



**Projet VERBATIM**

# Sous-Projet 6

# Lot 1 Intégration

## *Ergonomie et Approches fondées sur des techniques formelles*
## *Positionnement dans un cycle de vie*

**Version : 1.00**

**Date : 5/01/07**

Auteurs :
Laurence Nigay, IIHM LIG, (ex- CLIPS-IMAG), Université de Grenoble 1
Bruno d'Ausbourg, ONERA/DTIM

---





**LSR**


# Table des matières

# 1. Introduction

## 1.1 Rappel de l'objectif

Le présent document est rédigé dans le cadre du projet RNRT VERBATIM, dont l'objet est la VERification Biformelle et Automatisation du Test des Interfaces Multimodales.

Malgré la maturité acquise aujourd'hui en Interaction Homme-Machine, des difficultés persistent. Parmi elles, la discontinuité du processus de développement en partie due à la pluridisciplinarité de l'ouvrage : les ergonomes analysent l'activité, établissent les modèles de tâches, mais bien souvent leurs prestations s'arrêtent là pour des raisons conjoncturelles. Les informaticiens prennent alors le relais sans qu'une réelle capitalisation des connaissances n'ait été mise en œuvre. Malheureusement aucun outil ne contre aujourd'hui cette difficulté. De prestations très ciblées, ils couvrent le processus de développement de façon parcellaire. Aussi, les tâtonnements successifs restent-ils d'actualité pour converger vers une IHM de qualité.

Dans ce contexte, l'objectif de VERBATIM est d'étudier les liens entre l'ergonomie et en particulier les recommandations ergonomiques et les approches formelles du Génie Logicielle. Dans le temps imparti du projet, nous avons choisi trois approches formelles que nous avons appliquées en vue d'étudier l'utilisabilité de systèmes multimodaux, utilisabilité déclinée sous la forme de recommandations ergonomiques.

Dans ce document, nous rappelons les différences de perspectives entre l'Ergonomie et le Génie Logiciel, et nous exposons l'approche adoptée pour l'intégration basée sur des propriétés. Nous définissons les propriétés de l'interaction multimodale que nous avons considérées dans toutes les approches formelles. Nous concluons en situant les approches formelles au sein d'un cycle de vie pour en montrer leurs portées.

## 1.2 Historique

| Date | Version | Commentaire |
|---|---|---|
| 05/01/07 | 1.0 | Création/rédaction du document par L. Nigay et B. d'Ausbourg |

## 1.3 Glossaire

| Terme | Définition |
|---|---|
| LIG | LIG : Laboratoire d'Informatique de Grenoble<br>Laboratoire créé le 1 janvier 2007 et inclut les anciens laboratoires CLIPS et LSR |
| CARE | CARE (Complémentarité, Assignation, Redondance, Equivalence).<br>Nom de quatre propriétés qui caractérisent l'existence ou non de choix de modalités offerts à l'utilisateur ainsi que la combinaison de modalités. |
| IIHM | Ingénierie de l'interaction Homme-Machine.<br>Nom d'une équipe au sein du laboratoire LIG |

## 1.4 Références


| Abréviation | Référence |
|---|---|
| [ABO 95] | Abowd, G., Coutaz, J., Nigay, L., Structuring the Space of Interactive System Properties, Engineering for Human-Computer Interaction, Larson J. & Unger C. (eds), Elsevier Science Publishers B.V. (North-Holland), IFIP, 1992, pp 113-126. |
| [BAR 95] | Barthe, M., Ergonomie des logiciels, une nouvelle approche des méthodologies d'informatisation, MASSON, 1995, 191 pages. |
| [BRA 90] | Brangier, E., Ergonomie des logiciels : approche psycho-ergonomique de l'interaction homme-ordinateur, Etude bibliographique, Cahiers de notes documentaires n° 139, 2ème trimestre 1990, ND 1780-139-90, INRS, pp 391-404. |
| [CAI 93] | Cail, F., Présentation d'information sur écran de visualisation, Revue bibliographique, Cahiers de notes documentaires n° 151, 2ème trimestre 1993, ND 1928-151-93, INRS, pp 305-311. |
| [COU 95] | Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties. Conference Proceedings of INTERACT'95, S. A. Arnesen & D. Gilmore Eds., Chapman&Hall Publ., Lillehammer, Norway, june 1995, pp. 115-120. |
| [GIL 95] | Gilmore, D.J., Interface Design : Have we got it wrong ?, Human Computer Interaction, Interact'95, Nordby K., Helmersen P.H., Gilmore D.J., Arnesen S.A. (Eds), Chapman and Hall Press, 1995, pp 173-178. |
| [MCC 77] | McCall, J., Factors in Software Quality, General Electric Ed., 1977. |
| [NIE 93] | Nielsen, J., Usability Engineering, Academic Press Professional, 1993, 362 pages. |

| Abréviation | Référence |
|---|---|
| [NIG 97] | Nigay, L., Coutaz, J. Multifeature Systems: The CARE Properties and Their Impact on Software Design. Intelligence and Multimodality in Multimedia Interfaces, AAAI Press, 1997. |
| [NIG 06] | Nigay, L., Coordination par études de cas, Document VERBATIM SP6 lot 2, 2006, 14 pages. |
| [OVI 99] | Oviatt, S. Ten Myths of Multimodal Interaction. Communications of ACM, 42, 11, 1999, pp. 74-81. |

