.

[Rouff 1996] Rouff C. Formal specification of user interface. *ACM SIGCHI bulletin*, July 1996. vol. 28, n° 3, p. 27-33.

[Scapin et al. 1989] Scapin D.L. & Pierret-Golbreich C. MAD : Une méthode analytique de description des tâches. *Colloque sur l'ingénierie des Interfaces Homme-Machine (IHM'89), Sophia-Antipolis, France*, Mai 1989. p. 131-148.

[Sebillotte et al. 1994] Sebillotte S., Alonso B., Fallah D., Hamouche H., & Scapin D.L. *Note de recherche concernant le formalisme MAD*. INRIA Rocquencourt, Novembre 1994. Document interne.

[Sebilotte 1991] Sebilotte S. *Décrire des tâches selon les objectifs des opérateurs : De l'interview à la formalisation*. Le Travail humain, Paris, France : Presses Universitaires de France, 1991. p. 193-223.

[Szekely et al. 1993] Szekely P., Luo P., & Neches R. Beyong interface builders: model-based interface tools. *Human Factors in Computing Systems (InterCHI'93), Amsterdam, The Netherlands*, 24-29 April 1993. vol. conference proceedings(2), p. 383-390.

[Szekely et al. 1996] 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*, 1996.

Chapitre IV

Extensions et Traducteurs

1. Introduction

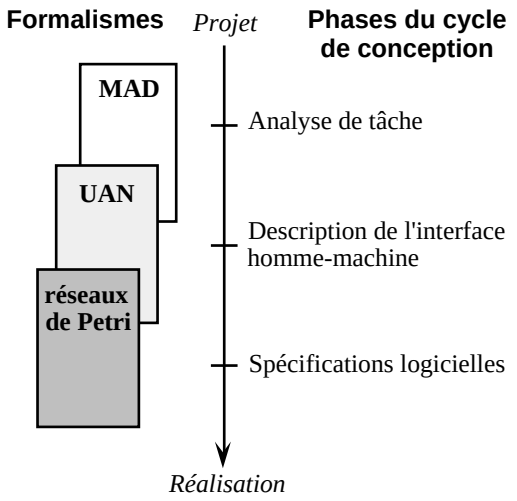

Figure IV-1 : Espace de couverture des formalismes vis-à-vis du cycle de conception.

Plutôt que de créer de toute pièce un nouveau formalisme adapté aux besoins de la spécification des singularités, nous avons préféré étendre les formalismes existants. Nous avons en effet constaté que, si aucun d'entre eux ne rassemble à lui seul toutes les qualités requises, chacun d'entre eux est adapté à l'une des phases du cycle de conception. Notre choix, justifié au chapitre III, s'est porté sur trois formalismes. Le premier, MAD [Scapin & Pierret-Golbreich 1989], a été conçu par des ergonomes afin de servir de support à l'analyse de tâche. À ce titre, il est associé à une méthode psycho-ergonomique dont l'aboutissement est une description MAD. Cependant MAD ne permet pas l'expression des retours d'information en provenance de l'interface. De même, les contraintes temporelles sont difficiles à exprimer. C'est pourquoi l'usage de UAN [Hartson, Siochi, & Hix 1990] est nécessaire. Grâce à son format tabulaire, mettant en vis-à-vis les actions de l'utilisateur et les retours d'information de l'interface, UAN permet de décrire précisément le comportement perceptible de l'interface. Cependant UAN devient rapidement peu lisible et ambigu lorsque l'on s'intéresse aux actions de très bas niveau comme l'expression du double-clic d'une souris par exemple. Il est alors nécessaire de se tourner vers les réseaux de Petri [Peterson 1981]. Leur rigueur mathématique et leur déterminisme sont en effet bien adaptés à la formalisation des comportements de bas niveau d'une interface, comme par exemple la spécification du double-clic d'une souris [Accot, Chatty, & Palanque 1996]. En résumé, et comme l'illustre la figure VI-1, chacun des formalismes choisis est mieux adapté que les autres à l'une seulement des trois phases du cycle de conception que nous avons distinguées. C'est pourquoi leur usage se doit d'être complémentaire.

La conséquence immédiate de ce choix est une multiplication des formalismes utilisés : nous en avons trois. En corollaire, ce choix impose de

retranscrire, à chaque phase du cycle de conception, la spécification complète de l'interface à partir des données de l'étape précédente. Ce surplus de travail n'est pas tolérable lorsque l'interface spécifiée est de taille imposante. C'est pourquoi l'usage de traducteurs est une absolue nécessité. En outre, nous avons mis en lumière au sein du chapitre III, les limites du pouvoir d'expression des formalismes choisis. Il est donc nécessaire de les étendre afin qu'ils soient aptes à spécifier erreurs et interruptions. L'usage des trois formalismes retenus étant complémentaire, ces extensions doivent s'appliquer à tous ces formalismes, et disposer de traducteurs de façon à garantir la conformité des spécifications.

MAD, UAN, et les réseaux de Petri disposent de nombreuses versions et extensions. Les modifications apportées aux formalismes originaux par différents auteurs sont parfois suffisamment sensibles pour constituer à elles seules un nouveau formalisme. Se pose alors le problème du choix parmi les versions disponibles provenant d'un même formalisme racine :

• La *Méthode Analytique de Description* des tâches (MAD) [Scapin et al. 1989] a subi quelques extensions depuis sa création pour prendre en compte notamment les interruptions. La version actuelle de MAD s'est stabilisée et fait l'objet d'un document de synthèse dont il faut souligner l'existence [Sebillotte, Alonso, et al. 1994]. Depuis peu, MAD a été étendue à nouveau, sous le nom de MAD*, par Hamouche [Hamouche 1995]. Nous avons choisi de prendre comme référence la version de MAD basée sur le document de synthèse. Cette version est la plus connue, la mieux documentée, et c'est également celle évaluée au chapitre III. Cependant les extensions effectuées, de même que les traducteurs proposés, peuvent probablement s'appliquer aussi à MAD* au prix de quelques adaptations mineures. En outre, certaines des extensions notables de MAD* ont été prises en compte lors de la définition des traducteurs.

• La *Notation des Actions Utilisateur* ou en version originale "User Action Notation" (UAN) [Hartson et al. 1990] a été également étendue depuis sa création, afin de tenir compte des contraintes temporelles [Hartson & Gray 1992]. Encore étendue en XUAN [Gray, England, & McGowan 1994], elle a été ensuite rendue partiellement exécutable sous le nom de eXUAN [McGowan 1995]. À notre connaissance, il n'existe aucun document de synthèse décrivant UAN et ses extensions. C'est de notre point de vue une lacune dommageable qui entrave son utilisation hors du domaine de la

recherche. Nous utiliserons comme référence la version de UAN qui nous a semblé la plus courante, décrite sommairement au chapitre III. Elle fait l'objet de l'introduction de l'article sur XUAN cité précédemment [Gray et al. 1994].

• Les *réseaux de Petri* [Peterson 1981] ont subi, eux aussi, certaines extensions. Parmi celles-ci, citons l'ajout de contraintes temporelles [Sifakis 1977], et l'approche objet [Palanque 1992]. Comme précédemment, et dans le but de généraliser et de simplifier les extensions et les traducteurs que nous proposons, nous avons choisi de considérer la version originale des réseaux de Petri auxquels nous avons dû ajouter les contraintes temporelles de Sifakis lues dans [Palanque, Bastide, & Senges 1995]. Néanmoins, compte tenu du fait que les extensions proposées ne sont en fait que des heuristiques et des structures remarquables, celles-ci peuvent s'appliquer a priori à toutes les extensions des réseaux de Petri, à la condition qu'elles respectent leur structure originale.

Ce chapitre s'articule autour de deux parties : la première est consacrée à la description des extensions des formalismes choisis et à leur évaluation. La seconde s'intéresse aux traducteurs entre les formalismes et illustre leur utilisation par un nouvel exemple de spécification du Publiphone.

2. Extensions

Nous avons défini au chapitre II la notion de singularité, dont nous avons extrait une classification reproduite figure IV-2 ci-contre. Cette classification, dont nous allons maintenant nous inspirer afin d'étendre les formalismes, distingue quatre principaux types de singularité : singularité avec parallélisme, singularité avec entrelacement, singularité avec séquence sans reprise, et singularité avec séquence puis reprise. Pour ce dernier cas, nous avons distingué trois cas particuliers de reprise fréquents :

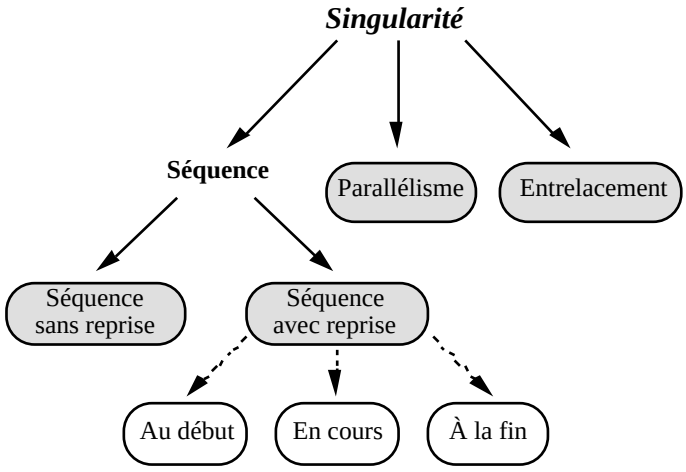

Figure IV-2 : Classification des singularités.

reprise au début, reprise en cours, et reprise à la fin de la tâche interrompue. Ces trois cas ne doivent cependant pas faire oublier le cas général de reprise en un point quelconque.

Cette section s'articule autour de trois thèmes. Tout d'abord, nous allons évoquer les principes retenus et les choix effectués lors de la définition des extensions. C'est l'objet du paragraphe 2.1. Nous récapitulerons ensuite les extensions proposées, sous forme de tableau, au paragraphe 2.2. Enfin, le paragraphe 2.3 a pour objet l'évaluation de ces extensions selon les principes de Namur évoqués au chapitre III.

2.1. Principes retenus

Les extensions proposées doivent préserver l'utilisabilité des formalismes étendus, nous évoquons ce point en premier (§ 2.1.a). Dans un deuxième temps, nous avons dû clarifier l'usage des priorités (§ 2.1.b). Puis, nous justifions notre choix de profondeur d'interruption, ainsi que notre définition de tâche élémentaire (§ 2.1.c).

2.1.a. Utilisabilité

La description des singularités, malgré leur importance, ne doit pas être un élément omniprésent au sein d'une spécification, au sens où elle ne doit pas masquer complètement l'interaction "normale". Par exemple, les singularités peuvent représenter des procédures de correction d'erreur qui sont exceptionnelles, et ne doivent en conséquence pas prendre une place prépondérante dans une spécification.

Les extensions proposées ne doivent pas non plus être des "verrues" sommairement greffées aux formalismes étendus. Elles doivent s'intégrer parfaitement à ceux-ci, et ne pas modifier leur finalité originale. En particulier, il n'est pas concevable de rajouter la notion d'attribut à un formalisme qui n'en dispose pas à l'origine. De plus, lorsque la notion d'interruption existe déjà, il convient de l'étendre en respectant autant que possible les principes et la syntaxe d'origine.

Contrairement à DIANE [Barthet 1988], nous avons choisi de spécifier explicitement les singularités. DIANE en effet suppose que toute tâche est potentiellement interruptible sauf mention contraire. L'intérêt majeur d'une telle démarche est une simplification de l'écriture de la spécification. Le concepteur ne distingue ainsi plus que les cas où il souhaite interdire les

interruptions. Cette démarche est bien adaptée aux systèmes d'information peu critiques où l'état d'interruption permanent est de rigueur. Cependant, dans le cas des systèmes critiques, nous désirons avant tout que le concepteur prennent en compte les singularités explicitement. Cette prise en compte se manifeste en particulier par la définition d'un prologue et d'un épilogue pour chaque tâche interruptible.

2.1.b. Priorités

La notion de priorité entre les tâches n'est pas partagée par tous les formalismes. Seul MAD la définit explicitement pour les tâches parallèles ou simultanées. Cette notion n'est explicite ni avec UAN, ni avec les réseaux de Petri. MAD utilise les priorités dans le but de déterminer les tâches pouvant interrompre la tâche en cours. De notre point de vue, les singularités doivent être définies explicitement et non pas implicitement par le biais d'un jeu de priorités comme le suggère MAD. En outre, MAD* complique le processus en y ajoutant plusieurs niveaux de priorités et la notion d'interruptibilité locale.

Les priorités sont habituellement définies sur le modèle des niveaux d'interruption des processeurs. C'est l'option prise par MAD*. L'un des inconvénients d'une telle méthode est qu'il faille définir, pour un grand nombre de tâches du système potentiellement interruptibles ou interrompantes, leur appartenance à l'un des niveaux d'interruption. On définit ainsi une relation d'ordre partielle (ou plus rarement totale) sur l'ensemble des tâches. Au-delà de la complexité d'une telle spécification, certaines situations peuvent provoquer une modification de la priorité entre les tâches selon le contexte. Imaginons deux tâches "répondre au téléphone" et "arrêter l'ordinateur". Il est assez évident que la première tâche est prioritaire sur la seconde. Cependant cet ordre "évident" peut se révéler caduque si des volutes de fumée sortent des évents d'aération de l'ordinateur en question…

Afin de maintenir la compatibilité entre les trois formalismes utilisés, nous avons préféré ne pas utiliser la notion de priorité. Ce choix n'est pas sans conséquences sur le déterminisme du déclenchement des interruptions. Sans l'utilisation des priorités, il n'est en effet pas possible de choisir parmi deux interruptions se déclenchant suite au changement d'une même condition. Un choix doit pourtant être fait. De notre point de vue, ce choix doit être intégré aux conditions de déclenchement des singularités. En conséquence, il ne doit

pas être possible à deux singularités associées à une même tâche de disposer de conditions de déclenchement vraies simultanément. Chaque condition de déclenchement doit donc prendre en compte l'existence des autres singularités potentielles, et leur propres conditions de déclenchement, de telle façon à ce que les zones de déclenchement soient disjointes. Ce point de vue équivaut à définir sous une autre forme des priorités locales entre les singularités associées à une même tâche. Bien entendu, ces priorités locales peuvent être différentes pour deux mêmes singularités associées à des tâches différentes.

2.1.c. Profondeur d'interruption et tâches élémentaires

Par défaut, la profondeur d'interruption des singularités de MAD et UAN est posée à un. C'est-à-dire que les sous-tâches d'une tâche interruptible ne sont pas interruptibles, sauf mention contraire. Le problème ne se pose pas dans les mêmes termes pour les réseaux de Petri, car tous les points d'interruption doivent être définis explicitement. Cependant dans le cas de MAD et UAN, il peut être utile de spécifier une profondeur d'interruption supérieure à un. Ce peut être réalisé en surchargeant l'attribut INTER de MAD, ou la relation temporelle "interruption" de UAN avec la profondeur désirée, le symbole infini "∞" signifiant jusqu'aux tâches élémentaires des réseaux de Petri.

Par définition avec MAD, et par construction avec les réseaux de Petri, les tâches élémentaires sont ininterruptibles. UAN n'indique rien de précis à ce sujet, mais il semble que ce soit également le cas. L'utilisation complémentaire de trois formalismes permet des affinements successifs de la spécification. En conséquence, les tâches élémentaires du point de vue de MAD peuvent encore être affinées à l'aide de UAN, puis encore à l'aide des réseaux de Petri. Afin de maintenir la compatibilité avec la profondeur d'interruption définie ci-dessus, nous posons que les tâches élémentaires de MAD et UAN peuvent être interruptibles. Seules celles des Réseaux de Petri ne le sont pas. Formulé différemment, ceci équivaut à considérer une notion de tâche élémentaire globale sur l'ensemble des formalismes. C'est-à-dire qu'une tâche n'est élémentaire et ininterruptible que si elle n'est affinée par aucun des trois formalismes utilisés.

2.1.d. Vocabulaire

Afin de maintenir une certaine compatibilité avec le vocabulaire utilisé habituellement, nous utiliserons désormais le mot "interruption" à la place de "singularité". Notons que c'est un abus de langage car une singularité avec

parallélisme n'interrompt pas au vrai sens du terme la tâche "singularisée". Cependant c'est là un moindre mal, car le mot "singularité" n'est pas (encore) entré dans le vocabulaire du domaine.

2.2. Particularités des formalismes

2.2.a. MAD

Les extensions proposées pour MAD s'appuient sur une réutilisation des constructeurs existants. Actuellement, l'attribut "INTER" de MAD n'est valide que pour des tâches parallèles ou simultanées. Nous proposons de le rendre valide également pour le constructeur séquence. La syntaxe proposée est la suivante : le constructeur est en charge de préciser le type de relation temporelle entre l'interruption et la tâche interrompue (parallélisme, entrelacement, ou séquence), tandis que l'attribut "INTER" indique la possibilité d'une interruption, et éventuellement précise son mode de reprise pour le cas de la séquence. En résumé on obtient :

- Constructeur SIM et attribut INTER : Interruption avec parallélisme
- Constructeur PAR et attribut INTER : Interruption avec entrelacement
- Constructeur SEQ et attribut INTER (pas de reprise) : Interruption avec séquence sans reprise
- Constructeur SEQ et attribut INTER (reprise en point_de_reprise) : Interruption avec séquence et reprise en un point
- Constructeur SEQ et attribut INTER (reprise au début) : Interruption avec séquence et reprise au début
- Constructeur SEQ et attribut INTER (reprise à la fin) : Interruption avec séquence et reprise à la fin
- Constructeur SEQ et attribut INTER (reprise en cours) : Interruption avec séquence et reprise en cours

2.2.b. UAN

Tout comme MAD, les relation temporelles liées à la notion d'interruption de UAN ont été étendues. La symbologie définie par les auteurs de UAN a été conservée dans son principe. Le symbole "<-" signifiant interruption a été concaténé avec la relation temporelle adéquate. Par exemple, la relation temporelle "<-//", signifie interruption "<-" puis parallélisme "//". Le cas des interruptions avec séquence mérite plus d'attention. Au symbole interruption "<-" nous devons concaténer le symbole séquence "," puis distinguer cinq nouveaux cas. Pour cela nous avons dû inventer de nouveaux symboles. La

croix ou “x” signifie “pas de reprise”, la flèche vers le haut ou “^” signifie reprise au début, la flèche vers le bas ou “v” signifie reprise à la fin, le trait au milieu ou “-” signifie reprise en cours, et enfin les parenthèses “(point_de_reprise)” indiquent le point de reprise dans le cas général. Cette symbologie utilise un jeu de caractères standard afin de s’adapter à la majorité des systèmes. En outre, de manière à alléger la notation, le symbole séquence “,” peut être omis, de même que le symbole reprise en cours ”-”. On retrouve ainsi une compatibilité ascendante avec UAN qui définit l’interruption avec reprise en cours par le symbole “<-”. En résumé on obtient :

- <-// : Interruption avec parallélisme
- <-<-> : Interruption avec entrelacement
- <-x ou <-,x : Interruption avec séquence sans reprise
- <-(pt) ou <-,(pt) : Interruption avec séquence et reprise en un point
- <-^ ou <-,^ : Interruption avec séquence et reprise au début
- <-v ou <-,v : Interruption avec séquence et reprise à la fin
- <- ou <-,- : Interruption avec séquence et reprise en cours

2.2.c. Réseaux de Petri

Le déterminisme et la rigueur mathématique des réseaux de Petri ne permet pas aisément l’ajout de relations temporelles concernant les interruptions. C’est pourquoi les extensions proposées ne sont en fait que des heuristiques de conception et des structures remarquables. Elles n’en sont pas moins utiles, en particulier elles permettent de définir formellement les extensions proposées pour MAD et UAN. En outre, une spécification à l’aide des réseaux de Petri impose la spécification obligatoire de tous les points d’interruption. Les tableaux récapitulatifs du paragraphe 2.3 ci-après donnent l’exemple du cas d’un seul point d’interruption. Si plusieurs points d’interruptions doivent être spécifiés, les structures proposées doivent être dupliquées. Notons que ces structures ont été définies, et ne sont valables que pour des tâches pouvant se décomposer en tâches purement séquentielles. Dans les autres cas, il est nécessaire de reporter la structure proposée sur chacune des sous-tâches de la tâche composée. On remarque aisément que la spécification des interruptions selon les réseaux de Petri est complexe, en particulier pour la cas de l’entrelacement. Le concepteur peut alors être amené, s’il le désire, à définir une macro-transition pour l’ensemble tâche interrompue et tâche d’interruption comme celle indiquée figure VI-3 ci-dessous.

