Vérification et validation formelles de systèmes interactifs fondées sur la preuve : application aux systèmes Multi-Modaux

Yamine Aït-Ameur, Idir Aït-Sadoune, Mickael Baron, Jean-Marc Mota

Vérification et validation formelles de systèmes interactifs fondées sur la preuve : application aux systèmes multi-modaux

YAMINE AIT-AMEUR, IDIR AIT-SADOUNE et MICKAEL BARON
(*) Laboratoire LISI-ENSMA, Poitiers
JEAN-MARC MOTA
(**) Thales Rail Signalling Solutions, Vélizy

Résumé : Cet article s'intéresse à la validation et à la vérification formelles d'IHM Multi-Modales (IHM3). Il décrit une partie des résultats obtenus dans le cadre du projet RNRT VERBATIM, dont l'objet est la VERification Biformelle et Automatisation du Test des Interfaces Multimodales. Ce projet s'intéresse, entre autres, à la mise en œuvre d'une technique formelle : la méthode B événementiel. Cette technique fondée sur la preuve ne souffre donc pas du problème d'explosion du nombre d'états rencontré généralement par les techniques de vérification sur modèles. Nous discutons les apports de cette technique pour la conception d'IHM3, en particulier, sa capacité à exprimer et à vérifier des propriétés de la famille CARE. Notre approche utilise des notations et techniques semi-formelles issues du domaine des IHM et propose de les formaliser. Enfin, nous appliquons notre approche sur une étude de cas appelé "Pages Jaunes CLIPS".

Mots clés : IHM multi-modales, technique formelle fondée sur la preuve, vérification, validation, propriétés CARE, modèle de tâches.

Abstract: This paper focuses on the formal validation and verification of multi-modal human computer interfaces. It describes part of the obtained results of the French RNRT VERBATIM project whose purpose is the Multimodal Interfaces BIformal Verification and Test Automation. This project focuses on the application of a formal technique, namely the event B method. This approach is based on a proof technique and therefore it does not suffer from the state number explosion problem occurring in classical model checking. We outline the capability of this technique to support the design of multi-modal human computer interfaces, in particular, the capability to support the expression and the verification of properties issued from the CARE family. The proposed approach uses notations and semi-formal techniques issued from the HCI design area. We apply our approach on a case study called "CLIPS Yellow Pages".

Key words: Multi-modal HCI, proof based technique, verification, validation, CARE properties, task model.

(*) Laboratoire LISI (EA 1232), ENSMA, Téléport 2, 1 avenue Clément Ader, BP 40109, 86961 Futuroscope Cedex.
(**) Thales Rail Signalling Solutions, 20-22 rue grange Dame rose 78140 Vélizy.

1. INTRODUCTION

La diversité des interfaces homme-machine (IHM) ainsi que les progrès réalisés dans la définition de nouveaux dispositifs d'interaction ont conduit à une complexité dans la conception et la mise en œuvre des IHM. Le recours à des modèles de conception et à des notations de description des IHM devient indispensable pour maîtriser cette complexité. Dans le but d'améliorer la flexibilité et l'utilisabilité des IHM, de nombreux travaux ont proposé des techniques, des notations et des méthodes pour les différentes étapes de développement d'une IHM. Toutefois leur application aux IHM multi-modales (IHM3) reste toujours un sujet de préoccupation pour les chercheurs.

L'étude de l'utilisabilité des IHM multi-modales implique la prise en considération de nouvelles notions absentes dans les IHM classiques. Parmi les propriétés propres aux IHM multi-modales nous trouvons les propriétés CARE (Complémentarité, Assignation, Redondance et Équivalence [Coutaz et al. 1995]) que le système doit assurer [Nigay et al. 1995].

Par ailleurs, il est bien établi que les méthodes formelles participent à l'augmentation de la qualité des développements de systèmes interactifs et d'interfaces homme-machine [Hix and Hartson 1993]. De nombreux travaux mettant en œuvre ce type de technique ont été menés dans ce domaine. Ils se fondent soit sur les techniques formelles orientées modèles (model checking) ou bien sur la preuve (theorem proving).

Nous proposons d'utiliser une technique formelle fondée sur la preuve (la méthode B événementiel [Abrial 2010]) afin de prendre en compte les propriétés CARE dès les premières étapes de conception, à un niveau de développement qui fait abstraction aussi bien des détails d'implantation que des éléments du noyau fonctionnel. Plus précisément, nous nous intéressons, dans cet article, à la mise en œuvre des techniques formelles fondées sur la preuve pour la conception, la validation et la vérification formelles de propriétés d'utilisabilité, en l'occurrence les propriétés de la famille CARE, des IHM3. Cet article étend [Aït-Ameur et al. 2006] et décrit une partie des résultats obtenus dans le cadre du projet RNRT VERBATIM[1] s'intéressant à la validation formelle des propriétés liées à la multimodalité en télécommunications mobiles.