# 2. Ergonomie et Approches formelles du Génie Logiciel

L'ergonomie a été définie en 1988 par la Société d'Ergonomie de Langue Française comme "la mise en œuvre de connaissances scientifiques relatives à l'homme, et nécessaires pour concevoir des outils, des machines et des dispositifs qui puissent être utilisés avec le maximum de confort, de sécurité et d'efficacité pour le plus grand nombre". L'ergonomie puise ses fondements dans la diversité et la variabilité des êtres humains et des situations d'usage. Dans ces diversité et variabilité, des déviations peuvent apparaître entre, d'une part, les tâches prescrites et, d'autre part, les tâches réelles (ou effectives). L'ergonomie vise à réduire ces écarts par la proposition de systèmes ajustés à l'homme. Elle s'appuie, pour ce faire, sur les apports des Sciences Humaines, les rassemble et les contextualise pour une applicabilité directe en ingénierie de l'interaction homme-machine. En ce sens, l'ergonomie peut être considérée comme un relais des Sciences Humaines auprès des informaticiens.

Dans sa quête de compatibilité homme-machine, l'ergonomie travaille à trois niveaux :

- au niveau physique, pour une adéquation entre l'homme d'un point de vue morphologique et son poste de travail ;
- au niveau social, pour un environnement de travail compatible avec les attentes des utilisateurs ;
- au niveau cognitif, pour un traitement et une représentation des informations conformes aux attentes et capacités des utilisateurs.

L'ergonomie des logiciels porte sur ce dernier aspect. La compatibilité y est recherchée à trois niveaux : perceptivo-moteur, linguistique et au niveau global de l'activité [BRA 90].

- Le niveau perceptivo-moteur concerne les dispositifs d'interaction. Il s'agit d'en garantir la compatibilité avec, d'une part, les caractéristiques physiques de l'utilisateur, d'autre part, la nature de sa tâche. Typiquement, une tablette graphique est plus appropriée à une tâche de dessin que le traditionnel triplet écran / clavier / souris.
- Le niveau linguistique porte sur le codage des informations. Il s'agit d'éviter les codages arbitraires qui compliquent la mise en correspondance entre les mondes physique et psychologique.
- Au niveau global de l'activité, il s'agit d'assurer une structure logicielle adaptée et adaptable aux modes de raisonnement de l'utilisateur dans la réalisation de sa tâche.

Pour chaque niveau, des recommandations ergonomiques ont été formulées. Par exemple, pour l'aspect perceptif, [CAI 93] propose une revue bibliographique traitant de la présentation des informations sur écran de visualisation. Dans le sous-projet 5, des recommandations ergonomiques pour l'interaction multimodale sont proposées.

La généralité et le volume de ces ouvrages rendent difficile leur application. Aussi, des propriétés ergonomiques organisant ces recommandations ont-elles été proposées. Dans le paragraphe 3, nous présentons les propriétés que nous avons adoptées dans VERBATIM.

Ces recommandations et propriétés guident les informaticiens dans leurs différentes activités :

- en conception - on parle alors d'ergonomie de conception : cette forme d'ergonomie proactive est récente, l'ergonomie ayant longtemps été considérée comme une "opération cosmétique de dernière heure" [BAR 95] ;
- lors d'une évolution - on parle alors d'ergonomie d'aménagement ;
- pour la correction d'une anomalie détectée en exploitation - on parle alors d'ergonomie de correction.

Dans le sous-projet 3, l'approche formelle adoptée par affinement successif de modèles B exploite les propriétés en conception. Dans les sous-projets 2 et 4, les propriétés sont utilisées pour l'évaluation et la correction d'anomalies.

Nous résumons à la figure 1, le lien établi entre l'Ergonomie et les approches formelles du projet VERBATIM.

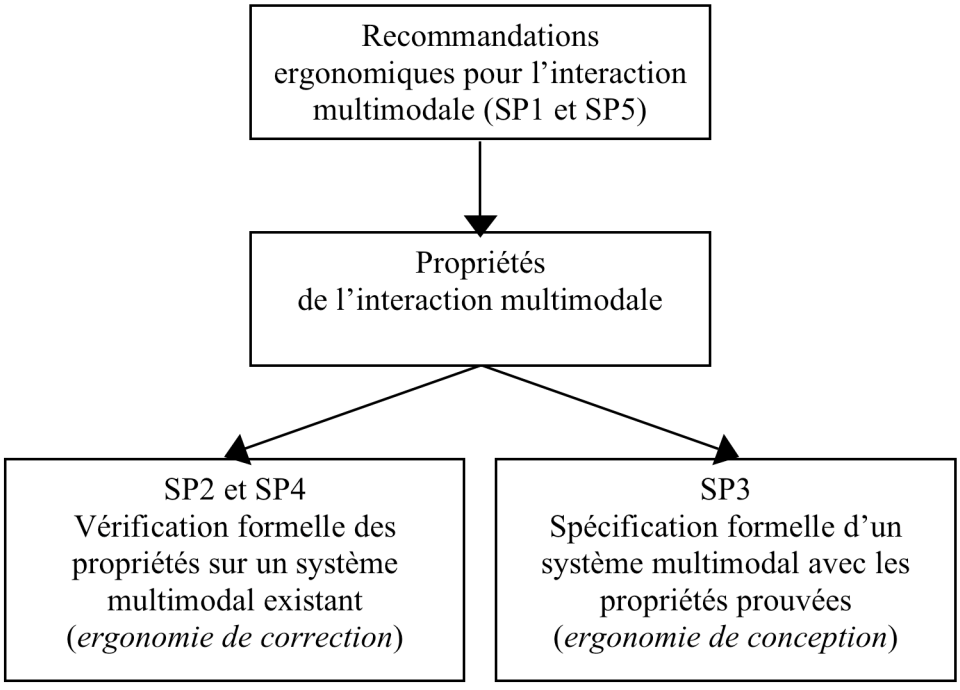


**Figure 1** : Ergonomie et approches formelles du Génie Logiciel.

# 3. Propriétés de l'interaction multimodale

## 3.1 Utilité et Utilisabilité

L'utilité et l'utilisabilité traitent de l'adéquation homme-machine selon une perspective anthropocentrée :