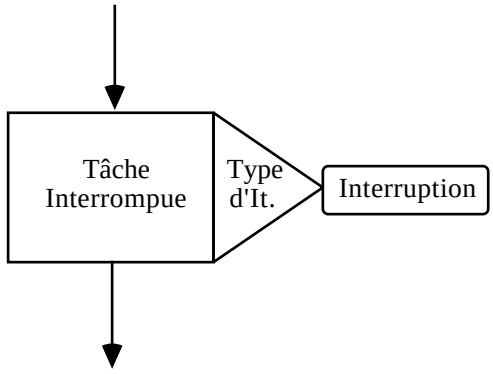

Figure VI-3 : Macro-transition décrivant une interruption.

2.3. Tableaux récapitulatifs

Les sept tableaux reproduits figure VI-4 ci-après récapitulent les différents types d'interruption identifiés. Pour chaque tableau, de haut en bas sont indiqués le type d'interruption, puis l'extension proposée pour MAD, puis l'extension pour UAN, et enfin la structure équivalente spécifiée à l'aide des réseaux de Petri.

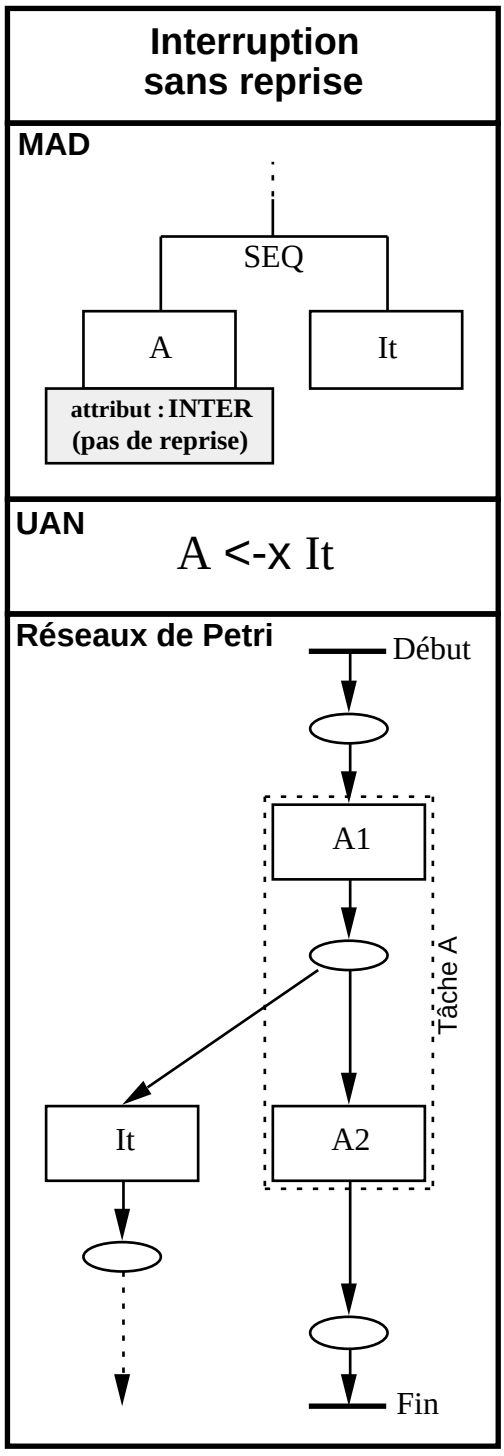

Figure VI-4 : Tableaux récapitulatifs des interruptions (1/7).

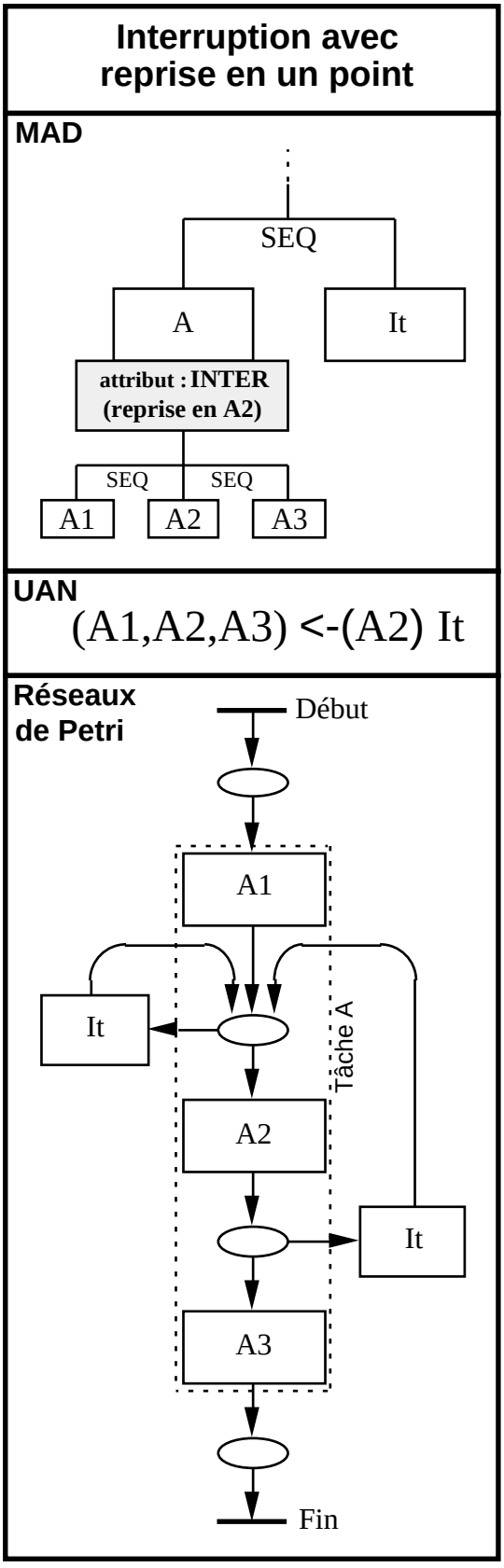

Figure VI-4 : Tableaux récapitulatifs des interruptions (2/7).

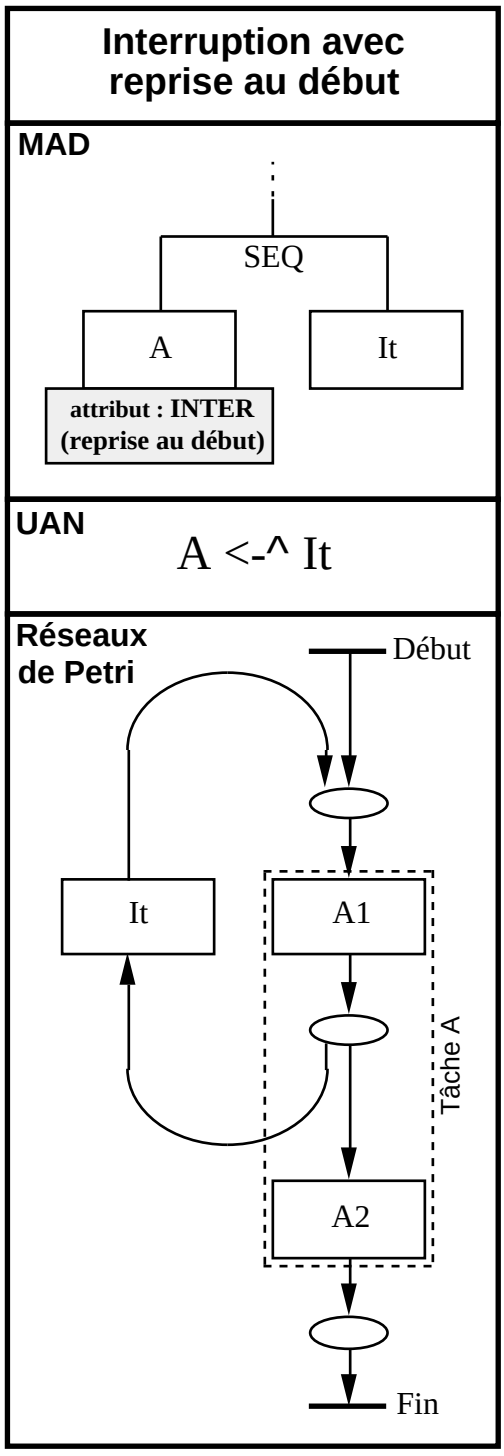

Figure VI-4 : Tableaux récapitulatifs des interruptions (3/7).

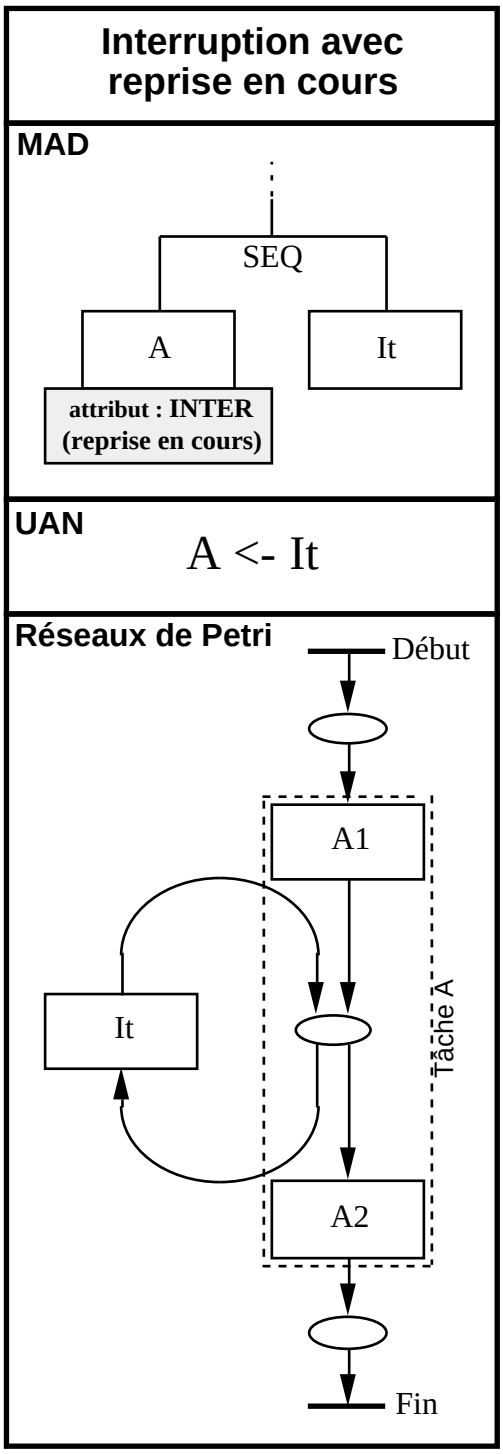

Figure VI-4 : Tableaux récapitulatifs des interruptions (4/7).

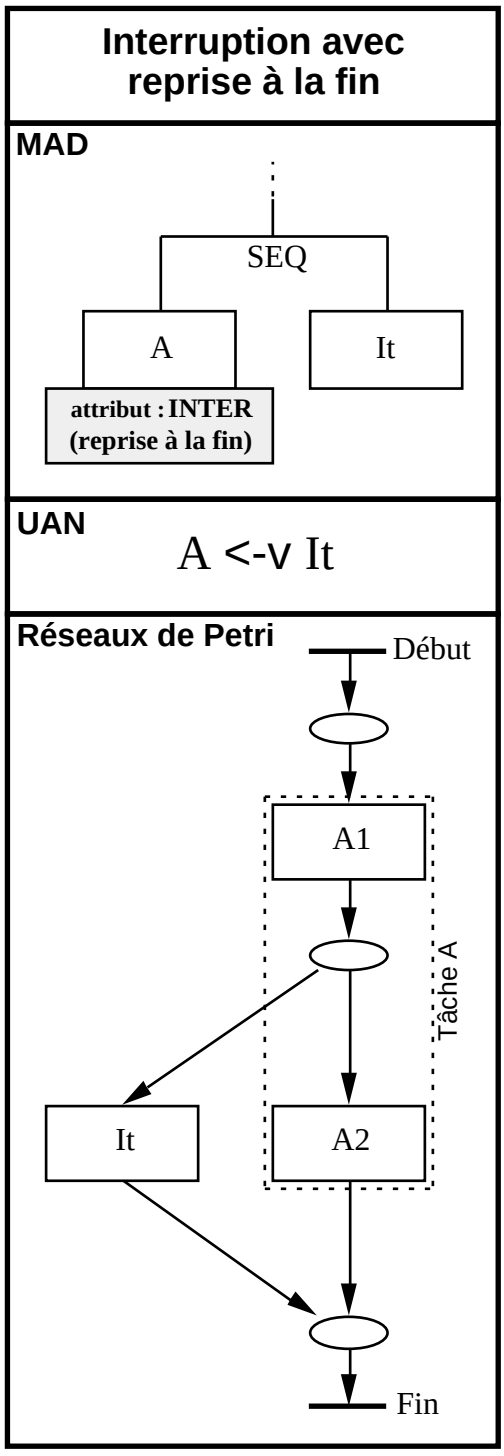

Figure VI-4 : Tableaux récapitulatifs des interruptions (5/7).

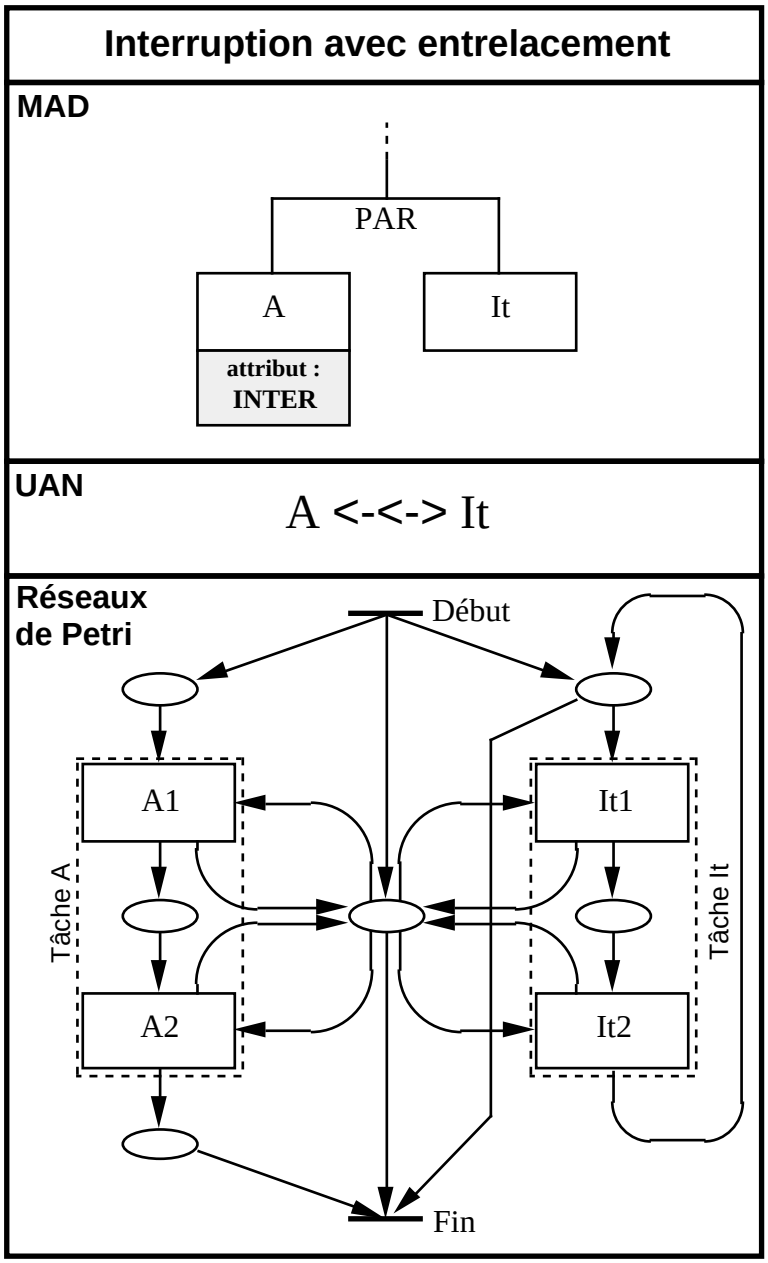

Figure VI-4 : Tableaux récapitulatifs des interruptions (6/7).

