Mise en œuvre d'IHM multimodales dans un système de CAO. Une approche fondée sur les méthodes formelles

Nadjet KAMEL[+*] & Yamine AIT AMEUR[*]

+Laboratoire de Recherche en Intelligence Artificielle
Université des Sciences et de la Technologie Houari Boumedien
BP N °32, 1611, El Alia, BabEzzouar, Alger, ALGERIE
nkamel@usthb.dz

**Laboratoire de l'Informatique Scientifique et Industrielle*
Ecole Nationale Supérieure de Mécanique et d'Aéronautique
BP 40109, 86961, Futuroscope Cedex, FRANCE
{kamel ,yamine}@ensma.fr

RÉSUMÉ. *Nous montrons dans cet article l'intérêt de la modélisation formelle de l'interaction multimodale dans un système de CAO. Nous avons défini un modèle formel d'interaction permettant la conception de l'interaction multimodale en entrée selon le type d'interface souhaité. C'est un modèle inspiré des algèbres de processus dont la sémantique est donnée par les systèmes de transitions étiquetées. Nous présentons également l'expression de deux propriétés d'utilisabilité en logique temporelle CTL (Computational Tree Logic). Une mise en ouvre de la modélisation de deux types d'IHM pour une application de CAO simple, ainsi que la vérification de ces propriétés, ont été effectués à l'aide du contrôleur sur modèle SMV (Symbolic Model Verifier). SMV est un outil qui permet la vérification de systèmes de transitions dont les spécifications sont exprimées en logique temporelle CTL.*

ABSTRACT. *This paper addresses the use of formal modelling of multimodal interaction within a CAD system. We define a formal model for the multimodal interaction mainly input multimodal interactions. This model allows the design of multimodal interaction according to the kind of the multimodal interface. It is inspired from process algebra model with transition systems as semantics. We present the expression of the usability properties by CTL (Computational Tree Logic) temporal logic formula. A modelisation of two kinds of multimodal interfaces for a toy CAD system is checked using the SMV (Symbolic Model Verifier) model-checker. SMV is a tool allowing to check finite state transition systems against specification in the temporal logic CTL.*

MOTS-CLÉS : *Interaction Multimodale, Méthodes Formelles,Utilisabilité d'un système de CAO*

KEYWORDS: *Multimodal Interaction, Formal methods, Usability of a CAD system*

1. Introduction

La plupart des systèmes de CAO (Gardan, 1991) actuellement sur le marché, comme CATIA, utilisent des modes d'interaction traditionnels avec souris et manipulation directe[1]. Ainsi, avec ce type de dispositifs, les seules interactions que l'utilisateur humain peut utiliser pour agir sur le système sont fondées sur le texte pour saisir des ordres, des valeurs ou des paramètres, sur l'utilisation de menus déroulants avec sous-menus, sous-sous-menus, etc… et l'utilisation du pointeur de la souris pour désigner et/ou sélectionner des objets géométriques ou d'autres éléments d'interaction comme les menus par exemple. Parfois, il est possible de combiner deux modes d'interaction comme par exemple la souris pour désigner un objet et le clavier pour entrer une valeur.

Les dispositifs d'interactions de type WIMP ont fait leur preuve et sont très largement utilisés dans les systèmes de CAO. Cependant, plusieurs critiques peuvent être formulées ; parmi elles :

- plusieurs interactions doivent être réalisées par un utilisateur pour réaliser et/ou concevoir un modèle géométrique, un modèle de pièce quelconque, etc. ;
- l'utilisation de la souris pour ouvrir une cascade de menus nécessite une connaissance approfondie des chemins d'interactions conduisant à la fonction recherchée, et donc une phase d'apprentissage importante ;
- la réalisation concurrente d'interactions est le plus souvent limitée à la souris et au clavier. L'ordre d'interaction revêt une grande importance.

Les trois critiques précédentes montrent quelques limites ergonomiques des IHM de systèmes de CAO. Afin de palier ces insuffisances, d'autres formes de menus, tels que les Pie menus présentant les items sous une forme circulaire, ont été définies. Elles ont permis d'accélérer l'accès aux différentes fonctionnalités du système, mais elles présentent l'inconvénient d'encombrer l'écran et de limiter le champ de visibilité de l'utilisateur.

Les limites identifiées ci-dessus constituent les principales motivations de notre proposition. En effet, nous considérons que d'autres modes d'interaction connus (comme la voix et le geste) et/ou non encore identifiés pourraient être utilisés dans les systèmes de CAO. La présence de nouveaux modes d'interaction augmente l'efficacité d'utilisation et améliore l'ergonomie des systèmes de CAO. Quelques expériences ont été réalisées en utilisant les croquis pour concevoir des modèles géométriques 3D. Nous citons les systèmes 3D Sketch (Mitani *et al.*, 2002) et Teddy (Igarashi *et al.*, 1999).

Lorsque ces nouveaux modes d'interaction cités précédemment sont introduits dans les IHM, on décrit alors des IHM MultiModales (IHMMM) que nous noterons IHM3 par la suite. En effet, ce type d'IHM offre plusieurs possibilités d'interactions

[1] Généralement appelés modes d'interaction de type WIMP (Windows, Icons, Mouse, Pointers).

que nous nous proposons d'associer aux systèmes de CAO afin d'en faciliter l'apprentissage et l'utilisation. Ainsi, l'ouverture d'une série de menus peut être réalisée avec la voix et par conséquent accélère l'accès aux commandes. Cela permet de libérer la souris pour réaliser d'autres traitements en parallèle lorsque l'interface utilisateur le permet.

Notre travail propose un modèle formel, rigoureux et générique permettant de prendre en compte tout type de modalité en entrée en vue de les fusionner pour déclencher une ou plusieurs fonctions visées. En conséquence, les chemins d'interaction seront réduits (accès rapide aux fonctions du système), l'effort d'apprentissage amoindri et l'ergonomie et la facilité d'utilisation augmentées. L'intégration de ce modèle d'interaction ne nécessite pas la modification de l'architecture des systèmes de CAO dans le cas où ces derniers séparent le traitement de l'IHM de celui de l'application. Notre modèle se place en effet comme un module (ou moteur) de fusion des différentes interactions multimodales en entrée (de l'utilisateur vers le système) pour générer une interaction équivalente (abstraite). Cette interaction abstraite a pour effet de déclencher une fonction du système de CAO. Par exemple, si dans le système une interaction abstraite est prévue pour ouvrir un menu, alors cette dernière peut être réalisée soit par des clics souris soit par une phrase prononcée et reconnue par le système de reconnaissance vocale du module de fusion.

Notons que ce modèle est générique et s'applique à toute IHM3. Nous montrons ici comment il peut être utilisé sur un système de CAO. Nous précisons que la notion de modèle que nous utilisons ne concerne pas les modèles géométriques utilisés classiquement dans le domaine de la CAO mais la notion de modèle formel en spécification des systèmes (programmes). Il s'agit de la notion de modèle classiquement utilisée en logique.

Outre l'extension des IHM3 aux systèmes en CAO, le modèle que nous proposons contribue également à une conception rigoureuse et validée a priori, c'est-à-dire, avant même la réalisation du système. La validation consiste à vérifier l'utilisabilité de l'interface multimodale caractérisée principalement par les propriétés CARE présentées en section 2.3. En effet, nous proposons une formalisation des interactions d'IHM3 qui peut être animée, testée et validée avant d'être intégrée dans un système de CAO déjà conçu ou bien à concevoir.