- l'utilité, à connotation fonctionnelle, rappelle aux informaticiens une règle de bon sens, à savoir : n'offrir à l'utilisateur que des services utiles au regard des tâches qu'il aura à accomplir. L'utilité cible une adéquation du système par rapport aux besoins de l'utilisateur ;
- l'utilisabilité relève de l'usage : elle traite de l'adéquation du système par rapport aux capacités cognitives, motrices et sensorielles de l'utilisateur. Elle porte sur le rendu (ou présentation) des services à l'utilisateur.

Utilité et utilisabilité sont des perspectives restreintes de considérations plus générales relatives à l'acceptation d'un système [NIE 93]. L'acceptation d'un système se réfère à son adéquation vis-à-vis des besoins et exigences des utilisateurs. Deux dimensions entrent en jeu : l'aspect social ; l'aspect pratique. C'est dans ce deuxième volet qu'interviennent l'utilité et l'utilisabilité, fédérées en la notion de serviabilité. La serviabilité ou « usefulness » du système se réfère à la faculté du système à permettre à l'utilisateur d'atteindre ses buts [NIE 93]. L'utilisabilité en qualifie les facilités d'apprentissage et d'appropriation, l'efficacité à l'usage, la robustesse aux erreurs et enfin, de façon plus subjective, le plaisir éprouvé à l'usage. L'utilisabilité ne se limite donc pas à des critères de performance dans l'accomplissement de tâches. Elle se réfère, de façon plus générale, à la satisfaction de buts personnels et collectifs [GIL 95].

Si depuis 1977, l'utilisabilité est reconnue à part entière comme facteur qualité logicielle [MCC 77], elle a depuis été ré-évaluée en catégorie de facteurs [ABO 92]. Elle regroupe ainsi trois facteurs qualité : la facilité d'apprentissage (learnability), la souplesse et la robustesse de l'interaction (Interaction flexibility and robustness). Elle regroupe ainsi trois facteurs qualité : la facilité d'apprentissage (learnability), la souplesse et la robustesse de l'interaction (Interaction flexibility and robustness). La table suivante en rappelle les définitions.

| **Catégorie** | **Facteurs** | **Définitions** |
|---|---|---|
| Utilisabilité | Facilité d'apprentissage | Facilité avec laquelle de nouveaux utilisateurs interagiront de façon efficace avec le système, selon une performance maximale. |
| | Souplesse de l'interaction | Eventail des choix laissés à l'utilisateur et le système dans les moyens d'interaction. |
| | Robustesse de l'interaction | Caractéristiques de l'interaction favorisant l'accomplissement et l'évaluation des buts. |

## 3.2 Propriétés CARE de l'interaction multimodale

L'interaction multimodale participe aux trois facteurs de l'utilisabilité : facilité d'apprentissage , souplesse et robustesse de l'interaction. Les intérêts attendus de la multimodalité sont de plusieurs ordres : par exemple rendre l'interaction homme-machine plus efficiente, plus souple et plus robuste [OVI 99]. Cet aspect de souplesse est particulièrement important : la liberté laissée à l'utilisateur de choisir une modalité ou une combinaison de modalités la plus efficace et la plus adaptée à un contexte d'utilisation évolutif (propriété inhérente aux situations de mobilité) constitue une condition déterminante de l'acceptation et de l'utilisation effective du système. Un autre avantage attendu de l'utilisation de ces interfaces est qu'ils facilitent la récupération d'erreurs en permettant par exemple aux utilisateurs de changer de modalité lorsque l'usage d'une modalité donnée génère des incompréhensions entre l'utilisateur et le système. Par ailleurs, on s'attend à ce que ces systèmes soient plus efficaces dans l'interprétation des actions de l'utilisateur grâce notamment à la « désambiguisation mutuelle » qui se réfère à la capacité du système à intégrer et confronter les informations provenant des différents modules d'interprétation des données issues des différentes sources d'entrée (modalités).

Nous proposons dans [COU 95] [NIG 97] quatre propriétés notées CARE (Complémentarité, Assignation, Redondance et Equivalence) qui caractérisent l'interaction mulitmodale. Nous les définissons en considérant les deux niveaux d'une modalité d'interaction : Dispositif et Langage d'interaction. Le langage d'interaction définit la forme d'une expression tandis que le dispositif physique en définit la substance.

La figure 2 montre les relations possibles entre dispositifs, langages d'interaction et tâches. Ces relations peuvent être permanentes ou temporaires, totales ou partielles. Une relation est permanente si elle est vérifiée quel que soit l'état du système ; elle est temporaire dans le cas contraire. Une relation entre un dispositif et un langage est totale si elle est vérifiée pour toutes les expressions de ce langage (sinon, elle est dite partielle). Une relation entre un langage et les tâches que le système permet d'accomplir est totale si elle tient pour toutes les tâches (sinon, elle est partielle).