Cet article est structuré de la façon suivante. La section qui suit décrit les modèles et notations utilisées dans le processus de spécification et de conception de systèmes interactifs. Nous détaillons également les propriétés qui doivent être établies et vérifiées pour les systèmes interactifs classiques et multi-modaux. Une deuxième section effectue un état de l'art sur l'utilisation des techniques formelles dans le domaine de l'interaction homme-machine. Nous présentons rapidement nos travaux antérieurs qui ont consisté à l'utilisation de la méthode B dans sa version classique et justifions le passage à la version B événementiel que nous décrivons dans la section suivante. Puis nous présentons successivement notre approche pour la modélisation de tâches utilisateurs et de conception de l'architecture logicielle nous permettant ainsi de modéliser le contrôleur de dialogue. Dans une dernière section, nous présentons l'étude de cas des pages jaunes développée au CLIPS-IMAG et mise en œuvre pour illustrer notre approche. Cette section montre comment les propriétés de la famille CARE sont validées sur une conception B événementiel. Les interactions multi-modales sont introduites progressivement grâce au raffinement en

[1] Réseau National de Recherche en Télécommunications. VERification Biformelle et Automatisation du Test des Interfaces homme machine Multimodales.
Site du projet http://iihm.imag.fr/nigay/VERBATIM/

codant un modèle de tâches utilisateurs. Enfin, nous terminons par un bilan de ce travail ainsi que quelques perspectives.

2. CONCEPTION DES INTERFACES HOMME-MACHINE

De manière générale, deux scénarii de développement différents aboutissant à des processus de développement différents peuvent être définis. Soit (1) étant donné un ensemble de propriétés, la spécification et la conception sont établies conformément à ces propriétés, soit (2) étant donnée une spécification, les propriétés doivent être vérifiées en adéquation avec la spécification établie. Dans les deux cas, la validation de propriétés est une activité fondamentale. Les propriétés doivent être validées par n'importe quelle spécification de la modélisation traduisant les exigences du cahier des charges.

2.1 Spécification / Conception

Comme pour les développements de programmes classiques, le domaine de l'IHM a vu un grand nombre de notations et de techniques pour la conception et la description. Par ailleurs, la finalité des différentes compétences comme le génie logiciel, l'ergonomie et la psychologie a fait que ces notations sont hétérogènes. Le besoin d'intégration est un problème majeur. La plupart de ces techniques restent semi-formelles et l'aspect graphique est considéré comme une caractéristique importante puisque qu'elle représente facilement la notation dans son ensemble. Nous avons choisi de diviser ces techniques et notations en deux parties :

– Les techniques et les notations orientées description permettent de décrire les besoins utilisateurs en terme d'utilisabilité d'une interface utilisateur. Elles sont souvent proposées par des non-informaticiens (ergonomie et psychologie par exemple). La majorité de ces notations sont centrées utilisateur et sont donc loin des implantations informatiques. Les besoins utilisateur sont exprimés par un ensemble de tâches qui permettent d'atteindre un but donné à partir d'un état initial. En voici quelques exemples : MAD [Scapin and Pierret-Golbreich 1990; Scapin and Pierret-Golbreich 1989], XUAN [Gray et al. 1994], CTT [Paternò 2001; Paternò et al. 2001] et K-MAD [Lucquiaud 2005; Baron et al. 2006].
– Les techniques et les notations orientées conception permettent d'exprimer la structure statique du logiciel implémentant l'interface utilisateur décrite et/ou du système. Ces techniques et notations séparent l'interface utilisateur du noyau fonctionnel. En voici quelques exemples : le modèle Seeheim [Pfaff 1985], le modèle PAC [Coutaz 1987], le modèle ARCH [Bass et al. 1991], le modèle MVC [Burbeck 1992] et les modèles hybrides [Guittet 1995; Depaulis et al. 2006]. Voir aussi [Kolski et al. 2009] pour une synthèse récente sur les modèles d'architecture à base d'agents.

2.2 Propriétés

Les propriétés dans les systèmes interactifs doivent non seulement vérifier les propriétés propres aux systèmes informatiques classiques (sûreté, équité) mais aussi doivent prendre en compte l'utilisateur (psychologie, comportement, aspects ergonomiques, etc.) ainsi que l'environnement (système d'exploitation, systèmes d'entrée/sortie, etc).

De nombreuses formulations et classifications des propriétés inhérentes aux IHM ont été proposées dans [Harrison and Thimbleby 1990; Dix et al. 1993; Duke and Harrison 1995; Roche 1998].

Nous pouvons les résumer suivant deux grandes classes de propriétés :

– Les propriétés de validité qui caractérisent un fonctionnement attendu ou voulu par un utilisateur. Dans cette catégorie de propriétés, nous pouvons distinguer la complétude (l'utilisateur doit atteindre un objectif donné de différentes manières), la flexibilité pour la représentation de l'information, le déroulement de tâches (atteignabilité, non-préemption, interaction à plusieurs fils), les propriétés liées au temps, les aspects psychologiques et ergonomiques.
– Les propriétés de robustesse qui sont relatives à la sûreté de fonctionnement de l'interface homme-machine et au système en général. Parmi ces propriétés, nous trouvons des propriétés liées à la visualisation comme l'observabilité, l'insistance et l'honnêteté.

À ces propriétés s'ajoutent celles qui sont relatives à un type précis d'IHM et en l'occurrence dans cet article nous traitons les propriétés inhérentes aux systèmes multi-modaux. De nombreuses études ont permis de classifier les actes multi-modaux [Bellik 1995] et nous retiendrons les propriétés CARE (Complémentarité, Assignation, Redondance et Équivalence [Coutaz et al. 1995]). Ces propriétés définissent les propriétés de robustesse et de flexibilité des systèmes mutli-modaux.