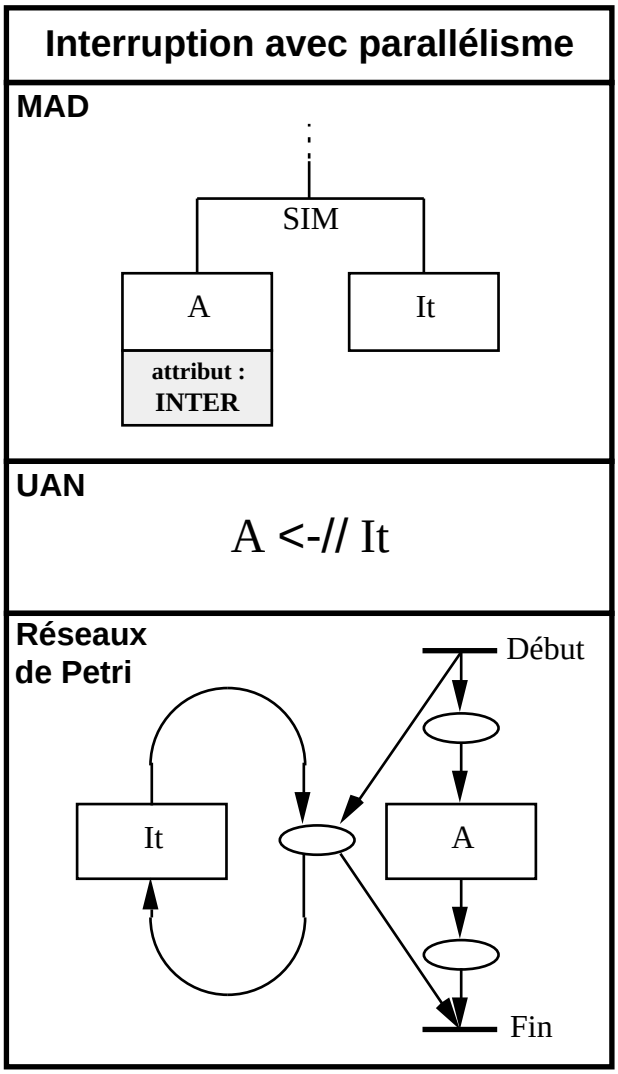

Figure VI-4 : Tableaux récapitulatifs des interruptions 7/7).

2.4. Évaluations selon les “Principes de Namur”

Nous allons maintenant vérifier la conformité des formalismes étendus avec les “Principes de Namur” évoqués au paragraphe 2.5 du chapitre III [Johnson 1996]. Cette évaluation fait appel aux méthodes d’évaluation recommandées par les auteurs des Principes, lorsqu’il est possible de les appliquer :

• *La capacité à abstraire* est respectée à la fois par l’usage complémentaire par affinements successifs de plusieurs formalismes, et par la structure même des formalismes utilisés. MAD et UAN possèdent en effet une structure permettant la décomposition en tâches et sous-tâches. De même, les réseaux de Petri possèdent une capacité à abstraire, certes plus limitée, par l’usage de macro-transitions regroupant une série de transitions.

• *La capacité à structurer* est satisfaite par l’usage complémentaire de trois formalismes, chacun adapté aux besoins d’une spécialité. Cependant, il n’est pas possible a priori de réaliser les trois spécifications indépendamment et en même temps comme le suggèrent les auteurs. En effet, les spécifications ayant lieu par affinements successifs, la complémentarité entre les spécialités ne peut se jouer que de manière séquentielle dans le temps, avec éventuellement des aller-retour. Néanmoins, il est possible de réutiliser certaines parties de spécifications à la manière des bibliothèques des langages de programmation, lorsque les services spécifiés ont été réalisés par ailleurs.

• *L’esthétique* ne peut encore être évaluée objectivement faute d’expérience suffisante sur l’utilisation pratique des extensions proposées. Cependant, le cas d’étude décrit au paragraphe 3.3 peut en donner un aperçu intéressant.

• *L’apprentissage progressif* des extensions proposées est difficile. En effet, elles constituent un tout et doivent donc être apprises en même temps. Cependant, afin de faciliter leur apprentissage, la sémantique des extensions proposées est inspirée de celle déjà utilisée par chacun des formalismes. De plus, celles-ci sont présentées sous la forme de tableaux récapitulant pour chaque type d’interruption la symbologie associée à chaque formalisme (Cf. figure IV-4).

• *L’existence d’outils d’aide à la spécification* reste le plus souvent théorique. MAD possède un éditeur de tâche, et les réseaux de Petri

disposent de vérificateurs et de générateurs de code. Aucune mise à jour de ces outils n'a été réalisée afin de prendre en compte les extensions proposées. Ces outils pourront être réalisés dans l'avenir, à la condition sine qua non que les formalismes se stabilisent. En effet, la réalisation d'outils est coûteuse et ne peut se justifier que si le formalisme sur lequel ils s'appuient évolue plus lentement que le temps nécessaire à la réalisation des dits outils !

• *La capacité à faciliter la détection des erreurs* n'a pas pu être évaluée dans notre cas, car elle nécessite une approche expérimentale. Il est en effet nécessaire d'évaluer sur un ou plusieurs exemples la capacité des concepteurs à détecter une erreur introduite volontairement au sein d'une spécification.

• *La sémantique* des extensions proposées est inspirée de celle déjà utilisée par chacun des formalismes afin de rendre aussi claires que possible les notations utilisées. De plus, l'interprétation de ces extensions est facilitée par leur présentation sous forme de tableaux récapitulatifs. De cette façon, les risques de mauvaise interprétation et d'ambiguïtés sont, nous l'espérons, réduits.

• *La finalité* des extensions proposées vise clairement la spécification des interruptions. Quant à la finalité des formalismes étendus, elle a été définie dans le cadre de ce mémoire en fonction des phases de conception identifiées (Cf. figure VI-1).

• *L'étude de cas* illustrant les extensions proposées est concrétisée par la formalisation du Publiphone, objet du paragraphe 3.3. Cet exemple s'inspire d'une interface dynamique où erreurs et interruptions sont omniprésentes. Sa spécification exige ainsi des formalismes utilisés un pouvoir d'expression représentatif des exigences d'une interface du monde réel.

• *L'adéquation avec la tâche du concepteur* des formalismes utilisés est dépendante du contexte de la spécification. Afin de garantir cette adéquation, nous avons associé à chaque phase du cycle de conception un formalisme qui y est a priori adapté (Cf. figure IV-1). Cependant, notre approche de la conception est fortement dirigée par l'analyse de la tâche. Il est peu probable qu'elle soit adaptée à d'autres types d'approche, comme par exemple celles basées sur l'étude des contraintes temporelles.

• *L'utilité sociale* des formalismes ne peut être évaluée que par une approche expérimentale exigeant un nombre important de sujets constitués en équipes de conception. Elles n'a donc pu être évaluée objectivement dans le cadre de ces travaux de recherche.

Les extensions proposées et les choix effectués satisfont la majorité des Principes de Namur. Cependant il apparaît nécessaire aux extensions proposées de subir l'épreuve du temps. C'est en effet par l'utilisation intensive et la multiplication des exemples basés sur ces extensions que les difficultés émergent. Une fois celles-ci identifiées, une adaptation des extensions peut alors être envisagée. Nous allons maintenant nous intéresser au second aspect évoqué dans l'introduction de ce chapitre : l'utilisation des traducteurs entre formalismes.

3. Traducteurs

3.1. Approches de conception

L'usage complémentaire de plusieurs formalismes permet la spécification d'un système par affinements successifs. De ce point de vue, l'utilisation des traducteurs peut se voir selon deux approches :

• Ils peuvent êtres destinés à la transformation complète de la spécification d'un système d'un formalisme à un autre, de manière à faire la liaison entre les différentes phases de conception. Charge ensuite aux concepteurs d'affiner la spécification résultante en fonction des besoins de la phase de conception où ils interviennent. Cette démarche permet d'obtenir au final une spécification complète en réseaux de Petri. Cette dernière spécification risque d'être en conséquence volumineuse et probablement difficilement interprétable, donc inutile, pour un programmeur. Cependant, cette dernière étape peut aussi être le point de départ d'une génération automatique de preuves, ou de code exécutable. Elle s'apparente alors à un processus de compilation dont le code exécutable, la spécification sous la forme de réseaux de Petri, n'a besoin d'être lisible que par une machine.

• Les concepteurs peuvent par ailleurs être amenés à tirer parti de la complémentarité des formalismes, et n'utiliser un formalisme que pour ce qu'il sait bien spécifier. L'ensemble des spécifications résultant de ce processus ressemble aux boîtes gigognes, où "l'ouverture" d'une spécification en laisse découvrir d'autres, décrivant quelques points

supplémentaires détaillant la spécification englobante. Cette approche ne peut néanmoins se passer des traducteurs car, comme le montre la figure VI-1, les formalismes ont des capacités qui se superposent. Il est alors parfois nécessaire de réaliser les traductions partielles des spécifications précédentes.

Dans ces deux cas de figure, la conception d'une interface ne peut pas raisonnablement se voir comme un processus linéaire du projet vers la réalisation. De nombreux retours arrière sont souvent nécessaires, suite à des erreurs, ou à des problèmes de conception découverts tardivement. Il est alors nécessaire de remonter les problèmes identifiés vers la phase de conception précédente. Les traducteurs peuvent alors être utilisés, en sens inverse, afin de participer à au processus de rétroconception[16]. Cependant, ces retours arrières sont beaucoup plus difficiles à réaliser que les affinements. Autant les relations temporelles entre MAD et UAN peuvent être considérées comme quasiment équivalentes, autant le passage de UAN vers les réseaux de Petri s'apparente à une production de code non réversible. Il est néanmoins possible d'identifier dans une description en réseaux de Petri les structures remarquables proposées au sein des tableaux récapitulatifs du paragraphe 3.2, mais cette reconnaissance tient plus de l'art divinatoire que d'un processus automatisé. De plus, les capacités des formalismes n'étant pas équivalentes, une perte d'information non négligeable peut avoir lieu. Par exemple, le passage de UAN vers MAD ne peut se fait qu'au prix d'une perte de la spécification des retours d'informations présents dans la description UAN. Ces problèmes identifiés, nous allons maintenant passer à la description précise des traducteurs évoqués.

16 Traduction française de "Reverse Engineering".

3.2. Tableaux de traduction

La réalisation de ces tableaux récapitulatifs s'appuie sur une contribution importante de Palanque, Bastides, et Senges [Palanque et al. 1995]. Les auteurs proposent le principe de l'utilisation de UAN et des réseaux de Petri et présentent une grande partie des équivalences entre UAN et les réseaux de Petri que nous décrivons ici. Cependant, les résultats concernant les interruptions et l'entrelacement des tâches ne nous ont pas paru satisfaisants, c'est pourquoi nous les avons redéfinis.

Les tableaux récapitulatifs figure VI-5 sont dans leur principe identiques à ceux précédemment évoqués concernant les interruptions. En outre, certains des traducteurs ont nécessité l'usage des extensions des formalismes : MAD* [Hamouche 1995] au lieu de MAD, ainsi que les extensions aux réseaux de Petri proposées par Sifakis [Sifakis 1977]. Dans ce cas une indication est apportée dans la case correspondant au formalisme.

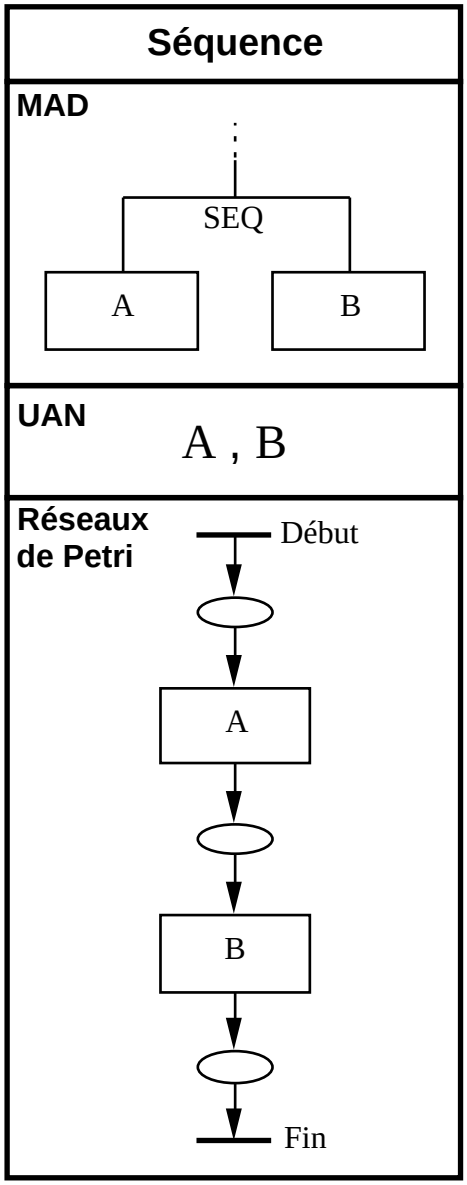

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (1/10).

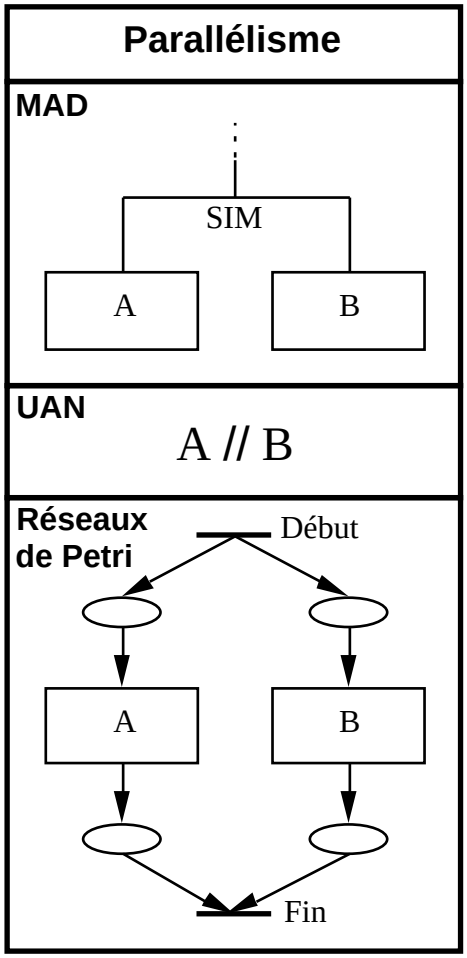

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (2/10).

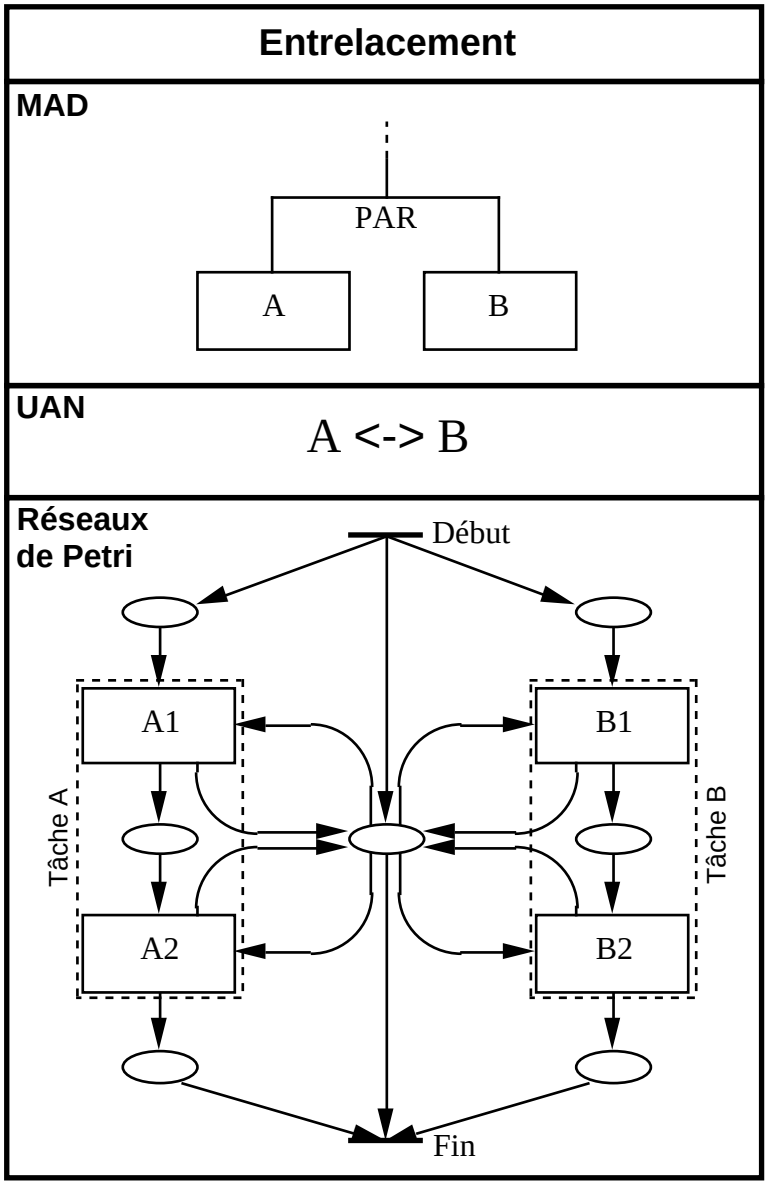

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (3/10).

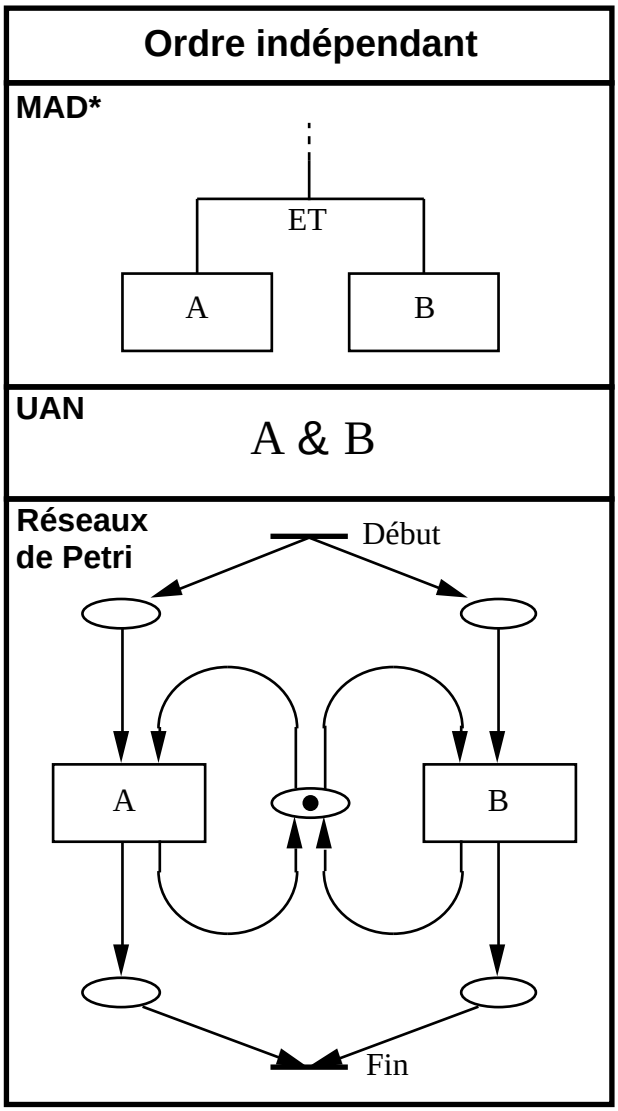

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (4/10).