Cet article est structuré de la façon suivante. La section 2 présente les IHM3, elle fixe leurs concepts de base, l'espace de conception que nous formalisons et leurs propriétés d'utilisabilité qui permettent d'évaluer la qualité d'une IHM3. Nous insistons sur les propriétés qu'un système de CAO doit satisfaire. En section 3, nous présentons un bref panorama des approches formelles utilisées dans ce domaine ainsi que notre proposition. La section 4 est le centre de cet article, elle présente notre modèle formel d'interactions multimodales. Une étude de cas est donnée en section 5. Elle montre sur une application maquette l'intérêt, pour un système de CAO, de disposer de tels modes d'interaction. La section 6 montre l'expression en

logique temporelle CTL (Computational Tree Logic) et la vérification opérationnelle des propriétés d'utilisabilité à l'aide de l'outil SMV. Enfin, nous concluons cet article et présentons les perspectives dégagées à l'issue de ce travail.

2. IHM multimodales (IHM3)

Les IHM3 sont présentes dans plusieurs types de systèmes : systèmes de navigation, systèmes destinés aux handicapés, systèmes de contrôle, etc…. Les interactions dans ce type d'IHM offrent une grande flexibilité à l'utilisateur et facilitent l'utilisation de ces systèmes. En contrepartie, leur caractère concurrent rend leur conception et leur analyse complexes. Elles nécessitent le développement de nouveaux modèles permettant de raisonner sur ce type de systèmes interactifs. Plusieurs travaux ont été effectués pour spécifier les concepts de base des IHM3 et pour définir des espaces de conception et les propriétés d'utilisabilité associées.

2.1. Concepts de base

Plusieurs concepts liés aux IHM3 ont été définis, mais un consensus sur leur définition reste absent. Par exemple, le concept de *modalité* est défini dans (Nigay *et al*., 1995), par la paire *(média, langage d'interaction)* où *média* désigne le dispositif physique utilisé pour acquérir ou fournir les informations, et *langage d'interaction* définit un ensemble d'expressions bien formées et significatives de symboles résultant d'actions sur les dispositifs physiques. (Martin *et al.,* 1996) définissent la modalité comme la manière d'utiliser un *média*. Par exemple le média stylo électronique peut être utilisé selon plusieurs modalités : écriture, geste de dessin, geste de commande. La modalité dans ce cas résulte du processus d'analyse ou de synthèse défini sur des ensembles de données d'entrée et de sortie. Nous retrouvons dans (Nigay *et al*., 1996) une synthèse et une mise au point sur la terminologie et sur les référentiels applicables à la conception d'IHM3. Le concept de *multimodalité* est lié à la multiplicité des modalités. Un système est dit multimodal s'il dispose d'au moins deux modalités pour un sens donné (en entrée ou en sortie). Puisque les actions sont produites par plusieurs modalités alors il peut être, parfois, nécessaire de les composer pour former une commande plus complexe. L'opération qui réalise cette composition est appelée *fusion* et elle est utilisée en général pour les interactions en entrée (de l'utilisateur vers le système). L'opération qui consiste à décomposer une commande en plusieurs actions de modalités différentes est appelée *fission* et elle est utilisée en général pour les interactions en sortie (du système vers l'utilisateur).

2.2. Espaces de conception

Les espaces de conception permettent de caractériser les systèmes en termes utiles au concepteur de logiciel. Ils se fondent sur un ensemble de critères combinés pour offrir un guide de conception ergonomique. L'espace de conception défini dans (Bellik, 1995), repose sur les trois critères : de production des énoncés (séquentielle ou parallèle), d'usage des modalités (exclusif ou simultané) et du nombre de modalités par énoncé. Un *énoncé* correspond à une tâche élémentaire (par exemple placement d'un modèle d'objet géométrique de base comme un cube, cylindre ou autre) réalisée par une ou plusieurs actions interactives de base que l'utilisateur effectue en utilisant une ou plusieurs modalités (clic avec la souris, commande vocale, etc…). Les tâches utilisateurs plus complexes sont réalisées par la composition séquentielle ou parallèle de ces énoncés. Par exemple, la tâche de conception d'un modèle d'objet géométrique complexe composé de plusieurs modèles de base tels que cylindre, cube, etc… est réalisée à l'aide des énoncés correspondant aux placements de ces objets de base. Cet espace de conception classe les IHM3 en sept types.

- *Exclusif :* une seule modalité est utilisée par énoncé et la production des énoncés est séquentielle.
- *Alterné :* plusieurs modalités peuvent être utilisées alternativement pour produire un énoncé. Les énoncés sont produits de manière séquentielle.
- *Synergique :* les énoncés sont produits de manière séquentielle mais plusieurs modalités peuvent être utilisées dans un même énoncé et de manière parallèle.
- *Parallèle exclusif :* plusieurs énoncés indépendants peuvent être produits en parallèle. Une seule modalité est utilisée pour chaque énoncé et à un instant donné une seule modalité est active.
- *Parallèle simultané :* plusieurs énoncés indépendants peuvent être produits en parallèle. Une seule modalité est utilisée pour chaque énoncé, mais à un instant donné plusieurs modalités peuvent être actives en même temps.
- *Parallèle alterné :* plusieurs énoncés indépendants peuvent être produits en parallèle. Plusieurs modalités peuvent être utilisées dans un même énoncé, mais une seule est active à un instant donné.
- *Parallèle synergique :* plusieurs énoncés indépendants peuvent être produits en même temps. Plusieurs modalités sont utilisables pour un même énoncé et plusieurs modalités peuvent être actives en même temps.

Nous utilisons l'espace de conception décrit ci-dessus pour paramétrer notre modèle formel proposé en section 4. Par ailleurs, dans le contexte de ce modèle, et à des fins de modélisation générique, nous définissons la modalité comme un type d'événement de base. Cette définition permet d'appréhender une modalité comme une abstraction de toutes les définitions de la notion de modalité proposées par la communauté des chercheurs dans le domaine des IHM3.

2.3. Propriétés des IHM3

Un ensemble de propriétés permettant de caractériser et de vérifier l'utilisabilité d'une IHM3 a été défini dans (Coutaz *et al.*, 1995). Ce sont les propriétés CARE : **C**omplémentarité, **A**ssignation, **R**edondance et **E**quivalence. Ces notions ont été introduites initialement dans l'espace de conception TYCOON (Martin *et al.*, 1994) qui fournit une étude sur l'usage de la multimodalité. Ces propriétés peuvent être appliquées aux dispositifs physiques, langages d'interaction ou aux tâches. Nous définissons chacune de ces propriétés pour une tâche *T* donnée et définie par un état initial et un état final. A titre d'illustration, nous considérons la tâche consistant à dessiner un cercle en précisant deux points à l'écran. Le premier point désigne le centre du cercle et le second point désigne un point périphérique.