3. LES TECHNIQUES FORMELLES EN IHM

Parmi les techniques formelles les plus largement utilisées en génie logiciel, les approches orientées modèles jouent un rôle majeur dans le domaine de l'interaction homme-machine. Ces méthodes sont basées sur la description de modèles au moyen d'un ensemble de variables qui sont modifiées par les opérations et les événements des modèles. Généralement, ces techniques sont divisées en deux catégories : la preuve automatique par vérification sur modèles et les systèmes fondées sur la preuve. Les deux techniques ont été appliquées au domaine de l'IHM.

La première catégorie est basée sur l'évaluation de propriétés logiques d'un système à transitions obtenu à partir des valeurs des variables. Parmi ces techniques, nous citons les logiques temporelles et les réseaux de Petri. Dans le domaine de l'IHM, ces techniques ont été utilisées la première fois pour la vérification formelle de propriétés de systèmes interactifs [Campos and Harrison 1997]. Par exemple, [Abowd et al. 1995] vérifie des interfaces utilisateur avec SMV (Symbolic Model Verifier) utilisant CTL (Computational Tree Logic), tandis que [Faconti et al. 1992] utilise LOTOS pour écrire des spécifications d'interacteur. [Brun 1997] développe un formalisme basé sur une logique temporelle, nommé XTL (eXtended Temporal Logic). La vérification sur modèles (model checking) est également utilisée par [Palanque et al. 1995; Navarre et al. 2005] où l'utilisateur et le système sont modélisés au moyen de réseaux de Petri orientés objets appelés ICO. Dans la même catégorie, [D'Ausbourg 1998] a utilisé un langage à flots de données pour la validation automatique de systèmes interactifs.

La seconde catégorie est basée sur des systèmes de preuve où le modèle est décrit par des variables, des opérations, des événements, des propriétés temporelles et des invariants. Les opérations doivent préserver ces invariants et une famille de propriétés (sûreté, non-blocage, vivacité, etc.). Pour assurer la conformité de ces spécifications, des obligations de preuve sont générées et doivent être prouvées. Selon l'implémentation qui en est faite, le système de preuve peut prouver automatiquement les obligations de preuve. Parmi ces techniques, nous trouvons Z, basée sur la théorie des ensembles [Spivey 1992], VDM

[Bjorner 1987], basée sur le calcul de pré/post-conditions [Hoare 1969], et B, basée sur le calcul de la plus faible pré-condition [Abrial 1996b; Abrial 1996a]. Dans le domaine de l'IHM, VDM et Z ont été utilisées pour la définition de structures atomiques comme les interacteurs [Duke and Harrison 1993a; Duke and Harrison 1993b], et Z et Object-Z sont utilisées maintenant plus intensément, enfin HOL (a Higher Order Logic Theorem Prover) a été utilisée pour la vérification de spécifications d'interfaces utilisateur [Bumbulis et al. 1996].

4. CONCEPTION À BASE DE MODULES

La première approche que nous avons développée exploite la méthode B [Abrial 1996b]. Cette démarche est fondée sur une approche de conception ascendante. Elle a permis la spécification, la vérification et le raffinement de spécifications formelles B. De plus, des programmes écrits en ADA et TCL-TK ont été produits au moyen de cette technique, démontrant ainsi que notre approche peut être utilisée à toutes les étapes du développement de systèmes interactifs [Aït-Ameur et al. 1998; Jambon et al. 2001; Aït-Ameur et al. 2003; Aït-Ameur et al. 2004].

Cette approche est appelée composition à base de modules puisque la décomposition des applications interactives étudiées a été représentée en B suivant le modèle d'architecture imposé par ARCH. Elle est ainsi entièrement basée sur une structuration modulaire des machines abstraites pour composer des sous-systèmes afin d'aboutir au système complexe.

Par ailleurs, cette approche a permis d'apporter une solution pour construire et valider des tâches utilisateurs. La validation de tâches s'appuie sur une démarche comparable aux traces explicites ou aux diagrammes de séquence UML dans le sens où des traces d'opérations sont explicitement construites et validées. L'objectif est d'obtenir une trace d'opérations atomiques (du contrôleur de dialogue) par décomposition en sous tâches où le seul opérateur de contrôle est la séquence. Si ce développement est valide (toutes les obligations de preuve générées ont été prouvées), alors la tâche est valide. Elle permet donc la validation a priori de tâches (faisabilité), c'est-à-dire qu'il existe une séquence d'opérations qui implémente la tâche de plus haut niveau. Elle permet aussi la validation a priori de la conception (complétude), c'est-à-dire que toutes les obligations de preuve de la modélisation de la conception sont prouvées. Cette approche ne nécessite pas de modification de la partie conception. Il y a donc homogénéité des langages de modélisation entre la partie conception et validation.