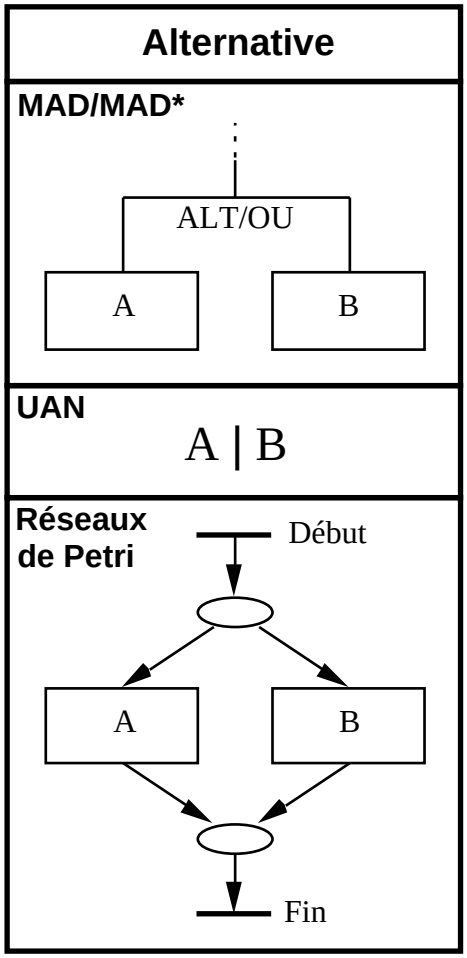

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (5/10).

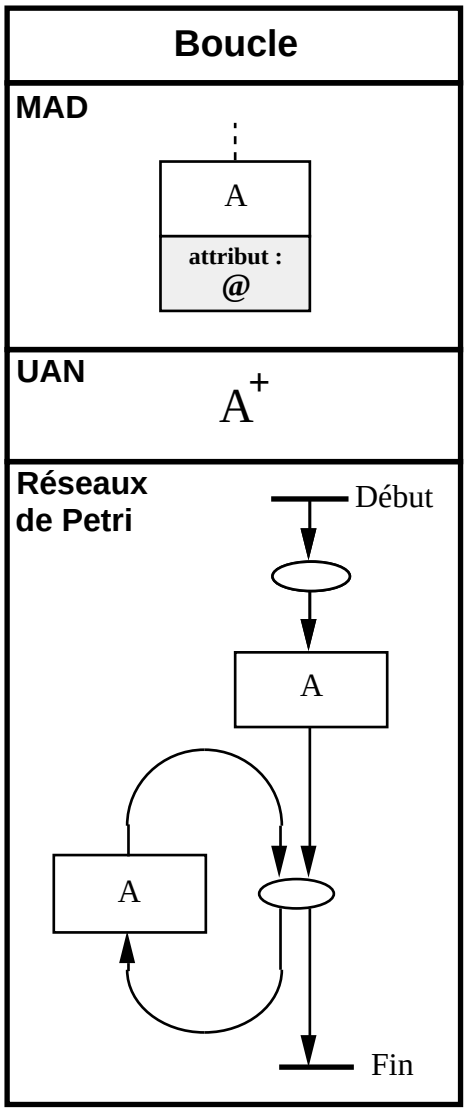

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (6/10).

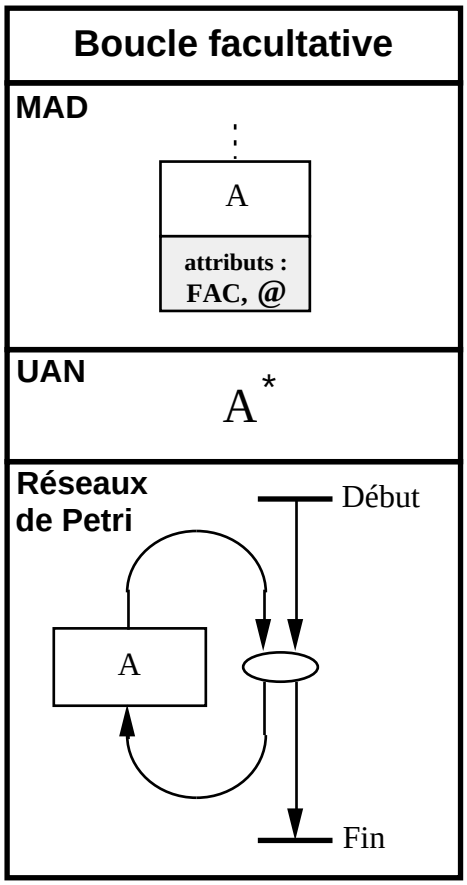

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (7/10).

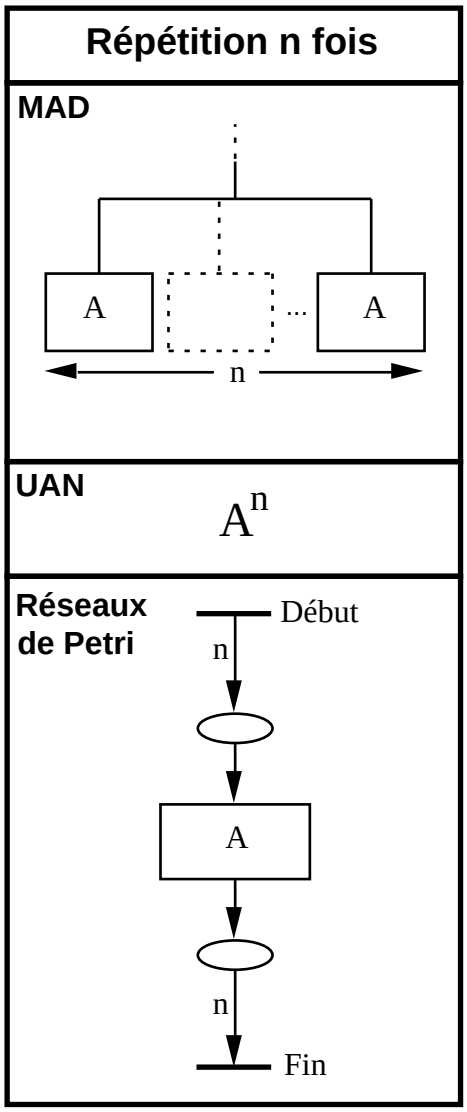

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (8/10).

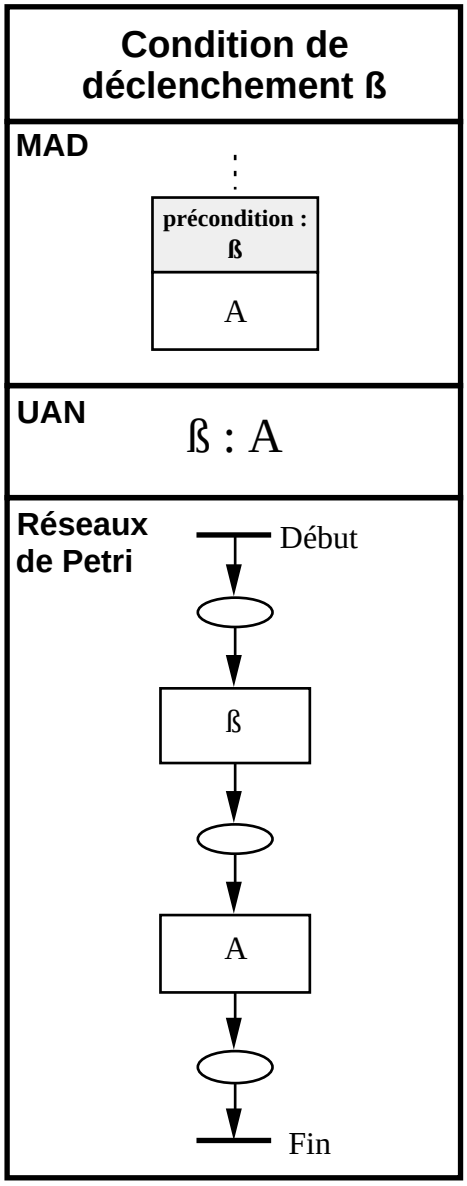

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (9/10).

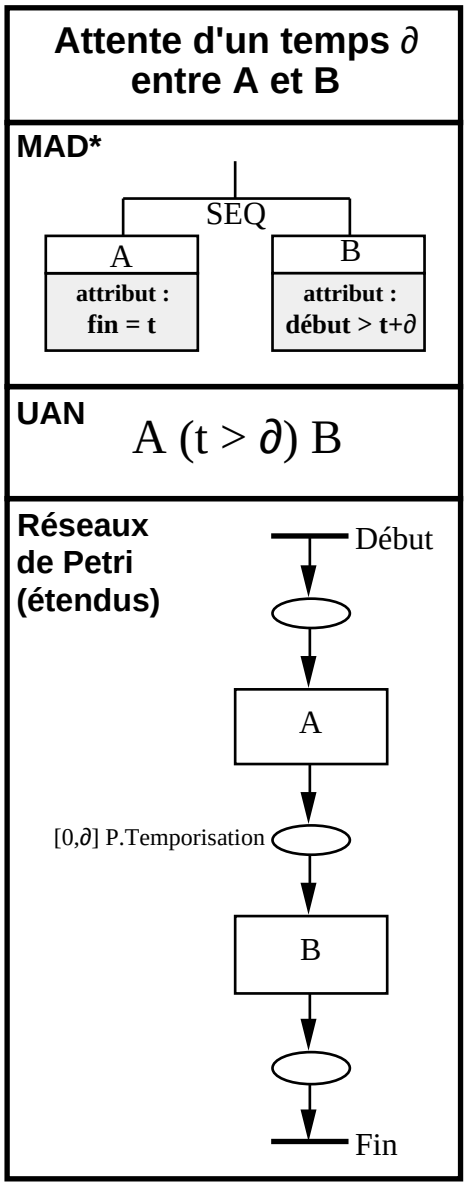

Figure VI-5 : Tableaux récapitulatifs des relations temporelles (10/10).

3.3. Exemple du Publiphone

Ce paragraphe reprend l'exemple du Publiphone évoqué au chapitre III. Les figures VI-6 et VI-7 détaillent cette spécification utilisant de manière complémentaire MAD, UAN et les réseaux de Petri. Nous avons mis à contribution chacun des trois formalismes cités afin de résoudre les problèmes typiques des trois phases de conception évoquées au début de ce chapitre : MAD offre une notation support à l'analyse des besoins, UAN permet la conception de l'interface, tandis que les réseaux de Petri détaillent quelques problèmes de temporisation que nous avons découverts.

Les figures suivantes se lisent de gauche à droite et de bas en haut, la figure VI-6 contient les descriptions selon MAD et UAN, tandis que la figure VI-7 reprend la description selon UAN et en détaille certains aspects à l'aide des réseaux de Petri. Le lecteur est en outre invité à se munir d'une feuille de papier vierge de manière à révéler progressivement les spécifications.

En découvrant la spécification du Publiphone selon MAD figure VI-6, on y découvre les actions de haut niveau effectuées afin de téléphoner. La partie de droite de cette même figure révèle les retours d'information en provenance du Publiphone ainsi que les conditions terminant brutalement la tâche globale. Cela se produit, par exemple, lorsque l'utilisateur attend plus de dix secondes avant de remettre une nouvelle carte lors du changement de carte.

La figure VI-7, quant à elle, détaille à l'aide des réseaux de Petri deux tâches posant problème. La première permet de rendre plus lisible le risque de terminer prématurément la tâche téléphoner lors du changement de carte évoqué précédemment, tandis que la seconde décrit le système permettant de ne pas oublier sa carte par mégarde. En effet, le Publiphone fait retentir une alarme sonore si l'utilisateur oublie de retirer sa carte après avoir raccroché.

Cet affinement incrémental permet aux concepteurs de sérier les problèmes de conception étape par étape. Il permet également par cette succession d'étapes d'inciter les concepteurs, nous l'espérons, à se poser les bonnes questions à l'étape où ils interviennent.

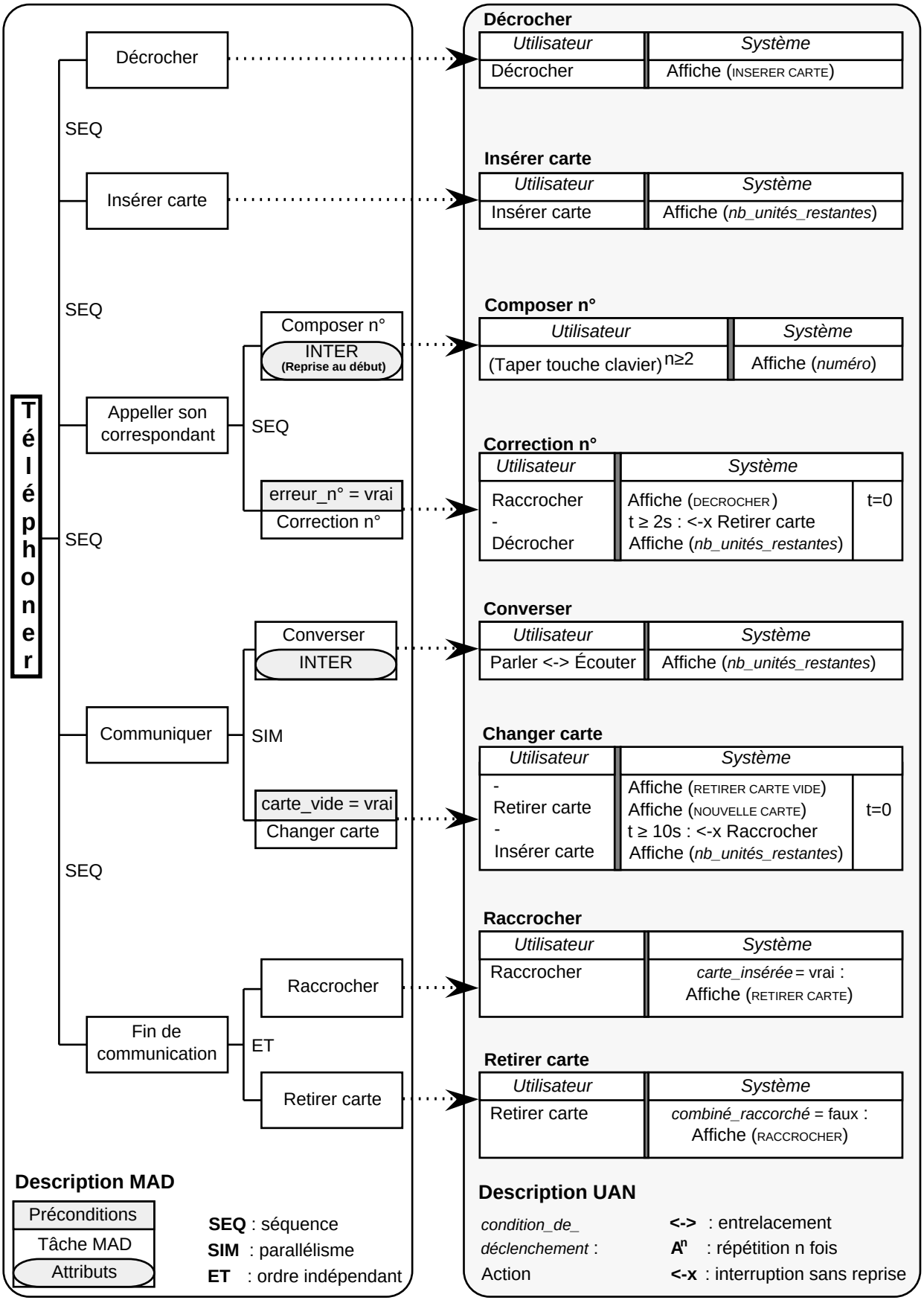

Figure VI-6 : Description de la tâche téléphoner selon MAD et UAN.

Décrocher

Utilisateur	*Système*
Décrocher	Affiche (INSERER CARTE)

Insérer carte

Utilisateur	*Système*
Insérer carte	Affiche (*nb_unités_restantes*)

Composer n°

Utilisateur	*Système*
(Taper touche clavier)$^{n\geq 2}$	Affiche (*numéro*)

Correction n°

Utilisateur	*Système*	
Raccrocher - Décrocher	Affiche (DECROCHER) t ≥ 2s : <-x Retirer carte Affiche (*nb_unités_restantes*)	t=0

Converser

Utilisateur	*Système*
Parler <-> Écouter	Affiche (*nb_unités_restantes*)

Changer carte

Utilisateur	*Système*	
- Retirer carte - Insérer carte	Affiche (RETIRER CARTE VIDE) Affiche (NOUVELLE CARTE) t ≥ 10s : <-x Raccrocher Affiche (*nb_unités_restantes*)	t=0

Raccrocher

Utilisateur	*Système*
Raccrocher	*carte_insérée* = vrai : Affiche (RETIRER CARTE)

Retirer carte

Utilisateur	*Système*
Retirer carte	*combiné_raccorché* = faux : Affiche (RACCROCHER)

Description UAN

condition_de_déclenchement : Action

<-> : entrelacement
A^n : répétition n fois
<-x : interruption sans reprise

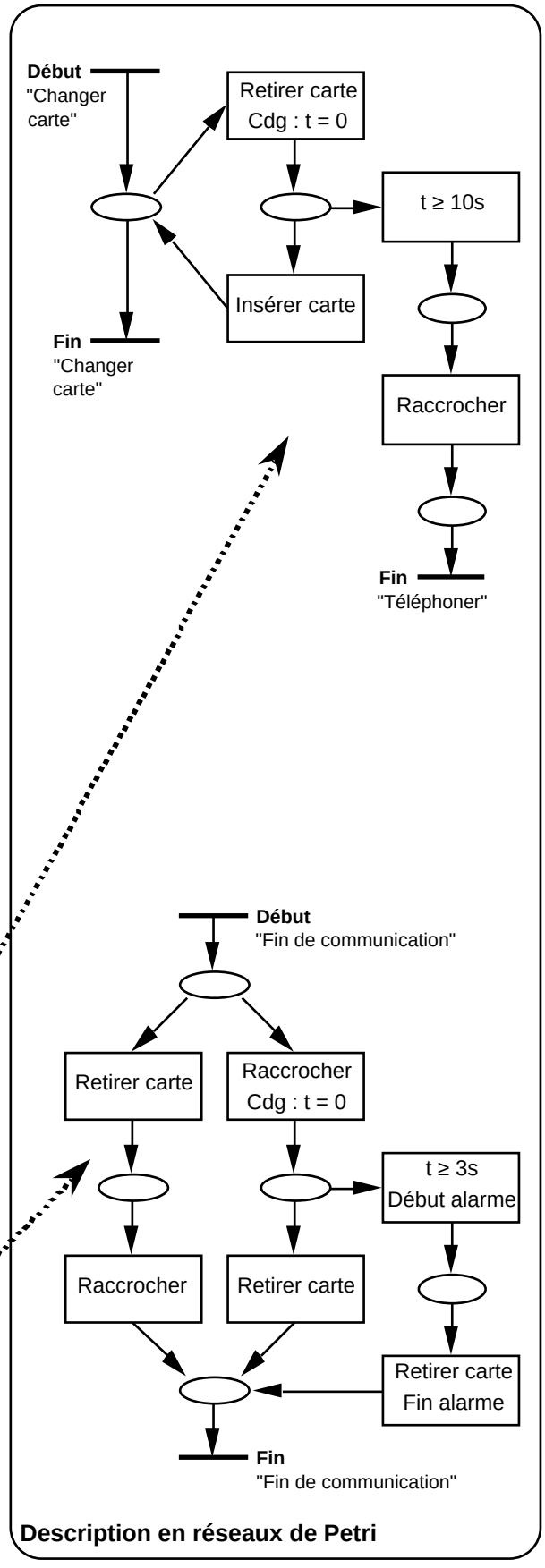

Figure VI-7 : Description de la tâche téléphoner selon UAN et les réseaux de Petri.