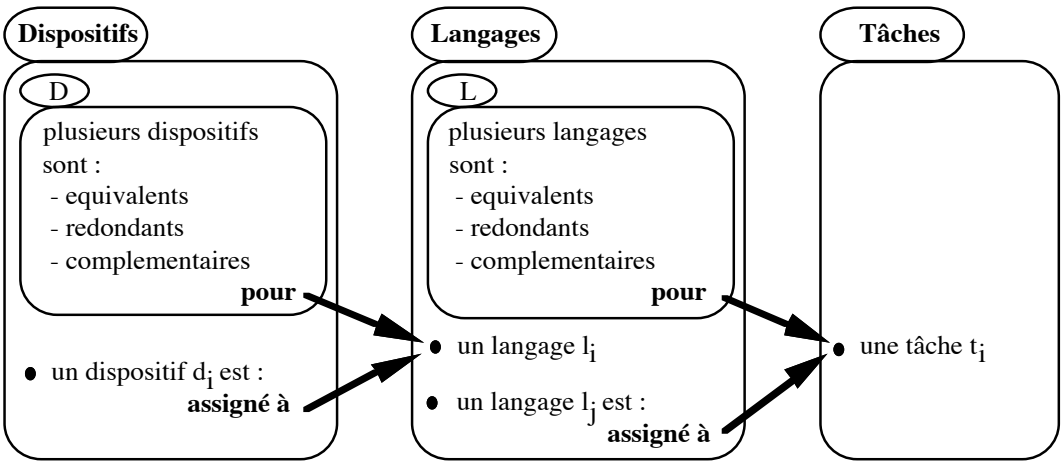


**Figure 2** : Propriétés CARE (Complémentarité, Assignation, Redondance et Equivalence) entre dispositifs, langages d'interaction et tâches.

Nous définissons les propriétés CARE, en commençant par les relations entre langages et tâches puis entre dispositifs et langages. Nous les illustrons avec l'étude de cas MATIS [NIG 06] du projet VERBATIM.

Nous notons *s* un état du système, ***D*** l'ensemble des dispositifs physiques de même direction (entrée ou sortie) offerts par le système, D un sous-ensemble non vide de ***D***, *d* un élément de D ou ***D***, ***L*** l'ensemble des langages d'interaction de même direction offerts par le système, *L* un sous-ensemble non vide de **L**, *l* un élément de L ou de **L**, ***T*** l'ensemble des tâches élémentaires réalisables avec le système, *T* un sous-ensemble non vide de **T**, et *t* un élément de T.

*Equivalence de langages pour une tâche t : L-Equivalence (L, s, t)*

Les langages d'interaction de L sont équivalents dans l'état s du système pour la tâche t, si t peut être exprimée au moyen de l'un quelconque des langages de L. *L-Equivalence (L, s, t)* est permanente si elle vraie quel que soit s. Elle est totale si elle est vraie pour toutes les tâches de **T**. Dans MATIS, l'utilisateur peut spécifier ses demandes de renseignement sur les horaires d'avion en utilisant la manipulation directe ou le langage naturel. Cette relation est permanente (la demande de renseignement peut toujours se faire via le langage naturel ou la manipulation directe) mais elle est partielle : elle ne tient que pour la tâche de demande de renseignements.

*Assignation de langage à une tâche : L-Assignation (l, s, t)*

Un langage d'interaction l est assigné à la tâche t dans l'état s, s'il n'existe pas de langages l'≠l tel que L-Equivalence (l', s, t). En d'autres termes, t ne peut être exprimée qu'au moyen de l. Dans MATIS, il y a L-Assignation permanente du langage de manipulation directe pour les tâches de gestion des fenêtres (changement de taille, fermeture, etc.). Cette assignation n'est que partielle puisque pour la demande de renseignement, il est possible d'employer aussi le langage naturel.

*Complémentarité de langages pour l'expression d'une tâche : L-Complémentarité (L, s, t)*

Les n langages l1, l2, ln de L sont complémentaires dans l'état s pour l'expression de la tâche t si l'expression de t peut être partitionnée en n sous-expressions exp1, exp2, expn, telles que chaque sous-expression expi est construite dans un et un seul langage li de L : L-assignation (li, s, expi). Les coréférences entre langages illustrent cette situation. Dans MATIS, le langage naturel et la manipulation directe sont partiellement complémentaires mais de manière permanente : cette relation tient quel que soit l'état mais seulement pour les tâches de spécification des champs d'une requête. Articuler la phrase *flights from this city* ou *flights from* accompagné de la sélection de la ville au moyen de la souris est un exemple de complémentarité entre expressions de langages distincts. Les deux expressions sont nécessaires à la constitution de la tâche de spécification "lieu de départ". Notons que la complémentarité autorise un usage synergique ou alterné de langages : il y a toujours combinaison mais les expressions peuvent être produites en parallèle ou de manière séquentielle.

*Redondance de langages dans l'expression d'une tâche : L-Redondance (L, s, t)*

Dans un état s, il y a redondance entre les langages de L, si ces langages sont L-équivalents sur s et t et s'ils peuvent être utilisés simultanément ou de manière séquentielle mais contrainte par une fenêtre temporelle. Dans MATIS, la redondance est permanente entre le langage naturel et le langage de manipulation directe mais elle est partielle : elle ne tient que pour les tâches de demande de renseignement. L'énoncé *flights to Boston* et la saisie concurrente et équivalente de *Boston* dans le champ de destination d'un formulaire donne naissance à un seul exemplaire de requête contenant Boston.