Toutefois, deux critiques de l'approche à base de modules peuvent être formulées :

(1) la première concerne la phase de conception. L'approche à base de modules ne permet pas de décrire l'état du dialogue d'une application interactive. En effet, elle ne définit qu'un ensemble d'opérations mais pas les événements qui permettent de les déclencher ni leur ordre. La description de systèmes concurrents n'est donc pas possible. Cette lacune est due principalement à la sémantique de B classique qui ne permet pas de décrire des systèmes réactifs comme les IHM. Par ailleurs, la conception ascendante oblige à reprouver à chaque composition des obligations de preuve plus complexes ;

(2) la seconde concerne la phase de validation de tâches. Nous avons montré dans nos travaux que l'établissement de scénarii se fondait sur l'utilisation exclusive de l'opérateur de séquence. Par conséquent l'établissement des traces d'opérations à valider est fastidieux à mettre en place. En effet, en comparaison avec les formalismes possédant une forte capacité à structurer les tâches utilisateur, comme CTT par exem-

ple, notre approche de validation à base de traces doit énumérer tous les cas possible par un seul opérateur alors que l'utilisation de différents opérateurs (interruption, désactivation, entrelacement et itérations) rendrait la validation de tâches plus expressive et complète.

L'approche que nous proposons dans cet article propose une réponse aux deux critiques formulées ci-dessus au moyen de la mise en œuvre de B événementiel.

5. LA MÉTHODE B ÉVÉNEMENTIEL

Nous utilisons la méthode formelle B événementiel [Abrial 2010] pour montrer qu'il est possible de traiter un développement formel complet d'un système interactif. À la différence des autres techniques formelles comme Z [Spivey 1992], la méthode B événementiel est outillée puisqu'il existe des outils comme l'Atelier B[2] et la plateforme Rodin[3]. L'Atelier B a été utilisé dans le cadre de ces travaux. Il permet de générer automatiquement toutes les obligations de preuve et est doté à la fois d'un prouveur automatique et interactif.

5.1 Modèle

Un modèle B événementiel décrit un système de transitions étiquetées (STE) par un ensemble d'états, un ensemble d'actions, un état initial et une relation de transition. Les actions effectuées par le système permettent la transition entre états. Le STE n'est pas manipulé explicitement. Il n'est décrit que via des événements dont le développeur donne la condition de déclenchement (garde de l'événement) et l'action instantanée associée (corps de l'événement). Un modèle est dit clos au sens où il décrit non seulement le système logiciel d'intérêt mais aussi son environnement d'exécution (ou une partie). La sémantique associée est une sémantique à base de traces d'événements. Un modèle est caractérisé par l'ensemble des séquences (traces) d'événements autorisés sous couvert des propriétés énoncées.

Un modèle est constitué de deux parties : une partie dite *statique* correspondant à la définition de l'état et de ses propriétés invariantes ; une partie dite *dynamique* décrivant les transitions.

En résumé, un modèle est composé des clauses suivantes (cf. figure 1) :

- MODEL correspondant à la déclaration du nom du modèle ;
- SETS introduisant des ensembles abstraits ou énumérés d'un modèle ;
- VARIABLES énonçant une liste de noms permettant la déclaration des variables (état) ;
- INVARIANT correspondant à la définition des propriétés des variables : il s'agit de propriétés logiques qui expriment en logique du premier ordre les propriétés invariantes d'un modèle ;
- ASSERTIONS permettant de définir des lemmes : il s'agit d'une liste de prédicats, portant sur les variables, déductibles de l'invariant. Ils correspondent à des lemmes utiles pour la preuve ou pour la compréhension du modèle ;
- INITIALISATION permettant de spécifier, en utilisant des substitutions, l'initialisation des variables (état initial) ;

[2] L'Atelier B est un outil qui permet une utilisation opérationnelle de la méthode B : http://www.atelierb.eu

[3] La plateforme Rodin est un environnement de développement basé sur Eclipse pour la méthode B événementiel : http://www.event-b.org/platform.html

MODEL	**REFINEMENT**
m	r
SETS	**REFINES**
s	m
VARIABLES	**SETS**
v	t
INVARIANT	**VARIABLES**
I	w
ASSERTIONS	**INVARIANT**
A	J
INITIALISATION	**VARIANT**
i	V
EVENTS	**ASSERTIONS**
Spécification des événements du modèle	B
END	**INITIALISATION**
	i'
	EVENTS
	Spécification des événements du raffinement
	END
(a)	(b)

Fig. 1. (a) Forme générale d'un modèle, (b) forme générale d'un raffinement

– EVENTS regroupant une liste d'éléments proches des actions de Back [Back and Sere 1989; Back et al. 2003] et des commandes gardées de Dijkstra [Dijkstra 1976] nommées événements (transitions entre états).

5.2 Les substitutions généralisées

L'initialisation ainsi que les événements sont décrits grâce aux substitutions généralisées. Ces dernières sont fondées sur le principe de plus faible précondition de Dijskstra [Dijkstra 1976]. Étant donnés une substitution S et un prédicat P alors $S[P]$ représente la plus faible précondition qui satisfait P après l'exécution de S. Les substitutions intervenant dans les modèles B événementiel sont définies par les expressions suivantes.

[SKIP] P	$\Leftrightarrow$	P	[1]
$[S_1 \parallel S_2]P$	$\Leftrightarrow$	$[S_1]P \wedge [S_2]P$	[2]
$[x := E]P$	$\Leftrightarrow$	$P(x/E)$	[3]
$[x :\in T]P$	$\Leftrightarrow$	$\forall\, x'.(x' \in T \Rightarrow P(x/x'))$	[4]
[**SELECT** G **THEN** S **END**]P	$\Leftrightarrow$	$G \Rightarrow [S]P$	[5]

$P(x/E)$ représente le prédicat P où toutes les occurrences libres de x sont remplacées par l'expression E.