4. Conclusion

Ce chapitre rassemble les applications pratiques des concepts dégagés au cours des trois chapitres précédents. Il met en lumière la possibilité de spécifier les différentes facettes de la notion d'interruption, et de la transcrire à l'aide de trois formalismes différents et complémentaires. L'usage des traducteurs proposés peut en outre facilement s'automatiser depuis MAD vers les réseaux de Petri. Par contre, la transformation inverse est beaucoup plus hasardeuse. Ceci pose les limites des traducteurs décrits.

Outre ces limitations, les extensions et traducteurs ont besoin d'être appliqués à la spécification d'un système plus conséquent que l'exemple de travail du Publiphone, afin d'en détecter les limitations opérationnelles. C'est pourquoi nous avons choisi de concevoir selon la démarche proposée un système d'édition de trajectoires d'aéronef. Ce système fait l'objet de la maquette CoTASS décrite en annexe de ce mémoire.

5. Bibliographie

[Accot et al. 1996] Accot J., Chatty S., & Palanque P. A formal description of low level interaction and its application to multimodal interactive systems. *Third International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'96), Namur, Belgium*, 5-7 June 1996.

[Barthet 1988] Barthet M.-F. *Logiciels interactifs et ergonomie : Modèles et méthodes de conception*. Paris, France : Dunod, 1988.

[Gray et al. 1994] Gray P., England D., & McGowan S. *XUAN: Enhancing the UAN to capture temporal relation among actions*. Department of Computing Science, University of Glasgow, February 1994. Department research report IS-94-02.

[Hamouche 1995] Hamouche H. *De la modélisation des tâches utilisateurs à la spécification conceptuelle et sémantique d'interfaces homme-machine.* Thèse en informatique : Université Paris VI, Décembre 1995.

[Hartson et al. 1990] 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, p. 181-203.

[Hartson et al. 1992] Hartson R.H. & Gray P.D. *Temporal aspects of tasks in the User Action Notation*. Human Computer Interaction, Lawrence Erlbaum Associates, 1992. p. 1-45.

[Johnson 1996] Johnson C. The Namur principles: criteria for the evaluation of user interface notations. *Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'96), Namur, Belgique*, 1996.

[McGowan 1995] McGowan S. *From UAN to eXUAN: Specifying simulations of the temporal properties of interaction*. University of Glasgow, Department of Computing Science, March 1995. Technical report TR-1995-5.

[Palanque 1992] Palanque P. *Modélisation par objets coopératifs d'interfaces homme-machine dirigées par l'utilisateur*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1992.

[Palanque et al. 1995] Palanque P., Bastide R., & Senges V. Task model - system model: towards an unifying formalism. *HCI International conference, Yokoohama, Japan*, 9-14 July 1995. p. 489-494.

[Peterson 1981] Peterson J.L. *Petri net theory and modelling of systems*. Prentice Hall, 1981.

[Scapin et al. 1989] Scapin D.L. & Pierret-Golbreich C. MAD : Une méthode analytique de description des tâches. *Colloque sur l'ingénierie des Interfaces Homme-Machine (IHM'89), Sophia-Antipolis, France*, Mai 1989. p. 131-148.

[Sebillotte et al. 1994] Sebillotte S., Alonso B., Fallah D., Hamouche H., & Scapin D.L. *Note de recherche concernant le formalisme MAD*. INRIA Rocquencourt, Novembre 1994. Document interne.

[Sifakis 1977] Sifakis J. *Use of Petri nets for performance evaluation*. Measuring, Modelling, and Evaluating Computer Systems, North Holland, 1977.

CONCLUSION

Il est maintenant temps de clore notre réflexion. Cette conclusion s'articule autour de trois thèmes. La première section résume les contributions de ce mémoire, tandis que la deuxième en dresse les limites. Enfin, la troisième et dernière section de cette conclusion propose de nouvelles perspectives de recherche.

1. Contributions

Notre démarche de recherche a trait à la description des interruptions et des erreurs humaines au sein du processus de conception des interfaces homme-machine. Les contributions de ce travail de recherche se scindent en deux thèmes : les concepts (§ 1.1) et les formalismes (§ 1.2). Nous avons de plus illustré ces contributions par un exemple d'implémentation (§ 1.3).

1.1. Concepts

Nous avons tout d'abord proposé, à l'intention du concepteur, une aide au choix des mécanismes de correction d'erreur (§ II-3.3). Cette aide permet de représenter selon deux dimensions d'analyse les deux types de correction, la correction arrière et la correction avant, en fonction du coût pour l'utilisateur et de l'état désiré du système. De plus, l'évolution dans le temps de la disponibilité des mécanismes de correction d'erreur peut être mise en lumière par l'adjonction d'une troisième dimension d'analyse.

Nous avons aussi introduit la notion de "singularité" qui étend notamment celle d'interruption en permettant le parallélisme et l'entrelacement entre la tâche interrompante et la tache interrompue. Cette notion englobe également les corrections d'erreur qui, nous l'avons montré, possèdent un sous-ensemble commun avec les interruptions. Par ailleurs, une modélisation de ces singularités est proposée (§ II-4.2). Cette modélisation décompose une singularité en trois tâches séquentielles : le prologue, le corps de l'interruption, et l'épilogue. Les étapes importantes du prologue et de l'épilogue d'une singularité ont été détaillées.

Les liens entre erreurs et interruptions ont été mis en exergue (§ II-4.3). En effet, une erreur se traduit lors de sa correction par une interruption de la tâche en cours. De même, l'interruption d'une tâche peut provoquer une erreur de l'utilisateur. Notons cependant que de notre point de vue, une erreur

ne peut être représentée que par la procédure de correction qui lui est associée.

1.2. Formalismes

Le chapitre III présente une méta-classification des formalismes de la littérature (§ III-2). Cette méta-classification distingue quatre types de classifications : selon le pouvoir d'expression, l'utilisabilité, l'origine, ou la finalité des formalismes. En corollaire, une méthode d'usage de ces classifications est proposée dans le but de guider le concepteur dans le choix d'un formalisme adapté à ses besoins (§ III-3).

En application des concepts proposés au chapitre II, des extensions aux formalismes MAD, UAN, et aux réseaux de Petri sont définies (§ VI-2). Ce chapitre propose également une dizaine de tableaux récapitulant les équivalences entre MAD, UAN, et les réseaux de Petri (§ VI-3). Ils ont pour but de servir de référence à l'usage complémentaire des formalismes lors du processus de spécification d'une interface homme-machine.

1.3. Exemple d'application

Afin de permettre au concepteur d'avoir une vue globale de l'utilisation des contributions proposées, une maquette informatique a été réalisée. Cette maquette, CoTASS, simule un éditeur de trajectoire d'aéronef. Elle est décrite dans l'annexe de ce mémoire.

2. Limites

Les contributions résumées à la section précédente souffrent néanmoins de quelques limites :

Les traducteurs proposés sont entravés par l'absence d'outils automatisés permettant de transcrire une spécification d'un formalisme vers l'autre. La réalisation de tels outils est une tâche conséquente qui n'a put être réalisée dans le cadre de ce mémoire. De plus, les traductions ne s'effectuent systématiquement que dans un seul sens, à la façon d'un processus de compilation. En particulier, la rétro-traduction est difficile des réseaux de Petri vers UAN.

En outre, la description de l'interruption avec entrelacement est problématique avec les réseaux de Petri. Elle n'est valide que si l'interruption

et la tâche interrompue peuvent se décomposer en tâches purement séquentielles. Ce problème est assez fondamental : l'entrelacement entre deux tâches peut en effet se voir comme une série de mutuelles interruptions entre les deux tâches avec reprise en cours.

Enfin, une dernière étape reste à franchir, celle de l'implémentation des singularités. L'application CoTASS, décrite en annexe, nous donne un premier exemple d'une telle implémentation. Cependant nous manquons encore de modèles d'architecture permettant de faire la liaison entre les spécifications des singularités, et leur réalisation pratique au sein de l'interface homme-machine.

3. Perspectives

Après avoir évoqué les contributions de ce mémoire, et les limitations de celles-ci, détaillons maintenant de nouvelles perspectives de recherche :

En premier lieu, comme le souligne l'une des limitations évoquées, les contributions de ce travail ne permettent que la description des interruptions et des mécanismes de correction d'erreur. Elles manquent de modèles d'implémentation. L'étude de ces modèles constitue un premier axe de recherche possible, et complémentaire au travail réalisé.

Les études ethnographiques évoquées au chapitre II ont montré l'importance des interruptions dans le cadre du travail coopératif. En outre, les systèmes critiques tendent à devenir de plus en plus coopératifs en particulier par l'utilisation des liaisons de données. Le collecticiel est un axe de recherche où les problèmes de formalisation sont pour l'instant non résolus [Jambon & Karsenty 1994]. C'est pourtant un axe de recherche peu étudié, où les difficultés semblent innombrables.

ANNEXE

APPLICATION Co.T.A.-S.-S.

1. Introduction

1.1. Motivations

Nous avons successivement évoqué la notion de singularité au chapitre II, le choix de trois formalismes correspondant à nos besoins au chapitre III, puis leurs extensions et traducteurs au chapitre IV. Il convient maintenant de montrer que l'utilisation pratique de ces contributions est possible. Pour cela, la réalisation d'une maquette informatique est envisagée. Plusieurs possibilités de réalisations s'offrent à nous :

• Il est envisageable de réaliser un outil informatique permettant l'édition d'une spécification selon les formalismes choisis et leurs extensions. Cet outil doit également permettre la traduction automatisée des spécifications entre ces formalismes. La réalisation d'un tel outil permet de monter que les formalismes utilisés peuvent s'écrire et se traduire l'un vers l'autre. Cependant, elle ne montre en aucune manière que les formalismes de spécification sont applicables à un cas réel. En effet, la réalisation d'un traitement de texte permettant d'écrire un document selon un alphabet particulier n'a jamais prouvé qu'une langue pouvait se baser sur cet alphabet. De plus, les extensions des formalismes proposées n'ont jamais été testées en vraie grandeur. C'est pourquoi de nombreuses modifications des formalismes sont à prévoir, rendant l'éditeur rapidement obsolète. Cette possibilité a donc été écartée.

• Il peut être alors jugé intéressant d'effectuer la spécification d'une application déjà existante, un cas d'école, dont on dispose de nombreux exemples de réalisation et de spécification. Les éditeurs destinés au courrier électronique, considérés comme la "souris blanche" de l'ingénierie des interfaces homme-machine, peuvent être de bons candidats. Cette démarche est très intéressante lors de la conception d'un formalisme, car les pierres d'achoppement des autres formalismes sont connues. Il est ainsi possible de faire progresser le premier sur ces limitations. Cependant, lors de l'utilisation d'un tel cas d'école, la majorité des problèmes potentiels de conception sont déjà connus par celui qui effectue la spécification. Ce n'est alors plus l'utilisabilité du formalisme qui est évaluée, mais plutôt l'expertise de celui qui l'utilise. La démarche est donc biaisée et ne montre pas qu'un formalisme peut s'appliquer à un cas réel de conception.

• Il est donc nécessaire d'envisager la spécification puis la réalisation d'une application nouvelle dont il n'existe que peu ou pas de précédent. La démarche du concepteur n'est ainsi plus biaisée par les résultats des études antérieures, il se trouve alors dans une situation proche de l'utilisation réelle d'un formalisme de spécification. Une telle démarche permet de révéler les problèmes des formalismes et de leurs extensions au cours de leur utilisation réelle. Ces problèmes sont donc susceptibles d'apparaître au cours de la spécification, mais aussi être détectés bien après la réalisation effective de l'application, lors de son évaluation. Cette démarche comporte le risque de remettre en cause à tout moment les extensions proposées. Cependant elle est nécessaire afin d'évaluer en toute honnêteté les extensions proposées. Nous avons donc choisi de l'appliquer par l'intermédiaire de la maquette CoTASS décrite ci-après.

1.2. La maquette CoTASS

CoTASS est une maquette informatique simulant une partie des systèmes de navigation d'un aéronef de transport civil. Ce projet prévoit deux phases, dont seule la première sera réalisée dans le cadre de ce mémoire. Cette première phase est destinée à évaluer la prise en compte des erreurs et interruptions au cours du processus de conception. La seconde phase est destinée à explorer les problèmes de spécifications propres au travail collaboratif et fera l'objet de recherches futures.

• *La première phase* est destinée à prouver la faisabilité d'un système d'édition de plan de vol embarqué. Dans notre scénario, la tâche principale du pilote est de modifier le plan de vol de son aéronef en accord avec les instructions du contrôle aérien ou de sa propre initiative. Le pilote est susceptible de faire des erreurs et d'être interrompu par des alarmes en provenance des systèmes.

• *La seconde phase* ajoute à la phase précédente la possibilité d'échange de trajectoires entre l'aéronef et une ou plusieurs positions de contrôle aérien. L'édition de trajectoire s'effectue alors de manière collaborative par échange de propositions. Cette phase s'inspire de recherches déjà effectuées du côté aéronef [Jambon & Coutaz 1993] et du côté contrôle aérien par Stéphane Chatty et François-Régis Colin au CENA [Chatty & Colin 1994]. Cette seconde phase a inspiré la dénomination du projet : l'acronyme CoTASS signifie Coordination Tripartite Air-Sol-Sol.

Cette réalisation s'inspire également, à des degrés divers, des nombreuses maquettes déjà réalisées comme par exemple celles décrites dans [Chatty & Lecoanet 1996 ; Cottenceau 1993 ; Knox & Scalon 1991 ; L'Ebraly 1994 ; Renou 1993]. De même, l'étude d'une partie de la volumineuse bibliographie technique sur le sujet nous a apporté de nombreuses idées [Airbus Industrie 1992 ; Boeing 1995 ; Casaux 1995 ; Combes 1993 ; Dupont 1994 ; Dupont 1995 ; Garmin 1995 ; Haertl 1995 ; Monzel & Bories 1993 ; Perry 1991 ; Pfuhl, Greving, & Mandelka 1993 ; SIA 1991 ; Sylla 1993 ; Waller 1992].

1.3. Fonctions du simulateur

L'aéronef simulé est un avion de ligne moderne du type "glass cockpit". Le moteur de simulation n'étant pas l'objet de cette étude, la partie simulation aérodynamique est limitée à sa plus simple expression. Cet avion est supposé toujours connaître exactement sa position en latitude, longitude et altitude par exemple grâce à un système de positionnement par satellite (GPS)[17]. Il est piloté uniquement à partir de son système de gestion de vol (FMGS), c'est-à-dire qu'aucun dispositif de pilotage manuel ou semi-manuel n'est disponible. De plus, de guidage n'est réalisé dans un premier temps que dans le plan horizontal. En corollaire, la gestion de la puissance des moteurs est supposée toujours automatique sous le contrôle de l'automanette. L'avion est supposé suivre un plan de vol défini avant le décollage et présent dans le FMGS. Le pilote peut modifier le plan de vol de son avion à tout moment au cours du vol. Pour cela, il dispose d'une interface essentiellement graphique limitée à seul écran (ND). En cela, l'interface proposée prend le contre-pied des interfaces actuelles basées sur l'utilisation exclusive d'un terminal composé d'un clavier abécédaire et d'un écran textuel (MCDU) [Airbus Industrie 1992] §4.04.20. Cette interface tolère les erreurs de manipulation et dispose de la fonction "défaire". De plus, la tâche du pilote doit pouvoir être interrompue par des alarmes en provenance des systèmes.

Cet éditeur de trajectoire ne doit pas être considéré comme un outil de dessin simplifié. En effet, les contraintes sur cet outil sont fortes :

- Le système est dynamique, car au cours de l'édition de plan de vol l'avion continue de voler en suivant le plan de vol actif. En conséquence, le contexte de l'éditeur change continûment.

[17] Les acronymes et termes techniques utilisés dans cette annexe sont définis à la fin de celle-ci.

- Le pilote doit pouvoir suspendre son édition de plan de vol a tout moment pour répondre à une interruption, puis y revenir. L'édition ne doit donc pas bloquer l'écran de navigation (ND) dont le pilote peut avoir un besoin immédiat et vital.
- De nombreuses fausses manipulations sont à prévoir, elle peuvent être très fréquentes si l'avion rencontre des turbulences.

Ces contraintes énumérées, nous allons maintenant nous intéresser plus précisément à l'analyse de la tâche du pilote.

2. Analyse de tâche

L'analyse de tâche proposée débute par une étude des scénarii d'utilisation possibles. Vient ensuite une extraction des tâches pertinentes puis une description MAD des tâches identifiées. Nous avons en ce sens suivi une démarche ergonomique simplifiée qui s'inspire de celle proposée avec le formalisme MAD [Sebillotte 1994]. Seule la première phase du projet est spécifiée et réalisée, il n'y a donc aucun échange de données possibles avec le contrôle aérien.

Nous avons fait appel à MAD afin de rédiger les résultats de l'analyse de tâche. Puis nous avons utilisé UAN dans le but de décrire plus précisément l'interface. Enfin, les réseaux de Petri ont été utilisées ponctuellement afin de lever les ambiguïtés au sein de cette dernière spécification. Nous avons également eu recours à une maquette papier afin de représenter le panneau de contrôle du système, nous en transcrivons ici l'une des propositions.

2.1. Scénarii possibles

L'étude de la bibliographie évoquée au paragraphe 1.2 permet la constitution de plusieurs scénarii d'utilisation de l'éditeur de trajectoire :

- *Navigation normale* : Le pilote dispose d'un plan de vol actif. Il l'étudie en modifiant son échelle et/ou le point de vue.

- *Modification de la navigation* : Le pilote modifie son plan de vol actif sur demande par radiotéléphonie du contrôle aérien, ou de sa propre initiative. Cette modification est immédiatement prise en compte par le FMGS.