Nous traitons maintenant les relations entre les dispositifs et les langages.

*Equivalence de dispositifs pour une expression el du langage l : D-Equivalence (D, s,el, l)*

Les dispositifs physiques de D sont équivalents dans l'état s pour construire l'expression el de l, si el peut être élaborée au moyen de l'un quelconque des dispositifs de D. *D-Equivalence (D, s, el, l)* est permanente si elle vraie quel que soit s. Elle est totale si elle est vraie pour toutes les expressions de l. Dans MATIS, l'utilisateur peut construire des expressions en langage naturel en utilisant soit le clavier soit le microphone. Cette D-equivalence est permanente mais partielle : il n'est pas possible de corriger les erreurs d'énonciation via le microphone (par exemple l'énoncé oral "flights from ... no! I mean : to" n'est pas autorisé alors que la correction d'erreurs de frappe est permise).

*Assignation de dispositif à une expression el du langage l : D-Assignation (d, s, el, l)*

Un dispositif d est assigné à l'expression el de l dans l'état s, s'il n'existe pas de dispositifs d'≠d tel que D-Equivalence (d', s, el, l). En d'autres termes, el ne peut être construite qu'au moyen de d. L'assignation est totale si elle est vraie pour toutes les expressions de l. Dans MATIS, il y a D-Assignation permanente et totale de la souris au langage de manipulation directe.

*Complémentarité de dispositifs pour la construction d'une expression el du langage l : D-Complémentarité (D, s, el, l)*

Les n dispositifs d1, d2, dn de D sont complémentaires dans l'état s pour la construction de l'expression el de l si el peut être partitionnée en n sous-expressions expl1, expl2, expln, telles que chaque sous-expression expli est construite au moyen d'un et un seul dispositif di de D : D-assignation (di, s, expli, l). La complémentarité est totale si elle tient pour toutes les expressions de l.

*Redondance de dispositifs dans la construction d'une expression el du langage l : D-Redondance (D, s, el, L)*

Dans un état s, il y a redondance entre les dispositifs de D, si ces dispositifs sont D-équivalents sur s et el, et s'ils peuvent être utilisés simultanément. Un tel fonctionnement s'applique par exemple en vision stéréoscopique. Un autre exemple possible est le montage

"redondant" d'un microphone et d'une caméra qui observe le mouvement des lèvres pour augmenter la robustesse d'un système de reconnaissance de la parole.

Il convient de noter que les propriétés CARE s'appliquent aussi à un niveau de granularité plus élevé en considérant une modalité comme le couple dispositif et langage. Les propriétés CARE définissent alors des relations entre modalités vis-à-vis des tâches que l'utilisateur peut effectuer. Dans le projet VERBATIM, bien que les deux concepts de dispositif et langage sont exploités, nous appliquons les propriétés CARE au niveau des modalités.

## 3.3 Propriétés CARE : souplesse et robustesse

A l'évidence, les propriétés CARE permettent de comparer des interfaces en termes de langages d'interaction, de dispositifs et de leurs relations. Le deuxième intérêt des propriétés CARE, central dans le projet VERBATIM, est leur application à l'évaluation ou à la conception des nouvelles interfaces. Nous les corrélons aux deux propriétés de souplesse et robustesse de l'utilisabilité (paragraphe 3.1).

L'équivalence et l'assignation caractérisent la portée des choix en matière de langage et de dispositif. La première joue en faveur de la souplesse ou de la robustesse (par exemple, dans un milieu bruité, le clavier peut être substitué au microphone), la seconde augmente le caractère préemptif du système (l'utilisateur n'a pas de choix). La redondance et la complémentarité qualifient la combinaison des choix. La première intervient, selon l'intention communicationnelle, comme un facteur de souplesse ou de robustesse. La seconde, qui impose un usage complémentaire de langages et/ou de dispositifs, peut se voir comme un facteur contraignant. Sous cet angle, le paradigme "mets ça là" souvent cité comme modèle d'interaction "naturelle", apparaît d'une utilisabilité douteuse : il requiert la complémentarité de deux langages mais aussi l'assignation de deux dispositifs. En vérité, des expérimentations par Magicien d'Oz montrent que les sujets en phase exploratoire utilisent volontiers l'équivalence et la redondance mais une fois les capacités du système cernées, une large place est faite à l'assignation.

Les caractères partiel et temporaire d'une relation CARE peuvent aussi servir de guide prédictif : une relation qui n'est pas totale ou qui n'est pas permanente indique que l'utilisateur devra se souvenir des contextes particuliers dans lequel elle s'applique.

## 3.4 Conclusion : CARE et les approches formelles du projet

En conclusion, les propriétés CARE que nous avons adoptées comme traduction des recommandations ergonomiques fournissent un cadre cohérent et suffisant si l'on s'en tient à l'aspect fonctionnel (tâche) du système. Comme souligné dans le paragraphe 3.1, il

conviendrait de considérer d'autres propriétés d'utilisabilité. Aussi il s'agit d'une traduction réductrice des recommandations ergonomiques. Par exemple toutes les recommandations du SP1 et SP5 ne sont pas catégorisables en termes des propriétés CARE. D'autres propriétés sont à considérer, plus fines au niveau des dispositifs physiques comme au niveau des langages afin de mieux cerner les coûts système mais aussi les coûts cognitifs, conatifs et affectifs.