Les substitutions [1], [2], [3] et [4] représentent respectivement l'événement nul, la substitution parallèle exprimant que les substitutions S_1 et S_2 sont réalisées en parallèle, l'affectation, l'affectation d'un élément quelconque appartenant à l'ensemble T. La substitution [5] est une substitution gardée où S est réalisée sous couvert de la garde G.

5.3 Événements

L'événement correspond à un changement d'état dénotant une transition dans le système modélisé. Il est composé de trois parties (cf. figure 2) : un nom ; une condition appelée

garde notée G ; une action (corps de l'événement) exprimée à l'aide d'une substitution généralisée qui dénote un changement d'état instantané notée S.

nom_événement =
SELECT G
THEN
S
END

Fig. 2. La forme d'un événement

Notons que l'action d'un événement n'est déclenchée que si sa garde est satisfaite. Une fois qu'un événement s'est déclenché, il peut, si sa garde est encore satisfaite, se déclencher à nouveau. Lorsque les gardes de plusieurs événements sont satisfaites simultanément, seul l'un d'entre eux se déclenchera.

événement_simple =
BEGIN
S
END

événement_gardé =
SELECT G
THEN
S
END

événement_non_déterministe =
ANY l
WHERE G
THEN
S
END

Fig. 3. Les 3 formes d'événements

Un événement peut se présenter sous trois formes différentes (cf. figure 3)

(1) un événement est dit *simple* lorsque celui-ci n'est pas gardé (ou autrement dit sa garde est la proposition $vraie$). Il peut toujours se déclencher. En cas de déclenchement, il modifie l'état du système selon la substitution S ;
(2) un événement est dit *gardé* lorsqu'il doit satisfaire une condition G, appelée garde, définie uniquement à partir de variables d'état, pour se déclencher ;
(3) un événement est dit *non déterministe* lorsque celui-ci se déclenche s'il existe des valeurs pour les variables l satisfaisant la condition G. La garde de cet événement est donc de la forme $\exists(l).(G)$ où l est une liste de variables locales distinctes. L'action de l'événement est modélisée par la substitution généralisée S qui peut utiliser les variables l.

Type	Obligation de preuve
(1) Initialisation	$[i]I$
(2) Événement gardé	$I \wedge G \Rightarrow [S]I$
(3) Assertion	$I \Rightarrow A$

Table I. Obligations de preuve pour la consistance d'un modèle

Pour s'assurer qu'un modèle est consistant (voir les obligations de preuve du tableau I), il faut montrer, d'une part, que l'initialisation i établit l'invariant I (1) et d'autre part,

que les modifications engendrées sur l'état par les substitutions S de chaque événement maintiennent l'invariant sous la garde G de l'évènement (2). Si le modèle comporte des assertions, il faudra établir que chaque assertion A est une conséquence de l'invariant I (3).

5.4 Un exemple de modèle B événementiel

Considérons la spécification d'une horloge présentée dans [Cansell 2003] de la figure 4. La spécification abstraite $Clock$ décrit une variable h pour les heures de l'horloge. Deux événements sont décrits. Le premier (événement $incr$) permet d'augmenter la variable h et se déclenche tant que $h \neq 23$. Le second événement est $zero$. Il est déclenché quand $h = 23$ pour réinitialiser la variable h.

5.5 Le raffinement

Un modèle B événementiel peut être raffiné et enrichi par de nouveaux événements et de nouvelles propriétés [Abrial and Hallerstede 2007]. Le processus de raffinement conduit le développeur à la conception finale de l'interface utilisateur, réalisée à la suite de plusieurs étapes de raffinement qui fournissent différents niveaux d'abstraction.

Un raffinement, comme le montre la figure 1, est défini par un nom de modèle déclaré dans la clause REFINEMENT et par sa filiation grâce à la clause REFINES qui indique le nom du modèle raffiné : r est appelé modèle concret et m modèle abstrait. En tant que modèle à part entière, il comporte une partie statique ainsi qu'une partie dynamique. Il est constitué donc des clauses présentées précédemment en section 5.1. A celles vues précédemment s'ajoute la clause VARIANT qui permet de définir les variants décrémentés par les événements : nous aurons l'occasion d'y revenir plus précisément en section 5.5.2. Le raffinement en tant que modèle doit prouver l'établissement et la conservation des propriétés énoncées comme invariantes. Mais il doit aussi assurer que tous les comportements du système concret sont des comportements du système abstrait.

5.5.1 *Le raffinement d'événement.* Chaque événement d'un modèle abstrait est raffiné en un événement plus concret. On retrouve au sein d'un raffinement tous les événements présents dans l'abstraction. Par contre ces mêmes événements peuvent subir des modifications que ce soit au niveau de la garde et/ou au niveau de l'action, pour réduire le non-déterminisme et accroître la précision de la description. Un événement abstrait est dit raffiné par un événement concret lorsque la garde concrète est contrainte plus fortement que la garde abstraite ou lorsque l'invariant concret est préservé par l'action conjointe des deux événements. Étant donné deux événements e et e', ayant respectivement les gardes G, G' et les substitutions S et S', l'obligation de preuve assurant que e' raffine correctement e, sachant que J correspond à l'invariant concret et I à l'invariant abstrait, s'écrit : $I \wedge J \wedge G' \Rightarrow G \wedge [S']\neg[S]\neg J$.