- *Préparation d'un plan de vol secondaire* : Le pilote prend l'initiative de faire une copie du plan de vol actif, pour le modifier selon de nouveaux

objectifs. Ce plan de vol sera ensuite éventuellement activé à la place du plan de vol actif dans le FMGS.

Remarquons que ces scénarii décrivent une interaction où corrections d'erreur et interruptions sont absentes. Cet oubli est volontaire car leur prise en compte au niveau des scénarii rendrait la lecture de ceux-ci complexe. Ces singularités ne sont évoquées que lors de l'identification des tâches ci-après.

2.2. Extraction des tâches

L'étude comparative de ces scénarii permet de dresser la liste des tâches réalisées par le pilote lors de l'utilisation de l'éditeur de trajectoire :

- Consulter un plan de vol
- Comparer un plan de vol avec le plan de vol actif
- Modifier un plan de vol
- Copier un plan de vol

Ces tâches définissent implicitement la notion de plan de vol dont nous pouvons énumérer les deux intenses :

- Plan de vol actif
- Plan de vol secondaire

L'ensemble de ces tâches peut se scinder en deux catégories. La première, la *visualisation*, laisse intactes les données des plans de vol. La seconde au contraire, la *modification*, permet au pilote de remanier les plans de vol du FMGS. En outre, à chacune de ces catégories correspond un comportement différent vis-à-vis des singularités :

- La *visualisation* regroupe les tâches de consultation d'un plan de vol et de comparaison d'un plan de vol avec le plan de vol actif. Nous avons ajouté à ces deux tâches les tâches articulatoires de changement d'échelle et de modification du point de vue (centré aéronef ou non) qui permettent au pilote d'adapter la visualisation d'un plan de vol à ses besoins.

  Toutes ces tâches sont réversibles, un changement d'échelle inopportun peut par exemple être corrigé par un changement d'échelle dans le sens contraire. En conséquence, la correction d'erreur pour cette catégorie de tâches est inclue dans les tâches elles-mêmes.

  En cas d'interruption, le pilote doit pouvoir placer rapidement son écran de visualisation (ND) dans un état standard lui permettant d'appréhender la

navigation courante. Cet état standard a été défini : plan de vol actif seul centré sur l'aéronef et à une petite échelle. L'ensemble des tâches permettant d'atteindre l'état standard constituent le prologue de l'interruption qui doit être défini. Comme les tâches sont réversibles, l'épilogue se construit en utilisant les tâches duales déjà utilisées lors de la correction d'erreur. Par sécurité, nous avons choisi de maintenir le plan de vol actif toujours affiché.

• La *modification* regroupe les tâches de copie d'un plan de vol vers un autre plan de vol et de modification d'un plan de vol. En outre, l'ensemble des tâches de visualisation d'un plan de vol précédemment évoquées doivent être disponibles lors de la modification.

La correction des erreurs est dans cette catégorie plus complexe que précédemment. En effet, les actions ne sont pas réversibles par exécution d'une tâche contraire simple. La correction avant est difficile. Par exemple la copie du plan de vol actif vers la plan de vol secondaire écrase la totalité des données de ce plan de vol secondaire. En conséquence, nous avons choisi de réaliser une fonction de correction arrière "défaire" limitée à la dernière action du pilote.

En cas d'interruption, le pilote doit pouvoir arrêter immédiatement l'édition de trajectoire en cours. En outre, ce prologue doit être compatible avec celui défini pour les tâches de visualisation car celles-ci sont inclues dans les tâches de modification. De manière inverse, à la fin de l'interruption, un épilogue doit permettre au pilote de reprendre sa tâche de modification.

2.3. Interruptions du système

Le système peut prendre l'initiative de notifier au pilote ses erreurs :

• Le système peut changer la couleur des branches d'un plan de vol en cours d'édition suite à une modification de trajectoire entraînant l'aéronef à entrer en collision avec le sol (GPWS). Cette interruption notifie simplement au pilote son erreur, sans effets de bords.

• Le système peut être amené à interrompre l'édition de trajectoire si le point d'origine de la modification vient d'être survolé par l'aéronef. Il n'est en effet pas possible de modifier une branche sur lequel l'aéronef s'est déjà engagé. Cette interruption tue la tâche en cours du pilote et lui fait recommencer son édition de branche au début.

2.4. Description MAD

La description de l'interface selon le formalisme MAD permet d'obtenir une vue globale de l'interaction comme le montre la figure A-1.

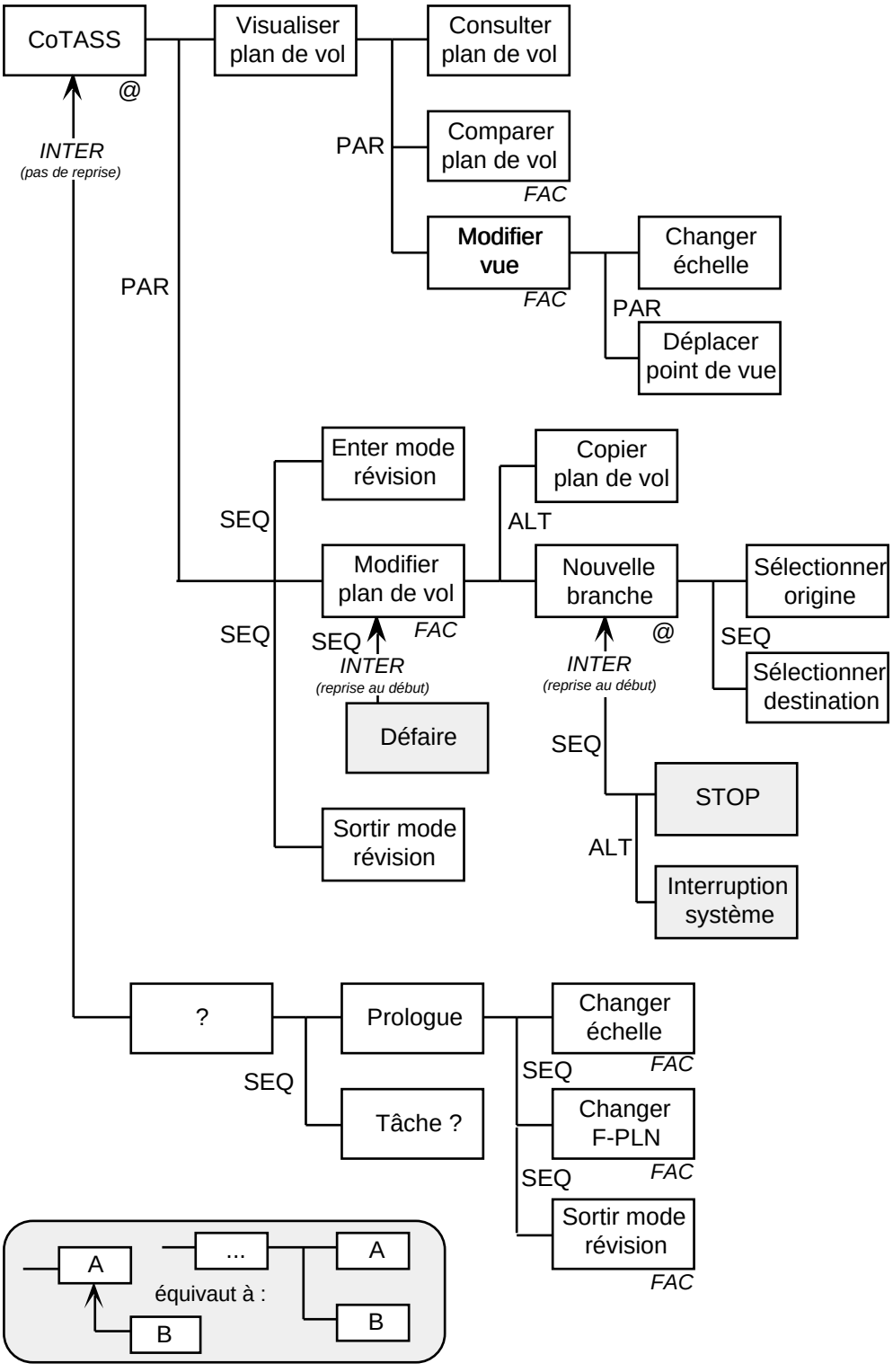

Figure A-1 : Description MAD de la maquette CoTASS.

2.5. Description UAN

Nous avons utilisé la notation UAN afin de préciser le comportement de l'interface lors de l'édition d'une nouvelle branche. En effet cette édition fait appel à un réticule qui suit la souris lors de ces mouvements dès que le mode ‹‹révision›› est engagé. De plus, une ligne élastique apparaît, dès que l'origine de la modification de trajectoire a été sélectionnée, entre ce point d'origine et la souris. La figure A-2 reproduit la spécification UAN de l'entrée et de la sortie du mode ‹‹révision››, ainsi que l'édition de trajectoire elle-même.

Tâche : Entrée mode «révision»

Actions utilisateur	Retours d'information	État interne
~[bouton_revision]		
souris v^	Display (réticule)	mode = «révision»

Tâche : Nouvelle branche

Actions utilisateur	Retours d'information	État interne
~[pt-origine]	réticule > ~	
souris v^	Display (ligne)	
~[pt-destination]	réticule > ~	
souris v^	Erase (ligne)	Mise à jour (nouvelle_branche)

Tâche : Sortie mode «révision»

Actions utilisateur	Retours d'information	État interne
~[bouton_revision]	réticule > ~	
souris v^	Erase (réticule)	mode = «consultation»

Figure A-2 : Description UAN des tâches d'édition de la trajectoire.

2.6. Description en réseaux de Petri

Les réseaux de Petri permettent de décrire précisément certains cas d'interaction où les cas d'interruption sont multiples, comme l'illustre la figure A-3.

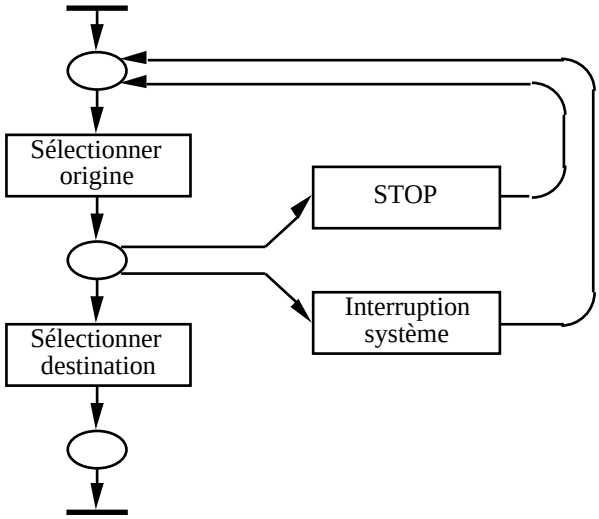

Figure A-3 : Description en réseaux de Petri des deux types d'interruption possibles lors de la création d'une nouvelle branche.

2.7. Maquette papier

Nous avons maintenant spécifié l'ensemble des tâches utiles à l'interaction. Les formalismes utilisés ne nous permettent pas d'éviter la réalisation de maquettes papier en complément des spécifications des actions. La figure A-4 reproduit la proposition d'interface de contrôle prévue initialement. Cette interface s'inspire des systèmes de contrôle habituellement utilisés dans l'aéronautique. En particulier le choix du plan de vol et le passage en mode ‹‹révision›› sont regroupés sur un seul bouton. Les objets disponibles dans la version actuelle du langage choisi, JAVA™, ne permettant pas une réalisation aisée de ces boutons rotatifs, ils ont donc été simulés par des listes de boutons.

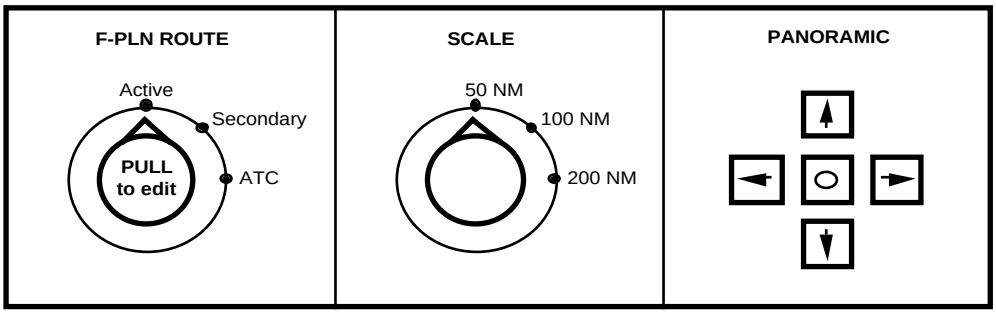

Figure A-4 : Proposition de tableau de contrôle.

3. Réalisation logicielle

Le passage de la spécification à l'implémentation s'est effectué "à la main" sans l'aide d'outils automatiques, ni même de générateurs d'interface. En effet, les outils disponibles au moment de la réalisation ont été jugés trop limités.

Cette section passe en revue certains des aspects techniques de la réalisation. Elle s'intéresse également à l'implémentation des singularités. Des extraits de la documentation du projet, ainsi que des copies d'écran de l'interface produite sont ensuite fournis.

3.1. Aspects techniques

L'implémentation de la maquette CoTASS se base sur le langage orienté-objet JAVA™ dans sa version 1.0. Le développement s'est effectué sur un serveur SUN™ sous Solaris®, mais la portabilité du langage permet de supposer, même si cela n'a pas été vérifié expérimentalement, que cette maquette peut fonctionner sur toute machine virtuelle JAVA™.

Nous avons dû utiliser un processus léger ("thread") pour la partie simulateur afin de garantir une mise à jour en temps réel de la position de l'aéronef. De même, l'affichage a imposé l'utilisation de "double-buffering" afin d'éviter un scintillement peu agréable. L'effort d'implémentation peut être estimé entre un et deux homme-mois, pour environ deux mille lignes de code informatique commenté.

Les bases de données utilisées sont des bases de données réelles codées en latitude, longitude, et altitude. La position de l'avion simulé, mise à jour toutes les secondes, est exprimée dans ce repère selon le système géographique WGS 84. En outre, des points de report quelconques peuvent être ajoutés à la base de données en les déchargeant par exemple à partir d'un GPS.

3.2. Implémentation des singularités

Les singularités, pour l'essentiel la fonction "UNDO" et la fonction "STOP", ont été réalisés à l'aide de méthodes de l'objet principal de l'application comme l'indique l'extrait de la documentation figure A-6. Les principaux effets de bords de ces méthodes sont la mise à jour des attributs, puis l'appel à

la méthode de rafraîchissement de l'écran "update". La méthode "STOP" dispose en outre d'un système d'exclusion mutuelle car elle peut être appelée par plusieurs processus légers.

Notons que l'appel de cette méthode "STOP" s'effectue aussi bien en provenance du noyau fonctionnel que de l'interface. Cette implémentation est intéressante, car elle indique que deux singularités spécifiées comme étant distinctes lors de l'analyse de tâche, car étant issus de deux agents distincts, sont réalisées par une même méthode. L'interface, donc l'utilisateur, est ici situé au même niveau que l'abstraction, le simulateur de vol, pour le contrôle de l'édition de trajectoire.

3.3. Documentation

La documentation des classes utilisées par CoTASS est fournie sous la forme d'une série de pages d'hypertexte au format HTML. Cette documentation a été générée automatiquement par l'utilitaire "javadoc" à partir des fichiers sources, commentés en utilisant le format adéquat [Flanagan 1996]. La langue anglaise a été préférée au français dans le but de permettre une diffusion plus importante des sources du programme. Les figures A-5 et A-6 montrent des extraits de cette documentation.

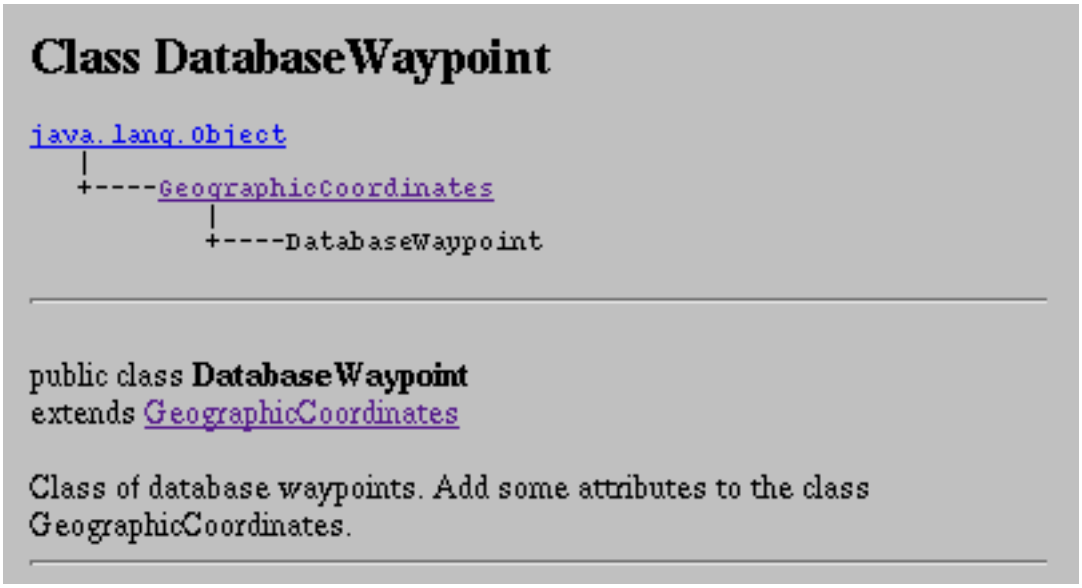

Figure A-5 : Exemple de documentation.