– Les modalités d'un ensemble *M* sont dites *complémentaires* si elles doivent toutes être utilisées, pour réaliser la tâche *T* et aucune modalité de l'ensemble *M* ne suffit à elle seule pour réaliser la tâche *T*. Par exemple, si l'utilisateur précise le premier point du cercle avec un clic de la souris et le second point en prononçant la phrase 'périphérique x y ' où (x,y) sont les coordonnées du point périphérique, alors la voix et la manipulation directe avec la souris sont dites complémentaires pour la réalisation de la tâche de dessin du cercle.

– Les modalités d'un ensemble *M* sont dites *équivalentes* si chacune d'elle permet de réaliser la tâche *T*. Par exemple, la voix et la manipulation directe sont dites équivalentes si le système permet à l'utilisateur de dessiner le cercle avec la souris et avec la voix. Avec la souris l'utilisateur clique sur la commande cercle dans le menu par exemple, puis désigne deux points sur l'écran. Le premier point correspond au centre et le second correspond à un point périphérique. Avec la voix, l'utilisateur prononce la commande « cercle de centre x y périphérique t z » ou (x,y) désignent les coordonnées du centre du cercle et (t,z) les coordonnées d'un point périphérique.

– Les modalités d'un ensemble *M* sont dites *redondantes* si elles sont équivalentes et sont toutes utilisées, en parallèle ou en séquence, pour réaliser la tâche *T*. Si l'utilisateur active les deux commandes équivalentes, présentées précédemment, de manière parallèle, alors les deux modalités voix et manipulation directe sont dites redondantes pour la tâche de réalisation du cercle. L'intérêt de cette propriété apparaît surtout dans les interactions en sortie. Si on alerte l'utilisateur par un message affiché et sonore cela augmentera l'insistance de l'IHM3. Elle permet également d'offrir des dispositifs d'interaction alternatifs (handicap par exemple).

– Une modalité est dite *assignée* à la tâche *T* si elle est l'unique modalité permettant de réaliser cette tâche. Si le système ne permet de dessiner le cercle qu'avec la manipulation directe utilisant la souris, on dit alors que la manipulation directe est assignée à la tâche de dessin du cercle.

Ces propriétés, lorsqu'elles sont formalisées et évaluées, permettent de mesurer la qualité et l'ergonomie d'une IHM3. Elles permettent également d'exprimer les

exigences d'interaction, en particulier celles qui proviennent des concepteurs (utilisateurs de systèmes) en CAO.

2.4. Besoins en CAO

Une interface de CAO est généralement caractérisée par un nombre important d'objets manipulés (modèles géométriques, modèles de pièces, modèles de moules, etc.) et par un nombre important de commandes, structurées en menus, permettant d'accéder aux fonctionnalités du système. Assurer l'utilisabilité de ce type d'interface est une composante importante du processus de développement de tels systèmes. Elle permet d'évaluer la capacité d'un utilisateur à apprivoiser l'usage du système. Un ensemble de propriétés d'utilisabilité a été établi par la communauté des chercheurs dans le domaine des IHM. Ces propriétés sont classées principalement en deux catégories de propriétés : propriétés de robustesse et propriétés de validité.

L'objectif de cet article est de proposer une approche permettant de vérifier les propriétés de flexibilité de l'interaction exprimant la multitude de possibilités à la disposition de l'utilisateur pour interagir avec le système. Ces propriétés sont exprimées par l'équivalence et la complémentarité.

Une IHM3 où l'utilisateur a le choix entre plusieurs modalités pour réaliser une tâche est plus souple et plus flexible qu'une IHM ne lui offrant qu'une seule modalité impliquant une séquence d'interactions et n'autorisant pas le parallélisme. Cette propriété est exprimée par l'équivalence de modalités. La propriété de complémentarité permet quant à elle de rentabiliser au mieux l'utilisation des modalités. Le clic de la souris, plus précis pour indiquer un endroit sur l'écran, peut être combiné de manière séquentielle avec la parole pour lancer une commande. Ce sont ces propriétés que nous nous proposons d'exprimer de manière formelle puis de vérifier à l'aide de l'outil SMV (Symbolic Model Verifier) (McMillan, 1992) dans la section 6.

3. Utilisation des méthodes formelles pour les IHM3

Les méthodes formelles, par l'utilisation de modèles et de techniques reposant sur des définitions mathématiques rigoureuses, offrent de nouvelles possibilités pour la conception et la validation des logiciels. Leur mise en application permet de garantir des propriétés fondamentales des systèmes, avec pour résultats concrets d'éliminer beaucoup d'erreurs dès la conception et de prendre en compte les besoins des utilisateurs en amont du processus de développement.

3.1. Travaux antérieurs

Peu de travaux mettant en œuvre des techniques formelles se sont intéressés à la modélisation, à la vérification et la validation d'IHM3. Trois approches ont retenu notre attention. En premier, les travaux (Paterno *et al.*, 1994) utilisent les interacteurs pour modéliser l'interface de *Matis* (Nigay, 1994) à partir d'un modèle de tâches défini avec UAN (Hartson *et al*., 1992). Les interacteurs ont été implémentés en Lotos (Bolognesi *et al*., 1989). Un ensemble de propriétés a été vérifié à l'aide de la logique temporelle ACTL avec le contrôleur sur modèle de l'outil Lite (Van Eijk, 1991). Dans cette approche les auteurs ne modélisent pas l'IHM3 mais uniquement les tâches utilisateurs exprimées dans la notation UAN. Ensuite, (Duke *et al.*, 1997) et (MacColl *et al.*, 1998) montrent comment des techniques telles que Z (Spivey, 1988) et CSP (Hoare, 1978) peuvent être utilisées pour modéliser une application multimodale. Dans ce travail, les auteurs font abstraction des modalités, et évitent la prise en charge de toutes les spécificités d'une IHM3. Les problèmes de fusion et de vérification des propriétés CARE ne sont pas traités. Enfin, (Palanque *et al.*, 2003) utilisent le modèle ICO, fondé sur les réseaux de Pétri, pour représenter la fusion de deux modalités : le geste et la parole. Ce travail ne traite pas la vérification de propriétés spécifiques aux IHM3.

3.2. Notre proposition

Par rapport aux approches précédentes, notre démarche est générale (Kamel, 2004b) (Ait Ameur *et al.,* 2004). Elle permet la modélisation des IHM3 selon le type d'IHM désiré et défini dans l'espace de conception présenté dans la section précédente. Le concepteur choisit le type d'IHM3 à concevoir, décrit les différentes interactions multimodales selon le modèle formel proposé, en composant les différents systèmes de transition associés aux interactions. Les propriétés de l'IHM3, désirées, peuvent être exprimées dans une logique temporelle pour être vérifiées à l'aide d'une technique formelle (Kamel, 2004a). Pour illustrer cette approche, nous présentons dans cet article le cas du contrôleur sur modèle SMV (McMillan, 1992), mais toute autre technique formelle supportant la représentation de systèmes de transitions aurait pu être retenue.

4. Modèle paramétré d'interactions multimodales

Nous présentons dans cette section le modèle formel paramétré (pour être générique) que nous avons défini pour la modélisation de l'interaction utilisateur en entrée avec un système interactif multimodal. Ce modèle est inspiré des algèbres de processus CCS et Lotos. La syntaxe du modèle est donnée par des règles de grammaire et la sémantique des mots de cette grammaire est donnée en termes de

systèmes de transitions étiquetées. Ce modèle permet de concevoir une IHM selon les classes d'IHM3 définies dans la section 2. Avant de présenter la syntaxe du modèle nous introduisons les éléments nécessaires pour sa définition.