Les trois approches formelles de VERBATIM (SP2, SP3 et SP4) considèrent les propriétés CARE comme des relations entre modalités ; ces dernières servent de point d'ancrage entre les recommandations ergonomiques et les approches formelles exploitées en conception et en évaluation. Dans le paragraphe suivant, nous situons les trois approches formelles au sein d'un cycle de vie afin d'en souligner leurs portées.

# 4. Cycle de vie et approches formelles de VERBATIM

Le modèle en V convient au développement de systèmes dont la finalité est bien cernée. Pour les systèmes à risque, on adoptera le modèle en spirale qui conduit, par itérations et incréments, de l'analyse à une solution satisfaisante. En pratique, le modèle en V n'est pas suivi à la lettre. Nous le retenons ici pour les besoins analytiques de l'exposé. La figure 3 en rappelle les étapes et souligne l'ancrage de l'ergonomie dans le processus de développement.

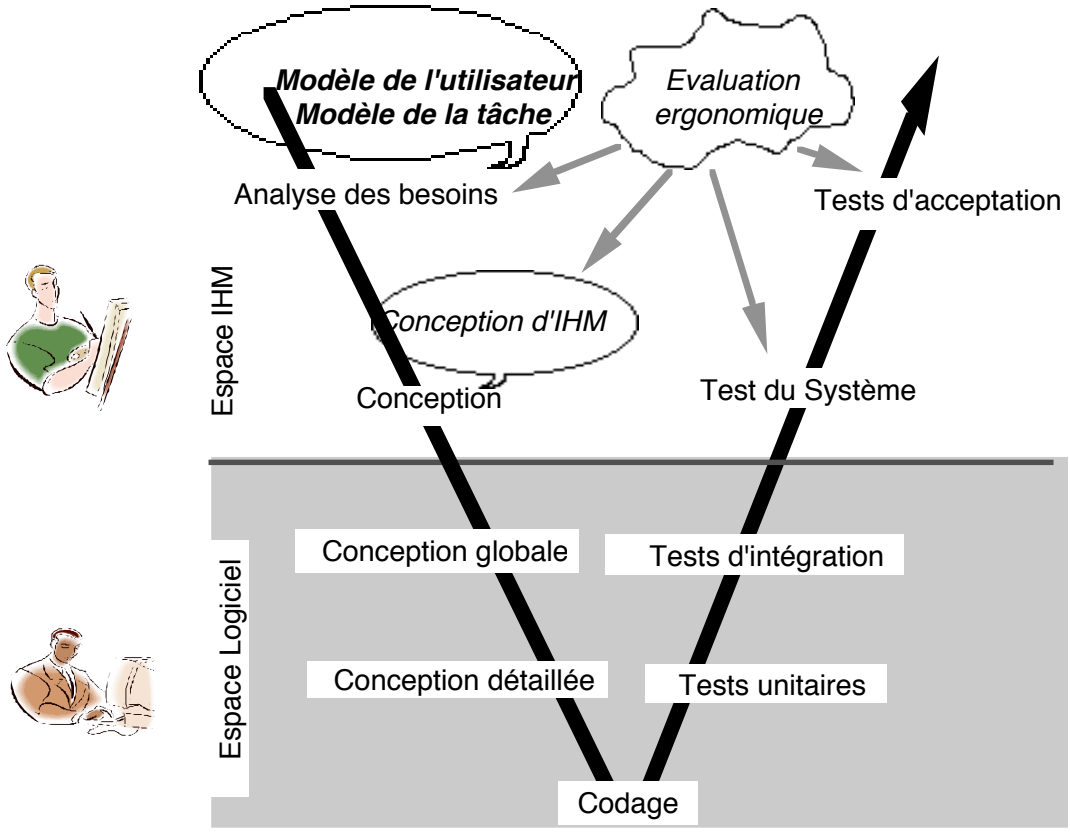


**Figure 3** : Cycle en V et ancrage de l'ergonomie.

La figure 3 montre la structuration du processus de développement en étapes. La pente descendante du V couvre les étapes de réification progressive du système depuis la définition des besoins jusqu'à son incarnation logicielle. Sur la pente ascendante, à chacune des étapes de réification correspond un ensemble de tests qui permettent de vérifier et/ou de valider l'étape en regard du code produit.

L'analyse des besoins permet d'établir, en relation avec le client, les services requis du système et les contraintes de développement. Cette activité donne lieu à un cahier des charges qui sert de document contractuel entre le maître d'œuvre et le client.

La conception du système consiste à définir une solution matérielle et logicielle qui répond à l'analyse des besoins et aux contraintes. Cette activité produit des spécifications externes. Dans le cas qui nous intéresse, ces spécifications décrivent l'interface utilisateur du système.

La conception globale constitue l'étape préliminaire orientée vers la mise en œuvre logicielle : nous quittons l'espace de conception de l'IHM pour pénétrer dans l'espace logiciel

réservé aux informaticiens. La conception globale donne lieu à un dossier d'architecture générale.

La conception détaillée décrit les choix algorithmiques, précise la signature des procédures et des fonctions, définit les structures de données les plus pertinentes et spécifie les protocoles de communication. C'est le monde privilégié de l'analyste programmeur. Ici, l'ergonome n'a plus sa place ! Il n'est plus consulté alors que des décisions logicielles dirigées par des considérations techniques peuvent avoir un impact néfaste ou imprévisible sur l'utilisabilité.