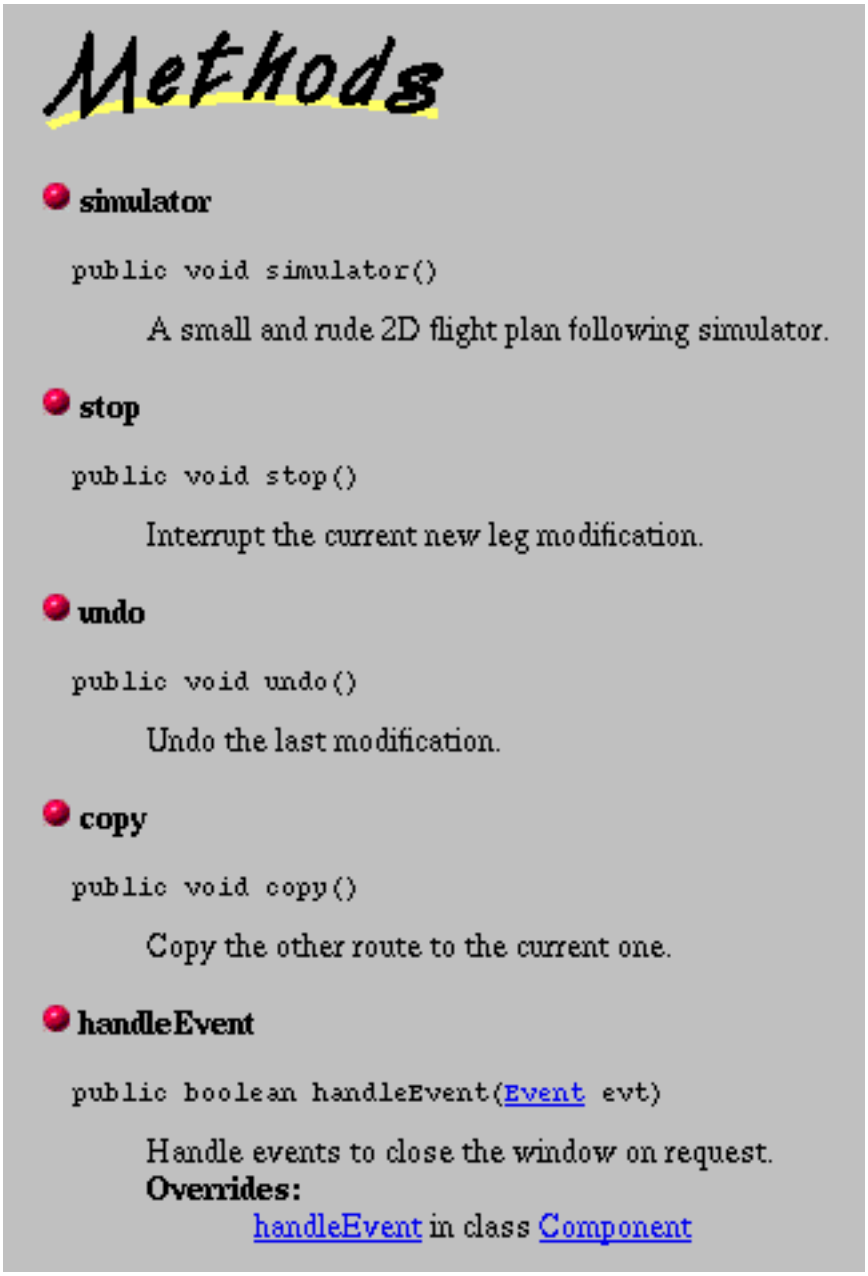

Figure A-6 : Exemple de documentation.

3.4. Copies d'écran

Les copies d'écran de la maquette cotass figures A-7, A-8, et A-9 illustrent les principales tâches et modes d'interaction utilisés.

Figure A-7 : Copie d'écran de CoTASS lors d'une édition de trajectoire en cours.

Figure A-8 : Copie d'écran de CoTASS lors d'une alarme proximité du sol.

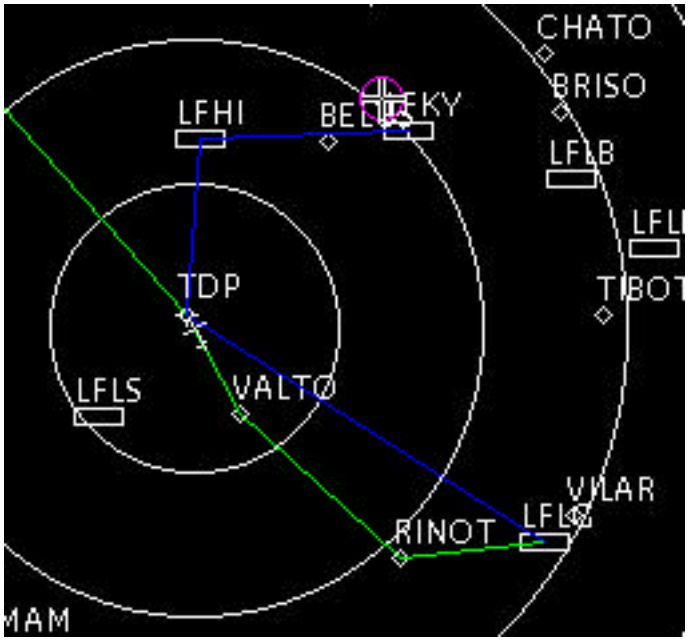

Figure A-9 : Copie d'écran de COTASS lors de l'édition de la trajectoire secondaire.

4. Lexique

Automanette : Système de gestion automatique des moteurs d'un aéronef.

FMGS (Flight Management and Guidance System) : Ordinateur embarqué permettant la gestion des paramètres de navigation et l'optimisation en temps et consommation de la route suivie par l'aéronef.

GPS (Global Positioning System) : Système de positionnement à quatre dimensions (Longitude, Latitude, Altitude, Temps Universel) par satellite de très grande précision (de l'ordre du mètre et de la milliseconde) mis en place par le Département de la Défense américain. La précision du système est actuellement volontairement dégradée pour des raisons stratégiques à une centaine de mètres.

GPWS (Ground Proximity Warning System) : Système embarqué d'alerte de proximité du sol. Ce système détecte les risques de collision d'un aéronef avec le sol, et en avertit l'équipage. De nombreuses interfaces, lumineuses, sonores, vocales, ou/et graphiques, sont utilisées.

MCDU (Multiple Control Data Unit) : Terminal embarqué doté d'un écran et d'un clavier alphanumériques, de touches de fonction, et de touches multiplexées. Il est destiné à l'entrée des paramètres de vol de l'aéronef.

ND (Navigation Display) : Écran regroupant les informations utiles à la navigation comme la route suivie, la position des aéroports et des balises de radionavigation, et éventuellement en surimpression les spots du radar météo, la position d'autres aéronefs, ou encore la proximité du sol.

5. Bibliographie

[Airbus Industrie 1992] Airbus Industrie. *Flight Crew Operating Manual A320*. 1992.

[Boeing 1995] Boeing. *FAA certifies Boeing FANS-1 navigation system*. The Boeing Company, June, 20 1995. World Wide Web Document "http://www.boeing.com/news.release.950620.html".

[Casaux 1995] Casaux F. *Propositions relatives à l'utilisation du TCAS2 en France*. Centre d'Étude de la Navigation Aérienne, 7 Février 1995. Rapport N95506.

[Chatty et al. 1994] Chatty S. & Colin F.-R. *Coordination téléphonique intégrée à l'image radar*. Centre d'Étude de la Navigation Aérienne, Mai 1994. Rapport CENA/PII/94.644.

[Chatty et al. 1996] Chatty S. & Lecoanet P. Pen computing for air traffic control. *Human factors in computing systems (CHI'96), Vancouver, Canada*, April 13-18 1996. vol. 1(2), p. 87-94.

[Combes 1993] Combes M. *Avionique de la navigation aérienne*. Toulouse : Cépaduès Éditions, 1993.

[Cottenceau 1993] Cottenceau J.-P. *Interfaces pilotes pour les échanges ATC par Data Link*. Mémoire de fin d'étude : École Nationale de l'Aviation Civile, Juin 1993.

[Dupont 1994] Dupont J. Les systèmes anticollision : Utiles malgré tous leurs défauts. *Air & Cosmos Aviation International*, 1994. n° 1454, p. 30-32.

[Dupont 1995] Dupont J. CELCIUSTECH introduit l'anticollision par GPS. *Air & Cosmos Aviation International*, 1995. n° 1505, p. 32.

[Flanagan 1996] Flanagan D. *Java in a nutshell / A desktop quick reference for java programmers*. Sebastopol (CA), USA : O'Reilly & Associates, 1996.

[Garmin 1995] Garmin. *GPS90 Navigateur™ Personnel*. Garmin® International, Mars 1995. Manuel d'utilisation Part#190-00084-00 Révision A.

[Haertl 1995] Haertl D. *Operational aspects of a distributed simulation*. DLR Human Engineering and Simulation Branch, October, 13 1995. World Wide Web Document "http://dv.bs.dlr.de/ff/fl/24/annette/wp2_doc.html".

[Jambon et al. 1993] Jambon F. & Coutaz J. Contrôle aérien et liaisons de données : Vers un plus grand partage de l'information. *Cinquièmes journées sur l'ingénierie des Interfaces Homme-Machine (IHM'93), Lyon, France*, Octobre 1993. p. 109-114.

[Knox et al. 1991] Knox C.E. & Scalon C.H. *Flight test with a data link used for air traffic control information exchange*. NASA Langley research center, September 1991. Technical paper NASA TP-3135.

[L'Ebraly 1994] L'Ebraly H. *Étude de l'interface homme-machine sur les systèmes d'assistance à l'opérateur / Application aux cockpits d'avion*. École Nationale Supérieure de l'Aéronautique et de l'Espace, Juin 1994. Rapport de fin d'étude.

[Monzel et al. 1993] Monzel F.-G. & Bories A. *Système de guidage et de contrôle de la circulation de surface*. Revue des télécommunications, Alcated Alstom Publications, 1993. p. 51-59.

[Perry 1991] Perry T.A.(. Improving the world's largets, most advanced system. *IEEE Spectrum*, February 1991. p. 22-36.

[Pfuhl et al. 1993] Pfuhl C., Greving G., & Mandelka G. *Quarante ans d'utilisation des systèmes d'atterrissage aux instruments*. Revue des télécommunications, Alcated Alstom Publications, 1993. p. 41-50.

[Renou 1993] Renou L. *Symbologies de négociation de trajectoire-4D*. Mémoire de fin d'étude : École Nationale de l'Aviation Civile, Juin 1993.

[Sebillotte 1994] Sebillotte S. *Méthodologie pratique d'analyse de la tâche en vue de l'extraction de caractéristiques pertinentes pour la conception d'interfaces*. INRIA Rocquencourt, Mai 1994. Rapport Technique n° 163.

[SIA 1991] SIA. *Procédures de radiotéléphonie à l'usage de la circulation aérienne générale : Phraséologie*. Service de l'Information Aéronautique (SIA), Juin 1991.

[Sylla 1993] Sylla B. *TNP operational scenarios*. Centre d'Étude de la Navigation Aérienne / Sextant Avionique, December 1993. Report CNS-R93068.

[Waller 1992] Waller M.C. *Flight benefit of integrated data link communication*. NASA Langley research center, April 1992. Technical paper NASA TP-3219.

BIBLIOGRAPHIE

[Abowd et al. 1989] Abowd G., Bowen J., Dix A., Harrison M., & Took R. *User interface languages: A survey of existing methods*. Programming Research Group at Oxford University Computing Laboratory, October 1989. Report PRG-TR-5-89.

[Abowd et al. 1992] Abowd G.D., Coutaz J., & Nigay L. *Structuring the space of interacting system properties*. IFIP transactions / Engineering for human-computer interaction, North-Holland : Elsevier Science Publishers, 1992. p. 113-129.

[Accot et al. 1996] Accot J., Chatty S., & Palanque P. A formal description of low level interaction and its application to multimodal interactive systems. *Third International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'96), Namur, Belgium*, 5-7 June 1996.

[Airbus Industrie 1992] Airbus Industrie. *Flight Crew Operating Manual A320*. 1992.

[Amalberti 1992] Amalberti R. Sécurité des vols et automatisation des cockpits. *Le transpondeur: bulletin de laison*, Avril 1992. n° 7, p. 7-16.

[Avions Pierre Robin 1986] Avions Pierre Robin. *Manuel de vol DR400*. Avions Pierre Robin, 1986.

[Balbo 1994] Balbo S. *Un pas vers l'évaluation automatique des interfaces homme-machine*. Thèse en Informatique : Université Joseph Fourier (Grenoble I), 5 Septembre 1994.

[Barnard 1985] Barnard J.P. *Cognitive resources and the learning of computer dialogs*. Interfacing throught, Cognitive aspect of Human-Computer Interaction, MIT Press, 1985. p. 112-158.

[Barthet 1988] Barthet M.-F. *Logiciels interactifs et ergonomie : Modèles et méthodes de conception*. Paris, France : Dunod, 1988.

[Bastien et al. 1993] Bastien C.J. & Scapin D.L. *Ergonomic criteria for the evaluation of human-computer interfaces*. INRIA Rocquencourt, Juin 1993. Rapport technique N°156.

[Bertaud du Chazaud 1994] Bertaud du Chazaud H. *Le dictionnaire des synonymes et contraires*. Paris, France : Le Robert, 1994.

[Billings 1991] Billings C.E. *Human-centered aircraft automation: A concept and guidlines*. NASA Ames research center, August 1991. Technical memorandum NASA-TM-103885.

[Billings 1996] Billings C.E. *Human-centered aviation automation: Principles and guidlines*. NASA Ames research center, February 1996. NASA Technical Memorandum NASA-TM-110381.

[Bodart et al. 1994] Bodart F. & Vanderdonckt J. *Guide ergonomique de la présentation des applications hautement interactives*. Namur, Belgique : Presses Universitaires de Namur, 1994.

[Boehm 1988] Boehm B.W. A spiral model of software development and enhancement. *IEEE Computer*, May 1988.

[Boeing 1995] Boeing. *FAA certifies Boeing FANS-1 navigation system*. The Boeing Company, June, 20 1995. World Wide Web Document "http://www.boeing.com/news.release.950620.html".

[Bordon 1994] Bordon S. *Formalisme pour la représentation des scénarios dans les didacticiels*. DEA Système d'information : Université Joseph Fourier (Grenoble I), Juin 1994.

[Bowen et al. 1995] Bowen J.P. & Hinchey M.G. Seven more myths on formal methods. *IEEE Software*, 1995.

[Boy 1989] Boy G. The block representation in knowledge acquisition for computer integrated documentation. *4th AIAA-sponsored knowledge acquisition integrated documentation for knowledge-based systems workshop, Banff, Canada*, October 1989.

[Boy 1993] Boy G. Knowledge acquisition in dynamic systems: how can logicism and situatedness go together? *EKAW'93*, 1993.

[Boy et al. 1993] Boy G. & Hollnagel E. *Cockpit evolution and human-centered automation*. EURISCO, October 1993. Technical Report T-93-002-GB-VI.

[Brun et al. 1995] Brun P. & Beaudoin-Lafon M. A taxonomy and evaluation of formalisms for the specification of interactive systems. *10th annual conference of the British Human-Computer Interaction Group (HCI'95), University of Huddersfeild (UK)*, August-September 1995.

[Card et al. 1983] Card S.K., Moran T.P., & Newell A. *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, 1983.

[Casaux 1995] Casaux F. *Propositions relatives à l'utilisation du TCAS2 en France*. Centre d'Étude de la Navigation Aérienne, 7 Février 1995. Rapport N95506.

[Chatty et al. 1994] Chatty S. & Colin F.-R. *Coordination téléphonique intégrée à l'image radar*. Centre d'Étude de la Navigation Aérienne, Mai 1994. Rapport CENA/PII/94.644.

[Chatty et al. 1996] Chatty S. & Lecoanet P. Pen computing for air traffic control. *Human factors in computing systems (CHI'96), Vancouver, Canada*, April 13-18 1996. vol. 1(2), p. 87-94.

[Chevalier 1992] Chevalier G. Les retombées d'un nuage d'innovations. *Les Cahiers de Science & Vie : Concorde / La puissance du rêve, le rêve de la puissance*, Juin 1992. n° 9, p. 72-76.

[Combes 1993] Combes M. *Avionique de la navigation aérienne*. Toulouse : Cépaduès Éditions, 1993.

[Cottenceau 1993] Cottenceau J.-P. *Interfaces pilotes pour les échanges ATC par Data Link*. Mémoire de fin d'étude : École Nationale de l'Aviation Civile, Juin 1993.

[Coutaz 1990] Coutaz J. *Interfaces homme-ordinateur : Conception et réalisation*. Paris : Dunod informatique, 1990.

[Coutaz 1994] Coutaz J. *NEIMO (NouvellEs Interfaces et MOdélisation)*. CLIPS-IMAG, Avril 1994. Proposition de projet IMAG.

[Coutaz et al. 1993a] Coutaz J., Faconti G., Paterno F., Nigay L., & Salber D. *MATIS: A UAN description and lesson learned*. ESPRIT BRA 7040 Amodeus-2, June 1993a. Working paper SM/WP14.

[Coutaz et al. 1993b] Coutaz J., Paterno F., Giorgio F., & Nigay L. A comparison of approaches for specifying multimodal interactive systems. *European Research Consortium for Informatics and Mathematics (ERCIM'93), Nancy, France*, 2-4 November 1993b. p. 165-174.

[Dix et al. 1993] Dix A., Finlay J., Abowd G., & Beale R. *Human-computer interaction*. Cambridge, U.K. : Prentice Hall, 1993.

[Dorwell et al. 1989] Dorwell J. & Long J. Towards a conception for an engineering discipline of human factors. *Ergonomics*, November 1989. vol. 32, n° 11, p. 1513-1535.

[Dupont 1994a] Dupont J. Le dossier des commandes de vol électriques. *Air & Cosmos Aviation International*, 1994a. n° 1452/53, p. 61-73.

[Dupont 1994b] Dupont J. Les systèmes anticollision : Utiles malgré tous leurs défauts. *Air & Cosmos Aviation International*, 1994b. n° 1454, p. 30-32.

[Dupont 1995] Dupont J. CELCIUSTECH introduit l'anticollision par GPS. *Air & Cosmos Aviation International*, 1995. n° 1505, p. 32.

[Flanagan 1996] Flanagan D. *Java in a nutshell / A desktop quick reference for java programmers*. Sebastopol (CA), USA : O'Reilly & Associates, 1996.

[Frank et al. 1993] Frank M. & Foley J. Model-based User Interface Design by Example and by Interview. *ACM symprosium on User Interface Software and Technology (UIST'93), Atlanta (Georgia), USA*, November 3-5 1993.

[Garavel 1989] Garavel H. *Compilation et vérification de programmes LOTOS*. Thèse en informatique : Université Joseph Fourier (Grenoble I), Novembre 1989.

[Garmin 1995] Garmin. *GPS90 Navigateur™ Personnel*. Garmin® International, Mars 1995. Manuel d'utilisation Part#190-00084-00 Révision A.

[Grau et al. 1995] Grau J.-Y. & Doireau P. *Erreur humaine et automatisation des cockpits*. Centre d'Études et de Recherche de Médecine Aérospatiale, Octobre 1995. Rapport de recherche 95-33.

[Gray et al. 1994] Gray P., England D., & McGowan S. *XUAN: Enhancing the UAN to capture temporal relation among actions*. Department of Computing Science, University of Glasgow, February 1994. Department research report IS-94-02.

[Gray et al. 1996] Gray P.D. & Johnson C.W. Supporting error-driven design. *Third International Eurographics Workshop on Design, Specification, and Verification of Interactive systems (DSV-IS'96), Namur, Belgium*, 5-7 June 1996.

[Grudin 1988] Grudin J. Why CSCW applications fail: problems in the design and evaluation of organisation interface. *Proceedings of the ACM conference on Computer-Supported Cooperative Work, Portland (Oregon), USA*, September 1988. p. 85-93.