4.1. Eléments syntaxiques

Nous classons les actions de base selon les modalités qui les produisent. La notion de modalité dans notre modèle définit le type des actions. Afin de paramétrer notre modèle nous avons structuré l'ensemble des actions comme suit :

- A_{mi} dénote l'ensemble des actions *ei* produites par la modalité *mi*. Si nous disposons de *n* modalités,
- $A = \bigcup_{i=1}^{n} A_{mi}$ et on note *e* les éléments de *A*.
- *AM* désigne l'ensemble des actions générées par un sous-ensemble de modalités du système. Si le système dispose de *n* modalités alors

$$AM = \bigcup_{k \in 1..n} A_{mk} \text{ avec } A_{mk} \subseteq A.$$

En plus des ensembles des actions de base, un ensemble d'opérateurs de composition est défini. Leur sémantique informelle est donnée comme suit : (*P* et *Q* sont deux termes de la grammaire et *e* un élément de l'ensemble *A*)

- δ est un terme qui désigne l'action nulle (qui ne fait rien. Arrêt) ;
- **[]** est le choix non déterministe entre deux tâches. L'exécution de *P*[]*Q* est soit celle de *P*, soit celle de *Q* ;
- **>>** est l'opérateur de séquence entre deux tâches. L'exécution de *P*>>*Q* est celle de P suivie de celle de *Q* ;
- **;** est l'opérateur de préfixage. L'exécution de *e* ;*P* est définie par l'exécution de l'action *e* suivie de l'exécution de *P* ;
- **|||** désigne l'opérateur de l'entrelacement. *P*|||*Q* est la composition parallèle entrelacée des deux exécutions de *P* et *Q* ;
- **||** est l'opérateur parallèle. *P*||*Q* est la composition parallèle entre les exécutions de *P* et *Q*. Leurs actions peuvent s'exécuter de manière parallèle ou entrelacée.

4.2. Syntaxe

La syntaxe des différents modèles est donnée par des grammaires avec deux règles pour chacune. La règle S est utilisée pour générer le modèle en fonction des tâches utilisateurs à un niveau élevé d'abstraction jusqu'au niveau des tâches

élémentaires qui sont les énoncés. Les énoncés sont les tâches qui déclenchent une fonction élémentaire du noyau fonctionnel de l'application. La règle E génère les énoncés en composant des actions de base *e*. Les énoncés sont paramétrés selon le modèle, par l'ensemble des modalités qui les génèrent. L'ensemble des opérateurs est restreint pour chaque modèle selon les caractéristiques du type de l'interface qu'il modélise.

Type exclusif : dans ce type d'IHM3 seuls les opérateurs de séquence (>>) et de choix ([]) sont utilisés pour composer les énoncés. Chaque énoncé est produit par une seule modalité (E_{Ami}) en séquence avec l'opérateur (;).

$$S ::= S\ []S \mid S >> S \mid E_{Ami}$$

$$E_{Ami} ::= ei\ ; E_{Ami} \mid \delta \text{ avec } ei \in A_{mi}$$

Type alterné : la seule différence entre ce type et le type exclusif est que chaque énoncé peut être produit par plusieurs modalités (E_{AM}).

$$S ::= S[]S \mid S >> S \mid E_{AM}$$

$$E_{AM} ::= e\ ; E_{AM} \mid \delta \text{ avec } e \in AM$$

Type synergique : dans ce type d'IHM3, les énoncés peuvent être produits par plusieurs modalités (E_{AM}) en séquence (;) ou en parallèle (|||, ||). Les énoncés sont composés uniquement en séquence (>>, []) pour formuler les tâches.

$$S ::= S\ []S \mid S >> S \mid E_{AM}$$

$$E_{AM} ::= e\ ; E_{AM} \mid e \mid\mid\mid E_{AM} \mid e \mid\mid E_{AM} \mid \delta \text{ avec } e \in AM$$

Type parallèle exclusif : les énoncés de ce type d'IHM3, sont produits par une seule modalité (E_{Ami}) en séquence (seul l'opérateur (;) est utilisé dans la règle E). Pour formuler des tâches, ils peuvent être composés en séquence (>>, []) ou en parallèle (|||) entrelacé, car une seule modalité est active à un instant donné.

$$S ::= S[]S \mid S >> S \mid S \mid\mid\mid S \mid E_{Ami}$$

$$E_{Ami} ::= ei\ ; E_{Ami} \mid \delta \text{ avec } ei \in Ami$$

Type parallèle simultané : dans ce type d'IHM, les énoncés peuvent être composés en parallèle mais chaque énoncé est produit par une seule modalité (E_{Ami}).

$$S ::= S[]S \mid S >> S \mid S \mid\mid S \mid E_{Ami}$$

$$E_{Ami} ::= ei\ ; E_{Ami} \mid \delta \text{ avec } ei \in A_{mi}$$

Type parallèle alterné : les énoncés dans ce type d'IHM peuvent être produits par plusieurs modalités (E_{AM}). Pour formuler des tâches, ils peuvent être composés en

séquence (>>, []) ou en parallèle entrelacé (|||), car une seule modalité est active à un instant donné.

$$S ::= S\ []S \mid S>>S \mid S|||S \mid E_{AM}$$

$$E_{AM} ::= e\ ;\ E_{AM} \mid e\ |||\ E_{AM} \mid \delta \text{ avec } e \in AM$$

Type parallèle synergique : ce type d'IHM représente l'inclusion de tous les modèles précédents. Il permet la composition en séquence (>>, []) ou en parallèle (|||, ||) des énoncés. Chaque énoncé peut être produit par une ou plusieurs modalités en séquence (;) ou en parallèle (|||, ||).

$$S ::= S[]S \mid S>>S \mid S||S \mid S|||S \mid E_{AM}$$

$$E_{AM} ::= e\ ;\ E_{AM} \mid e\ |||\ E_{AM} \mid e\ ||\ E_{AM} \mid \delta \text{ avec } e \in AM$$

4.3. Sémantique opérationnelle des opérateurs

La sémantique formelle des différents opérateurs de composition est donnée par des règles de transition définissant le système de transitions sous-jacent. Soient *P*, *P'*, *Q* et *Q'* des termes de la grammaire caractérisant chacun un état, *e, e_1* et *e_2* des actions de *A*. La transition $P \xrightarrow{e} Q$ décrit le passage d'un état caractérisé par *P* à un état caractérisé par *Q*, lorsque l'action *e* est exécutée. Les règles de transitions sont décrites selon le style de G.Plotkin (Plotkin, 1981) au moyen de règles de la forme $\frac{prémisses}{conclusion}$.