Puis intervient le codage dans un langage de programmation avant d'entreprendre les tests unitaires. Ces derniers permettent de vérifier que les composants modulaires du système répondent chacun à leurs spécifications. Les tests du système servent à vérifier que les éléments de la solution exprimés dans le dossier de spécifications externes sont présents. Ici, nous pénétrons à nouveau dans un espace commun à l'ergonome et au développeur.

Dans ce cycle en V de la figure 3, nous soulignons deux espaces :

• L'espace IHM se caractérise par la priorité qu'il faudrait accorder aux aspects ergonomiques et inclut en particulier l'analyse des besoins et la conception de l'interface consignée dans le document de spécifications externes. A cette étape de conception, intervient l'ergonomie pour la définition et la spécification de l'interface utilisateur

• L'espace logiciel, lié aux techniques d'implémentation logicielles, laisse la place aux compétences informatiques avec les conceptions globales et détaillées, le codage et les tests unitaires et d'intégration.

Comme le schématise la figure 4, dans le projet VERBATIM, nous étudions des propriétés dites externes d'utilisabilité (espace IHM) par des approches formelles du Génie Logiciel initialement utilisées pour vérifier des propriétés internes du Génie Logiciel (espace Logiciel).

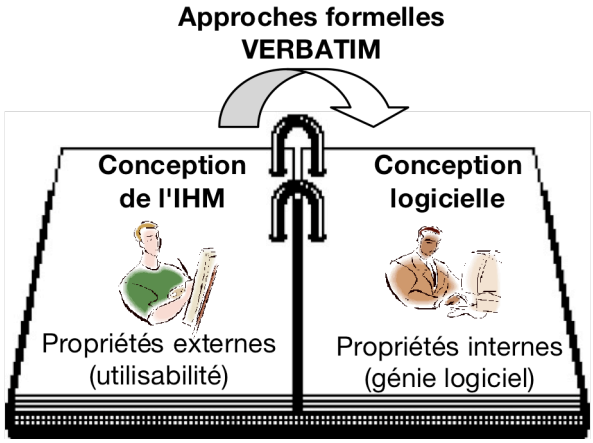


**Figure 4** : Approches formelles pour des propriétés externes d'utilisabilité et en particulier les propriétés CARE.

Les trois approches formelles considérées dans le projet VERBATIM couvrent des parties différentes du cycle de vie :

- Dans SP2, la technique de tests étudiée vise l'étape de tests du système de la figure 3. Cette approche ne se substitue pas à l'évaluation ergonomique expérimentale, mais permet d'identifier un premier ensemble de problèmes. Dans VERBATIM, nous étudions les propriétés CARE et d'autres aspects d'utilisabilité (paragraphe 3) sont à considérer par une approche expérimentale.

- Dans SP3, l'approche adoptée concerne la phase de conception de l'IHM de la figure 3, puis par affinement successif à obtenir le code du système interactif.

- Dans SP4, la technique d'analyse statique du code permet aussi de faire des tests et en cela concerne aussi l'étape de tests du système. Cependant l'approche adoptée consiste à obtenir un modèle d'exécution et donc permet d'obtenir une abstraction du code qui décrit le comportement interactif du système : phase de conception de l'IHM de la figure 3.

# 5. Conclusion

Plusieurs approches fondées sur l'utilisation de techniques formelles distinctes ont été mises en œuvre dans les différentes études de cas abordées dans le contexte du projet VERBATIM [NIG 06]. Ces approches diffèrent par leur sémantique, par leur système de preuve, par les modèles qu'elles exploitent (programmes ou spécifications) etc. Se pose alors le problème de l'hétérogénéité. L'intégration des techniques dans le projet repose sur l'exploitation des propriétés CARE. Ainsi les modèles des propriétés CARE sont communs aux différentes techniques. De plus les propriétés CARE servent de point d'ancrage entre l'Ergonomie et les approches du Génie Logiciel considérées.

Nous avons souligné enfin les différences des approches fondées sur des techniques formelles en les situant au sein d'un cycle de vie. Cette étude a souligné leurs différences (conception, évaluation) mais a aussi permis de souligner des points de rencontre au sein du cycle de vie. Aussi nous entrevoyons plusieurs liens à établir entre les trois approches considérées (SP2, SP3 et SP4).

**Liens entre SP2 et SP4 :**
Dans le cadre du sous-projet SP4, l'analyse des codes Java des applications interactives permet de produire un modèle d'exécution qui, outre la description du comportement interactif du programme, comporte également une description du comportement de l'environnement de ce programme. Cet environnement d'exécution est constitué à la fois de la machine virtuelle Java interprétant le programme et de l'utilisateur réalisant les entrées de ce programme, traitées comme des événements d'entrée de la machine virtuelle.
Les analyseurs du code Java ont la charge de produire ce modèle d'exécution. Lors de la construction de ce modèle, et plus particulièrement lors de la construction du modèle d'environnement, ils introduisent des directives de génération de scénarios de tests sous la forme de l'invocation de méthodes du type *generateScenarioStep (String s)*. Ces directives permettent d'introduire des commandes de traces dans les modèles formels Promela qui sont générés et qui font l'objet de vérifications par model checking ou d'exécution simulée. Le contenu de la trace est celui de la chaîne de caractères *s* fournie comme paramètre dans l'invocation des méthodes *generateScenarioStep (s)*.
Le modèle d'exécution généré par analyse des codes comporte systématiquement une telle directive lorsqu'il décrit une action réalisée par l'utilisateur. Cette action est repérée dans le programme par le traitement d'événement qu'elle génère. Elle correspond à une entrée de l'utilisateur à travers l'un des objets d'interaction proposé par le programme : bouton, champ de saisie, menu, image graphique, etc.
Le mécanisme présenté par l'introduction de ces directives peut être exploité de différentes façons dans le cadre des tests réalisés avec Lutess. Une première utilisation possible consiste à produire des préfixes pour guider les tests de la manière suivante. On peut vouloir en effet conduire des tests de l'application à partir d'une configuration d'état particulière de cette application. Pour trouver comment l'atteindre, on cherche à vérifier sur le programme (sur le modèle d'exécution du programme) qu'elle n'est pas atteignable. Les contre-exemples produits sont exécutés de manière simulée et produisent alors les séquences de traces *s* introduites dans le modèle d'exécution par le biais des directives *generateScenarioStep (String s)*. Ces