[Haertl 1995] Haertl D. *Operational aspects of a distributed simulation*. DLR Human Engineering and Simulation Branch, October, 13 1995. World Wide Web Document "http://dv.bs.dlr.de/ff/fl/24/annette/wp2_doc.html".

[Hall 1990] Hall A. Seven myths on formal methods. *IEEE Software*, September 1990. p. 11-19.

[Hamouche 1995] Hamouche H. *De la modélisation des tâches utilisateurs à la spécification conceptuelle et sémantique d'interfaces homme-machine.* Thèse en informatique : Université Paris VI, Décembre 1995.

[Harrison et al. 1994] Harrison M.D. & Duke D.J. *A review of the formalisms for describing interactive behaviour*. ESPRIT BRA 7040 Amodeus-2, January 1994. Working paper SM/WP28.

[Hartson et al. 1990] 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, p. 181-203.

[Hartson et al. 1992] Hartson R.H. & Gray P.D. *Temporal aspects of tasks in the User Action Notation*. Human Computer Interaction, Lawrence Erlbaum Associates, 1992. p. 1-45.

[Hix et al. 1993] Hix D. & Hartson H.R. *Developping user interfaces: Ensuring usability through product & process*. Newyork, USA : John Wiley & Sons, inc., 1993.

[Hollnagel et al. 1993] Hollnagel E., Warren C., & Cacciabue P.C. Automation in aerospace and aviation applications: Where are the limits. *Human-Machine*

Interaction and Artificial Intelligence in Aerospace (HMI-AI-AS'93) Workshop, Toulouse, France, September 28-30 1993.

[ISO-8807:1989 1995] ISO-8807:1989. *Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour*. Internation Standard Organisation (ISO), Last update 1995. Standard ISO 8807:1989.

[Jackson 1983] Jackson M.A. *System developement*. Englewood Cliffs (NJ), USA : Prentice Hall, 1983.

[Jacquet et al. 1996] Jacquet P., Ledru Y., Nicollin X., & Potet M.-L. *Exposition "logiciels critiques"*. Médiathèque IMAG, 15 janvier - 15 avril 1996. Documentation.

[Jambon 1995] Jambon F. Interruptions et formalismes de scripts de tâches. *Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95), Toulouse, France*, 11-13 Octobre 1995. p. 161-168.

[Jambon et al. 1993] Jambon F. & Coutaz J. Contrôle aérien et liaisons de données : Vers un plus grand partage de l'information. *Cinquièmes journées sur l'ingénierie des Interfaces Homme-Machine (IHM'93), Lyon, France*, Octobre 1993. p. 109-114.

[Jambon et al. 1994a] Jambon F. & Coutaz J. Retours d'information sensori-moteurs des dispositifs physiques : Tâche de sélection avec une souris sans "clic". *12th Triennal Congress of the International Ergonomics Association (IEA'94), Toronto, Canada*, Août 1994a. vol. 4(6), p. 464.

[Jambon et al. 1994b] Jambon F. & Karsenty L. Formalisation des interfaces et travail coopératif : Quelles conséquences ? *Sixièmes journées sur l'ingénierie des Interfaces Homme-Machine (IHM'94), Lille, France*, Décembre 1994b. p. 163-168.

[Johnson 1996a] Johnson C. The evaluation of user interface design notations. *Proceedings of the Design, Specification and Verification of Interactive Systems (DSV-IS'96), Namur, Belgique*, 1996a.

[Johnson 1996b] Johnson C. The Namur principles: criteria for the evaluation of user interface notations. *Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'96), Namur, Belgique*, 1996b.

[Johnson et al. 1996] Johnson C.W. & Telford A.J. Using formal methods to analyse human error and system failure during accident investigations. *Human-Computer Interaction Journal*, November 1996.

[Johnson et al. 1993] Johnson H. & Johnson P. ADEPT-advanced design environment for prototyping with task models. *Human Factors in Computing Systems (InterCHI'93), Amsterdam, The Netherlands*, 24-29 April 1993. vol. adjunct Proceedings(2), p. 56.

[Jones 1979] Jones C.P. *Automatic fault protection in the Voyager spacecraft*. American Institute of Aeronautics and Astronautics, 1979. Paper n°79-1919.

[Knox et al. 1991] Knox C.E. & Scalon C.H. *Flight test with a data link used for air traffic control information exchange*. NASA Langley research center, September 1991. Technical paper NASA TP-3135.

[Krakowiak 1985] Krakowiak S. *Principes des systèmes d'exploitation des ordinateurs*. Dunod informatique, 1985.

[L'Ebraly 1994] L'Ebraly H. *Étude de l'interface homme-machine sur les systèmes d'assistance à l'opérateur / Application aux cockpits d'avion*. École Nationale Supérieure de l'Aéronautique et de l'Espace, Juin 1994. Rapport de fin d'étude.

[Laronche 1993] Laronche M. *Les suites de la catastrophe du mont Sainte-Odile*. Le Monde. Paris, France : Samedi 18 décembre 1993. p. 24.

[Lenman et al. 1994a] Lenman S. & Robert J.-M. A framework for error recovery. *International Ergonomics Association (IEA'94), Toronto, Canada*, August 15-19 1994a. vol. 6(6), p. 374-376.

[Lenman et al. 1994b] Lenman S. & Robert J.-M. Investigating the granularity of the Undo function in human-computer interfaces. *Applied Psychology: An international review*, 1994b. vol. 43, n° 4 (special issue on Human Errors), p. 543-564.

[Lim et al. 1994] Lim K.Y. & Long J.B. *The MUSE method for usability engineering*. United Kingdom : Cambridge University Press, 1994.

[Long 1995] Long J. Integrating human factors with software engineering for human-computer interaction. *Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95), Toulouse, France*, 11-13 Octobre 1995. p. 1-19.

[Long et al. 1989] Long J. & Dowell J. Conceptions of the discipline of HCI: Craft, Applied Science, and Engineering. *Fifth conference of the BCS HCI SIG*, 1989.

[MacLean et al. 1990] MacLean P.D. & Guyot R. *Les trois cerveaux de l'homme*. Paris, France : Robert Laffont, 1990.

[Mathé 1990] Mathé N. *Assistance intelligente au contrôle de processus : Application à la télémanipulation Spatiale*. Thèse en Intelligence Artificielle : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1990.

[McDermind et al. 1984] McDermind J. & Ripkin K. *Life cycle support in the ADA environment*. Cambridge University Press, 1984.

[McGowan 1995] McGowan S. *From UAN to eXUAN: Specifying simulations of the temporal properties of interaction*. University of Glasgow, Department of Computing Science, March 1995. Technical report TR-1995-5.

[Mellor 1993] Mellor P. *Review of AMJ 25.1303 and related documents*. Center for Software Reliability, City University, 1993. News posted in "sci.aeronautics.airliners" newsgroup.

[Monzel et al. 1993] Monzel F.-G. & Bories A. *Système de guidage et de contrôle de la circulation de surface*. Revue des télécommunications, Alcated Alstom Publications, 1993. p. 51-59.

[Neumann 1996] Neumann P.G. Forum on risks to the public in computers and related systems ("comp.risks"). *Moderated Newsgroup*, 1996.

[Nicolet et al. 1990] Nicolet J.-L., Carnino A., & Wanner J.-C. *Catastrophes ? Non merci ! La prévention des risques technologiques et humains*. Éditions Masson, 1990.

[Nielsen 1994] Nielsen J. Enhancing the explonatory power of usability heuristics. *Human Factors in Computing Systems (CHI'94), Boston (MA), USA*, 24-29 April 1994. vol. conference proceedings(2), p. 152-153.

[Nigay 1994] Nigay L. *Conception et modélisation logicielles des systèmes interactifs : Application aux interfaces multimodales*. Thèse en informatique : Université Joseph Fourier (Grenoble I), 1994.

[Norman 1990] Norman D.A. *The design of every day things*. New York, USA : Doubleday Currency, 1990.

[Norman 1991] Norman D.A. Cognitive science in the cockpit. *Gateway (Crew System Information Analysis Center - University of Dayton Research Institute)*, Spring 1991. vol. II, n° 2, p. 1-6.

[O'Conaill et al. 1995] O'Conaill B. & Frohlich D. Timespace in the workplace: Dealing with interruptions. *Human factors in computing systems (CHI'95), Denver (Colorado), USA*, May 7-11 1995. vol. Conference companion(2), p. 262-263.

[Palanque 1992] Palanque P. *Modélisation par objets coopératifs d'interfaces homme-machine dirigées par l'utilisateur*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1992.

[Palanque et al. 1995a] Palanque P. & Bastide R. Spécifications formelles pour l'ingénierie des interfaces homme machine. *Techniques et Science Informatiques*, 1995a. vol. 14, n° 4, p. 473-500.

[Palanque et al. 1995b] Palanque P., Bastide R., & Senges V. Task model - system model: towards an unifying formalism. *HCI International conference, Yokoohama, Japan*, 9-14 July 1995b. p. 489-494.

[Palanque et al. 1994] Palanque P., Long J.B., Tarby J.-C., Barthet M.-F., & Lim K.Y. Conception d'applications ergonomiques : une méthode pour informaticiens et une méthode pour ergonomes. *Ergonomie et Informatique Avancée (ERGO-IA'94), Biarritz, France*, 28-10 octobre 1994.

[Pendibidou 1982] Pendibidou J.-M. Présentation du projet THÉRÈSE. *Technique et Science Informatiques*, 1982. vol. 1, n° 3, p. 253-257.

[Pérez-Quiñones et al. 1996] Pérez-Quiñones M.A. & Sibert J.L. Negaciating user-initiated cancellation and interruption requests. *Conference on Human Factors in Computing Systems (CHI'96), Vancouver (British Columbia), Canada*, April 14-18 1996. vol. Conference companion(2), p. 267-268.

[Perry 1991] Perry T.A.(. Improving the world's largets, most advanced system. *IEEE Spectrum*, February 1991. p. 22-36.

[Peterson 1981] Peterson J.L. *Petri net theory and modelling of systems*. Prentice Hall, 1981.

[Pfuhl et al. 1993] Pfuhl C., Greving G., & Mandelka G. *Quarante ans d'utilisation des systèmes d'atterrissage aux instruments*. Revue des télécommunications, Alcated Alstom Publications, 1993. p. 41-50.

[Pleczon 1992] Pleczon P. *Éléments de méthodologie et outils pour l'assistance à l'opérateur : Application à la conduite automobile*. Thèse en Informatique : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1992.

[Racca 1992] Racca E. Le concept de CRM / Crew Ressource Management. *Le transpondeur / bulletin de laison*, Avril 1992. n° 7, p. 55-59.

[Rasmussen 1986] Rasmussen J. *Information processing and Human-Machine Interaction : An approach to cognitive engineering*. North-Holland, 1986.

[Reason 1990] Reason J. *Human error*. Cambridge University Press, 1990.

[Reason 1993] Reason J. *L'erreur humaine*. Paris, France : Presses Universitaires de France, 1993.

[Renou 1993] Renou L. *Symbologies de négociation de trajectoire-4D*. Mémoire de fin d'étude : École Nationale de l'Aviation Civile, Juin 1993.

[Robert 1992] Robert P. *Le petit Robert : Dictionnaire de la langue française*. Paris, France : Le Robert, 1992.

[Rouff 1996] Rouff C. Formal specification of user interface. *ACM SIGCHI bulletin*, July 1996. vol. 28, n° 3, p. 27-33.

[Rouncefield et al. 1994] Rouncefield M., Hughes J.A., Rodden T., & Viller S. Working with "constant interruption": CSCW and the small office. *Conference on Computer Supported Cooperative Work (CSCW'94), Chapel Hill (North California), USA*, October 22-26 1994. p. 275-286.

[Royce 1970] Royce W.W. Managing the development of large software systems. *WESTCON, California, USA*, 1970.

[Salber 1995] Salber D. *De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs : L'exemple de la communication homme-homme médiatisée*. Thèse en informatique : Université Joseph Fourier (Grenoble I), 1995.

[Scapin et al. 1989] Scapin D.L. & Pierret-Golbreich C. MAD : Une méthode analytique de description des tâches. *Colloque sur l'ingénierie des Interfaces Homme-Machine (IHM'89), Sophia-Antipolis, France*, Mai 1989. p. 131-148.

[Sebillotte 1994] Sebillotte S. *Méthodologie pratique d'analyse de la tâche en vue de l'extraction de caractéristiques pertinentes pour la conception d'interfaces*. INRIA Rocquencourt, Mai 1994. Rapport Technique n° 163.

[Sebillotte et al. 1994] Sebillotte S., Alonso B., Fallah D., Hamouche H., & Scapin D.L. *Note de recherche concernant le formalisme MAD*. INRIA Rocquencourt, Novembre 1994. Document interne.

[Sebilotte 1991] Sebilotte S. *Décrire des tâches selon les objectifs des opérateurs : De l'interview à la formalisation*. Le Travail humain, Paris, France : Presses Universitaires de France, 1991. p. 193-223.

[SIA 1991] SIA. *Procédures de radiotéléphonie à l'usage de la circulation aérienne générale : Phraséologie*. Service de l'Information Aéronautique (SIA), Juin 1991.

[Sifakis 1977] Sifakis J. *Use of Petri nets for performance evaluation*. Measuring, Modelling, and Evaluating Computer Systems, North Holland, 1977.

[Swain et al. 1983] Swain A.D. & Guttmann H.E. *Handbook of human reliability analysis with emphasis on nuclear power plant applications*. Nuclar Regulatory Commission (NUREG), August 1983. Final report NUREG/CR-1278F.

[Sweet 1995] Sweet W. The glass cockpit. *IEEE Spectrum*, September 1995. p. 30-38.

[Sylla 1993] Sylla B. *TNP operational scenarios*. Centre d'Étude de la Navigation Aérienne / Sextant Avionique, December 1993. Report CNS-R93068.

[Szekely et al. 1993] Szekely P., Luo P., & Neches R. Beyong interface builders: model-based interface tools. *Human Factors in Computing Systems (InterCHI'93), Amsterdam, The Netherlands*, 24-29 April 1993. vol. conference proceedings(2), p. 383-390.

[Szekely et al. 1996] 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*, 1996.

[Tarby 1993] Tarby J.-C. *Gestion automatique du dialogue homme-machine à partir de spécifiactions conceptuelles*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1993.

[Télérama 1993] Télérama. *ShowView : Dès la semaine prochaine, dans nos grilles télé, un code pour programmer votre magnétoscope sans être polytechnicien*. Télérama. 1993. vol. 20 Octobre 1993,n° 2284, p. 105.

[Walker et al. 1995] Walker W. & Cragon H.G. Interrupt processing in concurent processors. *Computer*, June 1995. vol. 28, n° 6, p. 36-45.

[Waller 1992] Waller M.C. *Flight benefit of integrated data link communication*. NASA Langley research center, April 1992. Technical paper NASA TP-3219.

[Wanner 1992] Wanner J.-C. Y a-t-il encore un pilote dans l'avion ? *Le transpondeur: bulletin de laison*, Avril 1992. n° 7, p. 51-54.

[Wiener 1989] Wiener E.L. *Human factors of advanced technology (glass cockpits) transport aircraft*. National Aeronautics and Space Administration, June 1989. Report NASA-CR-17752.

[Wiener et al. 1980] Wiener E.L. & Curry R.E. *Flight-deck automation: Promises and problems*. National Aeronautics and Space Administration, June 1980. Report NASA-TM-81206.

[Woods 1990] Woods D.D. *Modeling and predicting human error*. Human performance models for computer-aided engineering, Washington : Academic Press, 1990. p. 248-274.

[Yang 1992] Yang Y. Anatomy of the design of an undo support facility. *International Journal of Man-Machine Studies*, 1992. vol. 36, n° 1, p. 81-95.

TITRE : Erreurs et interruptions du point de vue de l'ingénierie de l'interaction homme-machine.

MOTS-CLEFS : Interaction Homme-Machine, Interruption, Erreur, Analyse de tâche, Formalisme, MAD, UAN, Réseaux de Petri.

RÉSUMÉ : Les erreurs humaines et les interruptions sont des phénomènes courants mais négligés, voire ignorés, dans le processus de développement des systèmes interactifs. En outre, leur automatisation massive n'a pas éliminé erreurs et interruptions, mais en a accentué le caractère critique. Ce mémoire a pour objet l'amélioration de la fiabilité des systèmes homme-machine par la prise en compte explicite des erreurs et des interruptions dans la pratique de l'Ingénierie de l'Interaction Homme-Machine. Nous appliquons notre étude au cas exigeant des systèmes critiques automatisés et notamment aux systèmes aéronautiques, pour lesquels les erreurs humaines et les interruptions ont une importance décisive.

Dans la première partie du mémoire, dédiée aux concepts, nous présentons une revue des recherches sur l'automatisation et sur l'analyse des erreurs humaines telle que l'envisage la psychologie cognitive. A notre tour, nous proposons le concept de *singularité* comme notion pivot aux phénomènes d'erreur et d'interruption. Nous en fournissons un modèle qui explicite les activités mentales en relation avec la détection et la correction de singularité.

La seconde partie du mémoire a trait à l'expression formelle des singularités dans le processus de développement d'un système interactif. Après une revue des formalismes et des notations utilisés en Interaction Homme-Machine, nous retenons MAD, UAN et les réseaux de Petri pour leurs qualités et leur usage complémentaires. Pour chacun, nous proposons les extensions nécessaires à l'expression des singularités. Nous fournissons également les règles de traduction de ces extensions entre les trois formalismes retenus, évitant ainsi la perte de conformité au cours du processus de développement d'un système.

TITLE : Errors and Interruptions in Computer Human Interaction from the Software Engineering Perspective

KEYWORDS : Computer Human Interaction, Error, Interruption, Task analysis, Formalism, MAD, UAN, Petri nets.

ABSTRACT : Although human errors and interruptions are frequently observed in computer-human interaction, they have been overlooked or ignored by software designers and developers. This doctoral study addresses the problem of making explicit the notions of error and interruption within the software development process of interactive systems. The results are applied to highly automated reactive systems, such as avionic systems, for which errors and interruptions may have dramatical consequences on human security.

The first part of the dissertation addresses the conceptual aspects of the problem. It discusses the benefits and pitfalls of automation as well as the notion of error as developed by current research in cognitvive psychology. From this analysis, we propose the concept of *singularity* as a common ground for errors and interruptions. By making explicit the mental activities involved in error detection and correction, our model of singularity forces software designers and implementers to consider the right questions.

The second part of the dissertation is concerned with the formal expression of singularities. We have selected three formalisms, MAD, UAN and Petri nets, for their complementary properties and purpose within the software development process of interactive systems. For each of them, we have developed the extensions appropriate for the expression of singularities as well as a set of rules for translating these extensions across the three formalisms. By so doing, we maintain the conformity of singularity expressions along the software development process.