Etat d'arrêt : δ n'effectue aucune transition $\delta \not\longrightarrow$

Opérateur de préfixage : *e ;P* exécute *e* ensuite se comporte comme *P'*

$$e\ ;P \xrightarrow{e} P'$$

Opérateur de choix : si *P* exécute *e* ensuite se comporte comme *P'* Alors *P*[]*Q* exécute *e* ensuite se comporte comme *P'* (idem pour la seconde règle).

$$\frac{P \xrightarrow{e} P'}{P[]Q \xrightarrow{e} P'} \qquad \frac{Q \xrightarrow{e} Q'}{P[]Q \xrightarrow{e} Q'}$$

Opérateur de séquence : si *P* exécute *e* ensuite se comporte comme *P'* alors, dans le cas où *P'* n'est pas la fin, il se comportera comme *P>>Q* (première règle), sinon il se comportera comme *Q* (seconde règle).

$$\frac{P \xrightarrow{e} P' \ et\ P' \neq \delta}{P >> Q \xrightarrow{e} P' >> Q} \qquad \frac{P \xrightarrow{e} P' \ et\ P' = \delta}{P >> Q \xrightarrow{e} Q}$$

Opérateur d'entrelacement : si *P* exécute *e* ensuite se comporte comme *P'* alors, *P|||Q* exécute *e* et se comporte comme *P'|||Q* (idem pour la seconde règle).

$$\frac{P \xrightarrow{e} P'}{P \,|||\, Q \xrightarrow{e} P' \,|||\, Q} \qquad \frac{Q \xrightarrow{e} Q'}{P \,|||\, Q \xrightarrow{e} P \,|||\, Q'}$$

Opérateur de parallélisme : les deux premières règles sont identiques à celles de l'entrelacement. La troisième règle exprime le fait que si P exécute e_1 et se comporte comme P' et Q exécute e_2 et se comporte comme Q' alors $P||Q$ exécute les deux actions e_1 et e_2 et se comporte comme $P'||Q'$. Les modalités associées aux événements e_1 et e_2 doivent être différentes car on suppose qu'une modalité ne peut produire plus d'une action en même temps.

$$\frac{P \xrightarrow{e} P'}{P \,||\, Q \xrightarrow{e} P' \,||\, Q} \qquad \frac{Q \xrightarrow{e} Q'}{P \,||\, Q \xrightarrow{e} P \,||\, Q'}$$

$$\frac{P \xrightarrow{e_1} P' \; et \; Q \xrightarrow{e_2} Q' \text{ avec } \text{modalité}(e_1) \neq \text{modalité}(e_2)}{P \,||\, Q \xrightarrow{(e_1, e_2)} P' \,||\, Q'}$$

La relation de transition est obtenue par induction structurelle à partir des règles précédentes.

5. Etude de cas

Pour illustrer notre approche, nous avons choisi de modéliser les interactions d'une application de CAO simple. Un utilisateur choisit des formes géométriques 3D simples (cube, cylindre, etc…) pour les assembler pour concevoir des formes plus complexes. Plusieurs possibilités sont offertes à l'utilisateur pour formuler ses requêtes. Il utilise pour cela, la souris, la parole ou les deux. L'interface contient un ensemble d'icônes que l'utilisateur peut sélectionner pour désigner le type d'objet de base à manipuler (cylindre, cube, etc…). Des boutons sont également disponibles, des boutons sur lesquels l'utilisateur peut cliquer pour déclencher une commande telle que ajouter un objet, le supprimer ou le déplacer. Nous avons volontairement choisi une étude de cas simplifiée pour pouvoir montrer, dans cet article, l'ensemble des modèles formels développés ainsi que les moyens de les valider. Enfin, notons que nous ne nous intéressons pas aux traitements de la parole, mais nous supposons que le système est en mesure d'identifier les interactions vocales.

5.1. Actions de base

Nous avons choisi de modéliser notre IHM3 selon deux types. Un type *exclusif* où chaque commande doit être réalisée avec une seule modalité, et un type parallèle alterné où chaque commande peut être réalisée avec plusieurs modalités en parallèle entrelacé. La tâche que nous considérons consiste à concevoir un objet simple composé d'un cube au dessus duquel est placé un cylindre.

Avant de présenter les deux modélisations, nous présentons les ensembles d'actions de base du modèle :

A_{Souris} = {ClicCube, ClicCylindre, ClicAjou, ClicSup, Clic(x,y)}

A_{parole} = {'Supprimer', 'Ajouter', 'Cube', 'Cylindre', 'ça', 'ici', 'position x y'}

x, y sont des réels obtenus par le clic de la souris dans A_{souris} ou bien reconnus par la voix pour A_{parole}.

Les actions générées par la modalité souris sont :

– *ClicCube, ClicCylindre* : clic de la souris respectivement sur les objets cube et cylindre présents sur l'interface ;

– *ClicAjou, ClicSup* : clic de la souris respectivement sur les boutons *Add* et *Del* ;

– *Clic(x,y)* : clic de la souris sur le point de coordonnées (x,y) dans l'espace de conception de l'IHM3. En réalité, c'est un ensemble d'événements de même nature. A chaque couple (x,y) correspond une action. Pour notre étude de cas, (x_1,y_1) et (x_2,y_2) sont les coordonnées où doivent être placés le cube et le cylindre. Nous considérons qu'une conversion des points 2D en points 3D est disponible. Nous n'entrons pas dans les détails de cette conversion qui n'est pas essentielle à la compréhension de cette étude de cas.

Les actions de l'ensemble A_{parole} sont des mots prononcés par l'utilisateur et reconnus par le système.

Pour chacun des deux modèles choisis pour modéliser notre IHM, nous donnons

– les expressions des énoncés en fonction des actions de base qui les composent. Ensuite,

– l'expression de la tâche qui conçoit l'objet en fonction des énoncés qui la composent, et

– enfin, l'expression de la même tâche en fonction des événements de base qui la composent en remplaçant chacun des énoncés par son expression.

5.2. Modélisation d'une interface de type exclusif

La modélisation d'une IHM3 de type exclusif où chaque énoncé est produit par une seule modalité est d'abord présentée. Les énoncés sont composés en séquence avec l'opérateur du choix ou de la séquence pour formuler des tâches.

Les énoncés : nous définissons quatre énoncés E_1, E_2, E_3 et E_4 composé chacun d'un ensemble d'actions de base des ensembles A_{parole} et A_{souris}. Pour réduire l'espace pris

par l'ecriture des expressions, nous avons renommé les actions de base pour chaque énoncé *Ei* en *eij*.

E_1 = clicAjou ; clicCube ; clic(x_1,y_1) = e_{11}; e_{12}; e_{13}
E_2 = clicAjou ; cliccylindre ; clic(x_2,y_2) = e_{21}; e_{22}; e_{23}
E_3 = 'Ajouter' ; 'Cube' ; 'position x_1 y_1' = e_{31}; e_{32}; e_{33}
E_4 = 'Ajouter' ; 'Cylindre' ; 'position x_2 y_2' = e_{41}; e_{42}; e_{43}

Les tâches : nous définissons deux tâches à réaliser « *placer le cube et placer le cylindre* ». La tâche *PlacerCube* consiste à placer un cube aux coordonnées (x_1,y_1). Elle est réalisée soit avec l'énoncé E_1 soit avec l'énoncé E_3. Elle est le résultat de la composition de ces deux énoncés par l'opérateur du choix ([]). Le même raisonnement est appliqué pour la tâche *PlacerCylindre*. Les expressions des deux tâches en fonctions des énoncés sont données par

PlacerCube = E_1 **[]** E_3
PlacerCylindre = E_2 **[]** E_4

La tâche *Concevoir* est la tâche globale. Elle consiste à composer l'objet complexe sur l'espace de conception. Elle est réalisée par les deux tâches *Placercube* et *PlacerCylindre*. L'utilisateur peut placer le cube ensuite le cylindre ou bien, le cylindre et ensuite le cube. Ceci se traduit par l'expression suivante