séquences de chaînes *s* constituent le démarrage d'un scénario et reflètent les données d'entrée du programme, produites par les actions de l'usager, qui conduisent à la configuration recherchée. Le test peut ensuite se poursuivre de manière classique sous Lutess. On peut aussi utiliser, dans le cadre de Lutess, les scénarios générés comme des « patterns » ou des scénarios incomplets, à trous, indiquant des valeurs d'entrées successives à fournir au programme sous test. La figure 5 schématise ce lien entre SP2 et SP4.

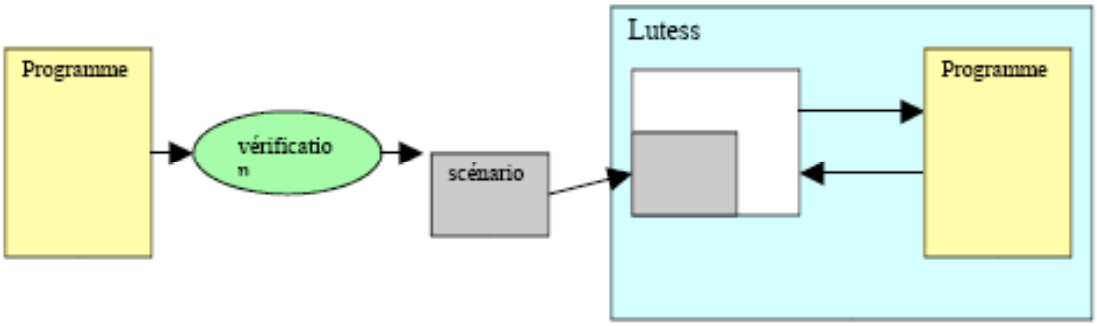


**Figure 5** : Lien entre les deux approches des sous-projets SP2 et SP4.

**Liens entre SP3 et SP4 :**
Le lien entre SP4 et SP3 a déjà commencé à être étudié en dehors du cadre du projet VERBATIM : en effet ce travail fait l'objet d'une thèse co-encadrée par l'ONERA et le LISI.
n'a pas directement été exploité dans le cadre du projet Verbatim.

L'un des travaux effectués dans le cadre du sous-projet SP3 a trait à la formalisation en B de modèles de tâches exprimés en CTT. On fait l'hypothèse que les modèles CTT peuvent contenir, en compréhension, un ensemble de scénarios abstraits (non dépendants des modalités d'interaction) d'utilisation du système interactif à développer. Ces scénarios sont abstraits dans la mesure où ils expriment des ordonnancements possibles de tâches utilisateur. Chacun de ces scénarios abstraits peut être réalisé. La réalisation concrète de ces scénarios abstraits peut revêtir un ensemble très étendu.
Le code, Java en l'occurrence, du programme de l'application déclare un ensemble d'instructions et de méthodes exécutées en réponse aux sollicitations concrètes de l'usager qui peuvent être exercées par le biais d'entrées introduites en agissant sur les différents dispositifs d'interaction qui sont offerts. Les séquences que constituent ces entrées successives de l'usager constituent autant de traces dont on peut dire qu'elles déroulent des scénarios concrets possibles d'utilisation du système d'interaction. Autrement dit, le code de l'application décrit de façon concrète l'ensemble des scénarios d'utilisation du système qu'il peut accepter, vis-à-vis desquels il réagit.
Il peut donc être intéressant de vérifier que les scénarios concrets que peut traiter le code implantant le système sont effectivement la concrétisation des scénarios abstraits que décrit le modèle de tâches CTT. En d'autre termes, on produit une forme de validation du programme en vérifiant que les scénarios d'utilisation que le code accepte font partie des scénarios qui pourraient constituer une concrétisation des scénarios abstraits englobés dans les modèles de tâches CTT, qu'ils en constituent un possible raffinement.

Dès lors, et très schématiquement, la démarche envisagée consiste à opérer un double mouvement d'abstraction et de concrétisation (raffinement). Un premier travail de formalisation en B est effectué sur les modèles CTT pour en produire une formulation en B. On raffine alors le modèle obtenu en introduisant des éléments d'interaction concrète permettant d'exprimer comment certaines opérations sont réalisées par l'utilisateur. On peut appeler ce modèle $M_T$. Parallèlement à cela, un travail d'abstraction est opéré sur les codes java de l'application. Il consiste à extraire de ce dernier les éléments relatifs à l'interaction et à produire un modèle d'exécution de l'application. Ce dernier est alors formalisé en B également pour produire un modèle $M_P$ du programme de l'application. Les outils de preuve associés à B sont alors utilisés pour démontrer que $M_P$ est un raffinement correct de $M_T$.