5.5.2 *Ajout de nouveaux événements.* Lors du raffinement, il est possible d'ajouter de nouveaux événements. Il faut donc prouver que les nouveaux événements raffinent bien l'événement vide $SKIP$ de l'abstraction en démontrant l'obligation de preuve suivante : $I \wedge J \wedge G' \Rightarrow [S']J$. De plus, une preuve supplémentaire doit être validée afin de montrer que le nouvel événement ne peut prendre le contrôle indéfiniment. Pour cela il faut montrer que chaque nouvel événement fait décroître un variant, défini au préalable dans la clause VARIANT du raffinement. Etant donné un événement concret e', défini par une garde G' et une substitution S', un invariant abstrait I, un invariant concret j ; sachant que V corre-

MODEL	**REFINEMENT**
$Clock$	$ClockWMinute$
VARIABLES	**REFINES**
h	$Clock$
INVARIANT	**VARIABLES**
$h \in 0..23$	h, m
ASSERTIONS	**INVARIANT**
$h < 100$	$m \in 0..59$
INITIALISATION	**ASSERTIONS**
$h := 13$	$(h \neq 23) \vee (h = 23) \Rightarrow$
	$(h \neq 23 \wedge m = 59) \vee (h = 23 \wedge m = 59) \vee (m \neq 59)$
EVENTS	**VARIANT**
$incr =$	$59 - m$
SELECT $h \neq 23$	**INITIALISATION**
THEN	$h := 13 \parallel m := 14$
$h := h + 1$	**EVENTS**
END;	$incr =$
$zero =$	**SELECT** $h \neq 23 \wedge m = 59$
SELECT $h = 23$	**THEN**
THEN	$h := h + 1 \parallel m := 0$
$h := 0$	**END;**
END	$zero =$ | $ticTac =$
END	**SELECT** $h = 23 \wedge m = 59$ | **SELECT** $m \neq 59$
	THEN | **THEN**
	$h := 0 \parallel m := 0$ | $m := m + 1$
	END; | **END**

Fig. 4. Modèle B événementiel d'une horloge et son raffinement

spond au variant et V_0 au variant avant le déclenchement de e', il faut montrer l'obligation de preuve suivante : $(I \wedge J \Rightarrow V \in \mathbb{N}) \wedge (I \wedge J \wedge G' \Rightarrow [S'](V < V_0))$.

5.5.3 *Le raffinement de modèle.* Un modèle concret raffine correctement un modèle abstrait s'il ne se bloque pas plus que le modèle abstrait. Pour cela nous devons montrer que la disjonction des gardes abstraites implique la disjonction des gardes concrètes. Sachant que les G_i représentent les gardes abstraites et les G'_i les gardes concrètes, l'obligation de preuve à montrer sera donc de la forme : $I \wedge J \wedge (G_1 \vee \ldots \vee G_m) \Rightarrow G'_1 \vee \ldots \vee G'_n$.

5.6 Un exemple de raffinement B événementiel

Dans la spécification du raffinement $ClockWMinute$ (cf. figure 4), nous introduisons une nouvelle variable m, dénotant les minutes, et un nouvel événement $ticTac$. Ce dernier incrémente la variable m tant que $m \neq 59$. Avant de démontrer qu'il ne prend pas le contrôle indéfiniment, nous montrons que le variant $59 - m$ décroît. L'introduction des minutes a un impact sur les événements $incr$ et $zero$. Nous enrichissons leurs gardes en ajoutant la contrainte $m = 59$ et aussi leurs actions en ajoutant la réinitialisation de la variable m. Une condition de non blocage est exprimée dans la clause ASSERTIONS qui, comme nous l'avons dit précédemment, consiste à vérifier que la disjonction des gardes abstraites implique la disjonction des gardes concrètes.

Task ::=	$Task >> Task$	- - Sequence
|	$Task[]Task$	- - Choix
|	$Task\|\|Task$	- - Parallèle
|	$Task \models Task$	- - Ordre indépendant
|	$[Task]$	- - Tâche optionnelle
|	$Task[> Task$	- - Désactivation
|	$Task\| > Task$	- - Interruption
|	$Task^*[> Task$	- - Désactivation d'une tâche infinie
|	$Task^N$	- - Tâche itérative finie
|	$Task_{At}$	- - Tâche atomique

Fig. 5. Grammaire de la notation CTT.

6. LA VALIDATION DE TÂCHES

Les insuffisances de l'approche de validation à base de traces explicites évoquées en section 4 nous ont conduit à proposer une extension où nous utilisons un langage de modélisation de tâches utilisateurs défini par la communauté IHM, en l'occurrence ConcurTaskTrees (CTT), qui permet :

– de décrire des tâches complexes par des expressions combinées à des opérateurs temporels ;

– d'éviter la définition de traces pour chaque tâche grâce à l'introduction d'autres opérateurs de contrôle. Il sera inutile d'énumérer toutes les séquences possibles de tâches à valider. Elles sont simplement caractérisées par le modèle de tâches CTT.