*(PlacerCube >> PlacerCylindre)***[]** *(PlacerCylindre >> PlacerCube).*

En fonction des énoncés, cette expression est dépliée en :

(E_1**[]**E_3)>>(E_2**[]**E_4))**[]**((E_2**[]**E_4)>>(E_1**[]**E_3))

et en fonction des actions de base, elle correspond à l'expression

((e_{11}; e_{12}; e_{13}))**[]**(e_{31}; e_{32}; e_{33})) **>>** ((e_{21}; e_{22}; e_{23})**[]**(e_{41}; e_{42}; e_{43})))
[]
((e_{21}; e_{22}; e_{23})**[]**(e_{41}; e_{42}; e_{43})) **>>** ((e_{11}; e_{12}; e_{13}) **[]** (e_{31}; e_{32}; e_{33})))

Le système de transitions correspondant à cette expression est obtenu par la composition des systèmes de transitions des énoncés qui le composent. Le système résultant est décrit sur la figure 2. Nous utilisons des flèches pleines pour les transitions réalisées avec les clics de la souris et des flèches en pointillés pour celles réalisées par la parole.

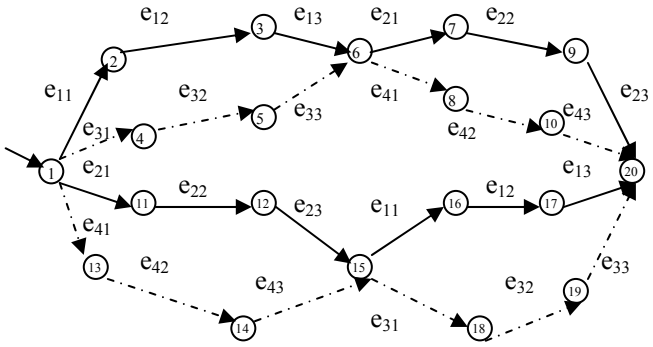


Figure 2. *Système de transitions de l'IHM de type exclusif*

5.3. Modélisation d'une interface de type parallèle alterné

Nous donnons ici, la modélisation d'une IHM3 de type parallèle alterné où chaque énoncé est produit par les deux modalités en séquence ou en parallèle entrelacé. Nous n'avons pas utilisé l'opérateur parallèle au niveau des tâches.

Les énoncés : les deux modalités participent à la réalisation des énoncés E_1, E_2, E_3 et E_4. Les énoncés E_3 et E_4 sont réalisés en combinant la parole et les clics souris en parallèle entrelacé. Dans cet exemple et contrairement au précédent, nous utilisons le paradigme de Bolt décrit par la phrase (prononcée oralement) « met ça ici » (Bolt, 1980). Par exemple, l'utilisateur prononce la phrase 'Ajouter ça ici' pendant qu'il clique sur l'objet à ajouter et son emplacement sur l'écran. Les coordonnées récupérées par les clics souris sont fusionnées avec la commande 'Ajouter ça ici'. Notons que les clics de la souris peuvent précéder la phrase ou inversement ou bien intervenir pendant qu'elle est prononcée.

E_1 = 'Ajouter' ; 'Cube' ; Clic(x_1,y_1) = e_{11}; e_{12}; e_{13}
E_2 = Ajouter' ; 'Cylindre' ; Clic(x_2,y_2) = e_{21}; e_{22}; e_{23}
E_3 = ('Ajouter' ;'ça' ;'ici')|||(ClicCube;Clic(x_1,y_1)
= (e_{31}; e_{32}; e_{33}) ||| (e_{34}; e_{35})
E_4 =('Ajouter';'ça';'ici')|||(ClicCylindre;Clic(x_2,y_2)
= (e_{41}; e_{42}; e_{43}) ||| (e_{44}; e_{45})

Les tâches : nous avons gardé les mêmes tâches que celles utilisées dans l'IHM3 de type exclusif en section 5.2. Les expressions des tâches *PlacerCube* et *PlacerCylindre* restent identiques.

$PlacerCube = (E_1 [] E_3)$
$PlacerCylindre = (E_2 [] E_4)$
$Concevoir = (placerCylindre >> placerCube) [] (placerCube >> placerCylindre)$
$= ((E_1 [] E_3) >> (E_2 [] E_4)) [] ((E_2 [] E_4) >> (E_1 [] E_3))$

L'expression de la tâche C*oncevoir* est donnée en fonction des actions de base par

$$(((e_{11};e_{12};e_{13}) [] ((e_{31};e_{32};e_{33}) ||| (e_{34};e_{35})) >> ((e_{21};e_{22};e_{23}) [] ((e_{41};e_{42};e_{43}) ||| (e_{44};e_{45})))) [] (((e_{21};e_{22};e_{23})) [] ((e_{41};e_{42};e_{43}) ||| (e_{44};e_{45}))) >> ((e_{11};e_{12};e_{13}) [] ((e_{31};e_{32};e_{33}) ||| (e_{34};e_{35}))))$$

Par faute d'espace, nous ne présentons pas le système de transitions correspondant. Comme pour la figure 2, il est obtenu par composition des systèmes de base.

6. Vérification formelle par model-checking : utilisation de SMV

La vérification sur modèle, connue par model-checking, est une technique basée sur la construction d'un modèle fini du système et la vérification des propriétés sur ce modèle. La vérification est effectuée par énumération exhaustive de l'espace d'états. L'approche que nous utilisons dans cet article a été développée par Clarke et Emerson (Clarke *et al.,* 1981) et Quille et Sifakis (Queillie *et al.*, 1981). Elle consiste à exprimer les propriétés dans une logique temporelle (Manna *et al.*, 1992) et à modéliser le système par un système de transitions fini. Un algorithme est utilisé pour vérifier si le système de transitions est un modèle pour la spécification. L'outil que nous utilisons pour la mise en œuvre de notre modèle est SMV. Il implémente cette approche de model-checking. Le langage d'entrée de SMV permet de décrire le modèle du système (système de transitions) et sa spécification. La spécification est décrite par des formules de logique temporelle CTL que SMV valide sur le modèle du système. Un générateur de code C, représentant le système de transitions décrit, permet d'obtenir le code associé implémentant le système. L'intérêt de SMV est de reposer sur une représentation symbolique à base de BDD (Binary Decision Diagram), des états du système. Ces diagrammes permettent de représenter le graphe des états du modèle de manière implicite ainsi que les ensembles des états qui satisfont une propriété donnée. Cette représentation permet une manipulation efficace à l'aide d'opérations ensemblistes. Il a été ainsi un des premiers outils en mesure de vérifier de façon exhaustive la correction de systèmes de taille très élevée.

Bien que SMV ait montré sa performance pour des systèmes de grande taille, une mise en œuvre d'un système réel nécessite d'autres techniques formelles telles que le raffinement, la décomposition et l'abstraction. Ces techniques permettent de réduire la complexité du processus de vérification. Nous n'abordons pas ces aspects dans cet article.