6.1 La notation de modélisation de tâches CTT

CTT (ConcurTaskTrees [Paternò 2001]) a été conçu pour décrire l'activité de l'utilisateur jusqu'au niveau de l'interaction. Il est résolument orienté vers la description de systèmes interactifs. Il permet de décrire des tâches utilisateurs en les combinant avec des opérateurs temporels. Un modèle de tâches CTT est basé sur une hiérarchie de tâches semblable à une structure d'arbre. Chaque tâche d'un même niveau est composée par un opérateur temporel qui détermine l'ordonnancement de ce niveau.

Sur la figure 5, la grammaire décrivant la syntaxe de la notation CTT est donnée. Elle présente des opérateurs temporels d'une algèbre de processus classique à la CCS [Milner 1980].

Une représentation graphique d'un arbre CTT est également associée à cette notation. Par exemple, une représentation graphique de l'expression donnée ci-dessous est présentée sur la figure 6. Elle a été produite par l'outil CTTE [Paternò et al. 2001] pour la tâche T_0 décomposée par la suite.

Dans les prochaines sections, nous montrons comment la sémantique de la notation CTT peut être formellement décrite en B événementiel. Des règles de traduction génériques sont proposées pour chaque opérateur.

6.2 Règles génériques pour la traduction des opérateurs de CTT

Cette approche utilise la capacité du raffinement de B événementiel. Pour toutes les règles de traduction, var_i représente l'état des variables, T_i le processus (la tâche), G_i la garde

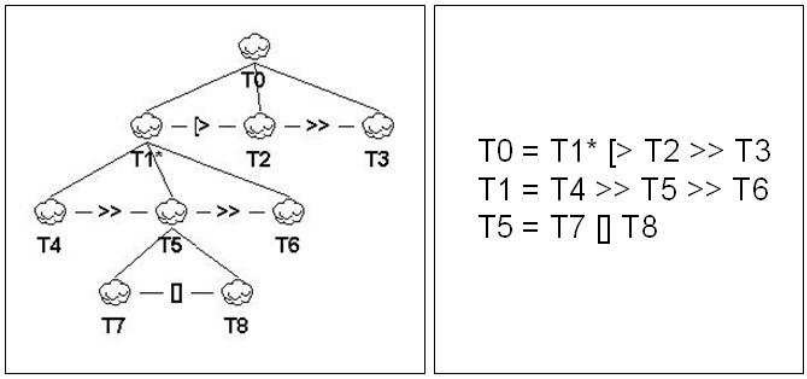

Fig. 6. Une représentation graphique d'une expression CTT obtenue avec l'outil CTTE

des événements et S_i une substitution généralisée correspondant aux actions effectuées par le processus T_i.

Chaque opérateur de CTT est représenté par un raffinement. Ainsi une règle BNF de la forme $T_0 ::= T_1 op T_2$ est décomposée en deux machines : la première contient l'événement T_0 et la seconde raffinant la première contient les événements T_0, T_1 et T_2 et effectue la décomposition de T_0 en T_1 op T_2 et où op est un opérateur CTT.

Les règles de traduction en B événementiel des opérateurs basiques (séquence, choix, parallèle, itération finie et tâche atomique) sont données dans cette section. Ces opérateurs sont utilisés pour coder les opérateurs restants (ordre indépendant, tâche optionnelle, désactivation, désactivation d'une tâche itérative et interruption).

MODEL T_0
INVARIANT $I(var_i)$
EVENTS
$EvtT_0 =$
SELECT
G_0
THEN
S_0
END;

Fig. 7. Modélisation de la tâche racine T_0

Pour ces règles de traduction, nous utilisons une tâche T_0 représentée par l'événement $EvtT_0$ (figure 7), considérée comme la tâche racine. Cette tâche est décomposée pour traduire les opérateurs basiques. L'ensemble des variables var_i décrit tous les états des variables qui caractérisent le processus de la tâche T_0. D'autres variables peuvent être ajoutées au niveau du raffinement, si besoin, pour observer des nouveaux éléments introduits par la décomposition. S_0 est une substitution, sous couvert que la garde G_0 est vraie, qui exprime le changement d'état de var_i pour le processus de la tâche T_0.

RAFFINEMENT $RefSequenceT_0$
REFINE T_0
INVARIANT
$J(var_i, var_j) \wedge EtatAct \in \{0,1,2\} \wedge J'(var_j)$
ASSERTIONS
$G_0 \Rightarrow ((EtatAct = 2 \wedge G_1) \vee (EtatAct = 1 \wedge G_2) \vee (EtatAct = 0 \wedge G'_0))$
VARIANT
$EtatAct$
INITIALISATION
$EtatAct := 2$
EVENTS

$EvtT_1 =$	$EvtT_2 =$	$EvtT_0 =$
SELECT	**SELECT**	**SELECT**
$EtatAct = 2 \wedge G_1$	$EtatAct = 1 \wedge G_2$	$EtatAct = 0 \wedge G'_0$
THEN	**THEN**	**THEN**
$EtatAct := 1 \parallel$	$EtatAct := 0 \parallel$	S'_0
S_1	S_2	**END;**
END;	**END;**	