6.1. Formalisation du système multimodal

Pour vérifier les propriétés CARE, il faut raisonner sur les modalités des actions effectuées. Pour cela, il est utile de remplacer les étiquettes des actions, dans les systèmes de transitions, par leurs modalités respectives. Le système de transitions obtenu est une abstraction correcte du système d'origine. Il est codé dans le langage d'entrée du contrôleur sur modèle SMV dans les clauses ***VAR*** *et* ***ASSIGN***.

```
MODULE main
VAR
  etat : {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};
  modalite : {parole, souris};
ASSIGN
  init(etat) := 1;                              -- Initialisation des variables
  init(modalite) :={parole, souris};
  next(etat) :=                                 -- définition de l'automate de la variable etat
    case
              etat=1 : {2,4,11,13};
              etat=2 : 3; etat=3 : 6; etat=4 : 5; etat=5 : 6;
              etat=6 : {7,8};                    -- non déterminisme
              etat=7 : 9; etat=8 : 10; etat=9 : 20; etat=10 : 20; etat=11 : 12;
              etat=12 : 15; etat=13 : 14; etat=14 : 15;
              etat=15 : {16,18};
              etat=16 : 17; etat=17 : 20; etat=18 : 19; etat=19 : 20;
              1 : etat;
   esac;
  next(modalite):=                               - définition de l'automate de la variable modalite
    case
              etat=1 | etat=3 | etat=9 | etat=12 | etat=17 | etat=5 | etat=10 | etat=14 |
              etat=19 | etat=6 | etat=15          :{souris, parole};
              etat=2 | etat=7 | etat=11 | etat=16 :souris;
              etat=4 | etat=8 | etat=13 | etat=18 :parole ;
              1 : modalite;
    esac;
```

Figure 3. *Code SMV de l'IHM de type « exclusif »*

Deux variables d'état caractérisent le système. La variable *etat* représente l'état global du système. La variable m*odalite* représente la modalité utilisée pour réaliser les actions et transiter d'un état à un autre dans le système de transitions. Pour toute transition effectuée d'un état source vers un état cible avec une modalité *mi*, la

valeur de la variable *modalite* dans l'état source est égale à *mi*. L'opérateur **init** permet d'initialiser une variable et l'opérateur **next** appliqué à une variable définit sa valeur suivante dans le système. Nous donnons sur la figure 3 le code SMV correspondant au système de l'IHM3 de type *exclusif* représenté sur la figure 2. Pour l'instant, la génération du système de transitions ainsi que son codage dans le langage de SMV ont été effectués manuellement, mais un générateur de systèmes de transitions à partir de la syntaxe du modèle ainsi que sa traduction vers SMV est en cours de développement.

6.2. Expression des propriétés d'utilisabilité

Les propriétés vérifiées par le contrôleur sur modèle SMV sont exprimées dans la logique temporelle CTL dans la clause ***SPEC***. En plus des opérateurs de la logique du premier ordre, d'autres opérateurs temporels sont définis pour cette logique. Les opérateurs utilisés dans cet article : **EG***P*, **AG***P, Q***U***P* et **E***P*, signifient de manière informelle, respectivement, « il existe un chemin où tous les états vérifient *P*», « tous les états de tous les chemins vérifient *P*», « *Q* est vraie jusqu'à ce que *P* soit vraie » et « il existe un chemin où *P* est vraie ».

Nous avons choisi d'exprimer et de vérifier les deux propriétés d'utilisabilité qui sont l'*équivalence* et la *complémentarité*, sur les deux IHM3.

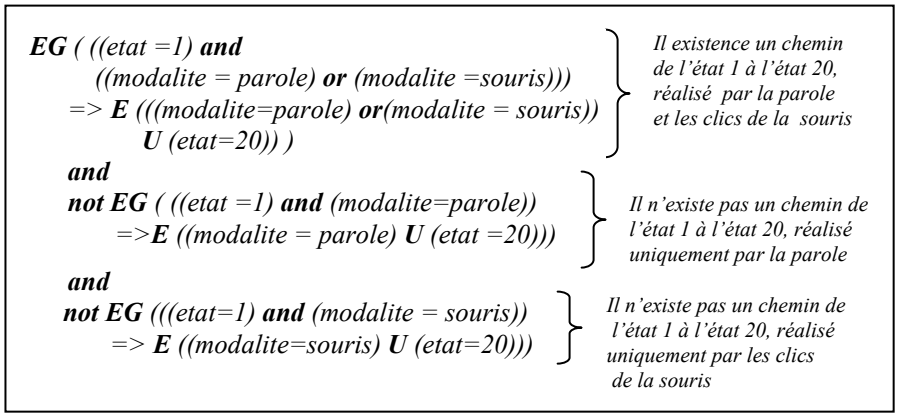
AG *(((etat=1)* ***and*** *(modalite=parole))*
⇒ ***E*** *((modalite=parole) ∪ (etat=20)))* } *Chemin de l'état 1 à l'état 20, réalisé par la parole*
And

AG *(((etat=1)* ***and*** *(modalite=souris))*
⇒ ***E****((modalite=souris) ∪ (etat=20)))* } *Chemin de l'état 1 à l'état 20, réalisé par les clics souris*

La formule ci-dessus exprime l'*équivalence* des deux modalités. Elle indique formellement que

- sur toutes les exécutions, si un état est identifié comme l'état initial *(etat=1)* de la tâche considérée, alors il existe un chemin allant de cet état tel que tous les états sont atteints par la modalité *parole* jusqu'à atteindre l'état final de la tâche *(etat=20)* et

- sur toutes les exécutions, si un état est identifié comme l'état initial *(etat=1)* de la tâche considérée, alors il existe un chemin allant de cet état tel que tous les états sont atteints par la modalité *souris* jusqu'à atteindre l'état final de la tâche *(etat=20).*



La formule ci-dessus exprime la *complémentarité* des deux modalités. Elle indique formellement que

- il existe une exécution telle que, si un de ses états est identifié comme l'état initial *(etat=1)* de la tâche considérée, alors il existe un chemin allant de cet état tel que, tous ses états sont atteints par la modalité *parole* ou *souris*, jusqu'à atteindre l'état final de la tâche *(etat=20)* ; et
- il n'existe pas d'exécution telle que, toutes les transitions, entre l'état initial et l'état final, de la tâche sont effectuées par la modalité parole ; et
- il n'existe pas d'exécution telle que toutes les transitions, entre l'état initial et l'état final de la tâche, sont toutes effectuées par la modalité souris.

Nous avons codé les deux systèmes ainsi que les propriétés associées en SMV. La propriété d'équivalence des deux modalités souris et parole est vérifiée par le système de l'IHM3 de type exclusif, mais ne l'est pas par le système de l'IHM3 de type parallèle alterné. La propriété de complémentarité est vérifiée pour l'IHM3 de type parallèle alterné, mais ne l'est pas par l'IHM3 de type exclusif.

L'intérêt de la propriété d'équivalence est que si une des deux modalités est absente pour une raison ou une autre, ou bien si l'utilisateur ne peut pas utiliser la souris pour une raison de handicap, l'IHM3 est toujours utilisable. Par ailleurs dans le type parallèle alterné, l'utilisateur profite des avantages des deux modalités en même temps en accélérant ainsi le processus de conception en utilisant la phrase *'Ajouter ça ici'* et les deux clics (*ClicCube, Clic(x_1,y_1)*) en même temps pour réaliser sa tâche. Cependant, l'inconvénient est que si une des deux modalités est absente, alors l'IHM3 n'est plus utilisable. L'utilisation du type parallèle dans un système de CAO implique une réduction des chemins d'interaction et une utilisation plus efficace et plus ergonomique de ces systèmes.

Les opérateurs que nous avons utilisés dans notre modèle formel sont basiques, mais d'autres opérateurs peuvent être modélisés tels que les opérateurs de synchronisation, d'interruption etc... Les formules exprimant les propriétés CARE resteront valables car elles se basent toutes sur l'**existence** d'un chemin vérifiant les

contraintes liées à l'utilisation de la modalité. L'interruption, par exemple, introduit de nouveaux chemins dans le système de transitions mais le chemin exprimant l'exécution complète de la tâche reste toujours présent dans le système de transitions.

7. Conclusion

Le travail présenté dans cet article est une contribution à la prise en compte de la multimodalité dans les IHM des systèmes de CAO. Nous avons développé un modèle formel permettant de concevoir un module de fusion de modalités dans les systèmes de CAO. De plus, ce même modèle permet, grâce à l'expression et à la vérification de propriétés, de mesurer la qualité des chemins d'interactions offerts par un système donné. Nous sommes convaincus que l'utilisation d'interactions parallèles permet d'augmenter l'efficacité de l'interaction et l'ergonomie du système pour laisser le concepteur (utilisateur du système de CAO) extrêmement concentré sur les objets et sur les modèles en cours de construction.

L'utilisation de méthodes formelles a permis de décrire de manière rigoureuse la totalité de l'interaction multimodale indépendamment de tout système particulier. En ce sens, notre approche est modulaire est n'est donc pas dédiée à un système particulier.

8. Bibliographie

(Ait Ameur *et al.,* 2004) Aït-Ameur Y., Kamel N., A generic formal specification of fusion of modalities in a multimodal HCI. *IFIP World Computer Congress*, edited by Jacquart, René, Toulouse, France, Kluwer Academic Publishers, 2004, pp. 415-420.

(Bellik, 1995) Bellik Y., Interfaces multimodales : concepts, modèles et architectures, thèse de doctorat d'université 1995.

(Bold, 1980) Bold R. A., Put-that-there: voice and gesture at the graphics interface. In *ACM SOGGRAPH Computer Graphics*, New York, 1980. ACM Press.

(Bolognesi *et al.,* 1989) Bolognesi T., Btinksma E., Introduction to the ISO specification language LOTOS. Van Eijk P, Vissers C & Diaz M (Eds.) *The Formal Description Technique LOTOS*, Elsevier (North-Holland), 1989, pp. 23-73.

(Clarke *et al.,* 1981) Clarke E.M., Emerson E. A., Synthesis of synchronization skeletons for branching time temporal logic. In *logics of programs: workshop,* volume 131, York-town Heights, New York, 1981. Springer-Verlag.

(Coutaz *et al.,* 1995) Coutaz J., Nigay L., Salber D., Blandford A., May J., Young R.M. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In K. Nordby, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (eds) *Human Computer Interaction: Interact '95*. Chapman and Hall: London. pp. 115-120, 1995.

(Duke *et al.,* 1997) Duke D., Harisson M. D., mapping User requirements to implementations. In *Software Engineering Journal* (Vol 10(1), p54-75) 1997.

(Gardan, 1991) Gardan Y., *La CFAO, introduction, techniques et mise en œuvre*. Paris : Hermes, 1991, 418p.

(Hartson *et al.,* 1992) Hartson H.R., Gray P.D., Temporal aspects of tasks in the user action notation. *Human_computer interaction*, 1992, vol.7, pp.1-45.

(Hoare, 1978) Hoare C.A.R., Communicating Sequential Processes. *CACM* 21(8) 1978.

(Igarashi *et al.,* 1999) Igarashi T., Matsuoka S., and Tanaka H., Teddy : A Sketching Interface for 3D Freeform Design. In Proceedings of *ACM SIGGRAPH'99*, pp. 409-416. 1999.

(Mitani *et al.,* 2002) Mitani J., Suzuki H., Kimura F., 3D Sketch: sketch-based model reconstruction and rendering. In *from geometric modeling to shape modeling*, pages 85-98, Kluwer Academic Publisher, 2002.

(Kamel, 2004a) Kamel N.(a), Utilisation de SMV pour la vérification de propriétés d'IHM multimodales *16° Conférence Francophone sur l'Interaction Homme-Machine (IHM'2004)*, vol. 1, Namur, Belgique, ACM Press, 2004, pp. 219-222.

(Kamel, 2004b) Kamel N.(b), Modélisation et vérification formelle des IHM multimodales (Rencontres Jeunes Chercheurs). *Rencontre Jeunes Chercheurs en Interaction Homme-Machine*, vol. 1, Lacanau, 2004.

(MacColl *et al.,* 1998) MacColl, Carrington D., testing MATIS : a case study on specification-based testing of interactive systems, *FAHCI* (1998), pp.57-69, ISBN 0-86339-7948

(Manna *et al.,* 1992) Manna Z., Pnueli A., *the temporal logic of reactive and concurrency systems : specification*. Springer-verlag, 1992.

(Martin, 1994) Martin J.C., Cadre d'étude de la multimodalité sur les types et buts de coopération entre modalités. Proc. *Interface to Real & Virtual World*, Ec2 Publ., 97-106, 1994.

(McMillan, 1992) McMillan K. L., The SMV system, Technical report, Carnegie Mellon University, 1992.

(Nigay, 1994) Nigay L., Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales, PhD dissertation, Grenoble University, 1994, 315 pages.

(Nigay *et al.,* 1995) Nigay L., Coutaz J., A Generic Platform for Addressing the Multimodal Challenge. Proc. Of *CHI'95*, 1995, 98-105.

(Nigay *et al.,* 1996) Nigay L., Coutaz J., Espaces conceptuels pour l'interaction multimédia et multimodale, *TSI, spéciale Multimédia et collecticiel*, AFCET & HERMES Publ, Vol 15(9), 1996, p. 1195-1225.

(Palanque *et al.,* 2003) Palanque P., Schyn A., A model-based approach for engineering multimodal interactive systems. Proceeding of *the ninth IFIP TC13 international Conference on Human-Computer Interaction (Interact'2003)*, sep 1-5, 2003. Zürich, Switzerland.

(Paterno *et al.,* 1994) Paterno F., Mazzanotte M., Analysing Matis by Interactors and ACTL. Amodeus Project Document: system Modelling/WP36, 1994.

(Plotkin, 1981) Plotkin G. D., A structural approach to operational semantics. Rapport de recherche DAIMI FN-19 Université d'Arhus Computer Science departement Arhus Danemarke 1981.

(Queille *et al.,* 1981) Queille J., Sifakis J., Specification and verification of concurrent systems in Caesar. *In 5th international symp. On programming,* Lecture Notes in Computer Science, pages 337-351. Springer Verlag, 1981.

(Spivey, 1988) Spivey J.M., *The Z notation : A reference Manual*. Prentice Hall Int., 1988.

(Van Eijk, 1991) Van Eijk P., The lotosphere integrated environment. Proceedings *4th international conference on formal description technique, (FORTE91)*, Sidney, november 1991, North Holland, pp. 473-476. 1991.