Fig. 8. Codage en B événementiel de l'opérateur séquence

6.2.1 *Séquence ">>".* Considérons la tâche suivante $T_0 ::= T_1 >> T_2$ qui exprime l'activation de la tâche T_1 suivie de la tâche T_2. La traduction en B événementiel est donnée par un modèle (cf. figure 8) avec deux événements $EvtT_1$ et $EvtT_2$ correspondant respectivement aux tâches T_1 et T_2. La traduction utilise un variant $EtatAct$ initialisé à 2 et qui décroît jusqu'à la valeur 0. Si la garde de $EvtT_1$ est vraie, l'événement est déclenché, sa substitution généralisée est exécutée et le variant décroît. Ensuite, l'événement $EvtT_2$ se déclenche si sa garde est vraie et si la valeur du variant est mise à 1 par $EvtT_1$. La disjonction des gardes est donnée dans la clause ASSERTIONS. La variable var_j définit l'état des variables du raffinement. Elles sont liées aux variables abstraites du modèle par l'invariant de collage $J(var_i, var_j)$. Cet invariant de collage est défini dans tous les raffinements suivants.

Notez que l'événement $EvtT_0$ termine l'opération de séquence des tâches T_1 et T_2. L'invariant de collage assure que l'événement $EvtT_0$ du raffinement raffine celui de l'abstraction.

6.2.2 *Choix "[]".* Considérons la tâche $T_0 ::= T_1 \; [] \; T_2$ définissant un choix non déterministe entre la tâche T_1 et T_2, c'est-à-dire que T_1 ou T_2 est déclenchée. La traduction en B événementiel est donnée par un modèle (cf. figure 9) avec deux événements gardés $EvtT_1$ et $EvtT_2$. Le variant $EtatChoix$ est arbitrairement initialisé soit à 1 soit à 2. Selon la valeur de la garde de chaque événement, $EvtT_1$ ou $EvtT_2$ est déclenché. Chaque événement fait décroître la valeur du variant à la valeur 0 interdisant les autres événements de se déclencher. L'événement $EvtT_0$ finalise le codage de l'opérateur choix entre les deux tâches. Il décrit le raffinement de l'événement correspond à la tâche T_0.

6.2.3 *Tâche itérative finie "T^N".* Considérons la tâche itérative suivante $T_0 ::= T_1^N$. Le principe de codage d'une boucle en B événementiel consiste à déclencher N fois les événements associés à la tâche T_1. Un variant est utilisé. Il décroît de N vers 0. La traduction d'une boucle codée en B événementiel (cf. figure 10) nécessite trois événements : un

RAFFINEMENT $RefChoixT_0$
REFINE T_0
INVARIANT
$J(var_i, var_j) \wedge EtatChoix \in \{0,1,2,3\} \wedge J'(var_j)$
ASSERTIONS
$G_0 \Rightarrow ((\exists(p).(p \in \{1,2\} \wedge EtatChoix = 3)) \vee (G_1 \wedge EtatChoix = 1) \vee (G_2 \wedge EtatChoix = 2) \vee (EtatChoix = 0 \wedge G'_0))$
VARIANT
$EtatChoix$
INITIALISATION
$EtatChoix := 3$
EVENTS

$Evt_{InitChoix} =$	$EvtT_1 =$	$EvtT_2 =$	$EvtT_0 =$
ANY p	**SELECT**	**SELECT**	**SELECT**
WHERE	$EtatChoix = 1 \wedge G_1$	$EtatChoix = 2 \wedge G_2$	$EtatChoix = 0 \wedge G'_0$
$p \in \{1,2\} \wedge$	**THEN**	**THEN**	**THEN**
$EtatChoix = 3$	$EtatChoix := 0 \parallel S_1$	$EtatChoix := 0 \parallel S_2$	S'_0
THEN	**END;**	**END;**	**END;**
$EtatChoix := p$			
END;			

Fig. 9. Codage en B événementiel de l'opérateur choix

RAFFINEMENT $RefIterationFinieT_0$
REFINE T_0
INVARIANT
$J(var_i, var_j) \wedge EtatLoop \in NAT \wedge Debut \in \{0,1\} \wedge J'(var_j)$
ASSERTIONS
$G_0 \Rightarrow (Debut = 1 \vee (EtatLoop > 0 \wedge Debut = 0 \wedge G_1) \vee (EtatLoop = 0 \wedge Debut = 0 \wedge G'_0))$
VARIANT
$EtatLoop + Debut$
INITIALISATION
$Debut := 1$
EVENTS

$Evt_{InitLoop} =$	$Evt_{Loop} =$	$Evt_0 =$
SELECT	**SELECT**	**SELECT**
$Debut = 1$	$EtatLoop > 0 \wedge Debut = 0 \wedge G_1$	$EtatLoop = 0 \wedge Debut = 0 \wedge G'_0$
THEN	**THEN**	**THEN**
$EtatLoop :\in NAT \parallel Debut := 0$	$EtatLoop := EtatLoop - 1 \parallel S_{Loop}$	S'_0
END;	**END;**	**END;**

Fig. 10. Codage en B événementiel de la caractéristique itération finie

premier événement $Evt_{InitLoop}$ pour l'initialisation du variant $EtatLoop$, un deuxième événement Evt_{Loop} pour exécuter le corps de la boucle et décroître le variant, et enfin un troisième événement $EvtT_0$ pour terminer la boucle. Le variant $EtatLoop$ modélisant le nombre d'itérations est initialisé par l'événement $Evt_{InitLoop}$. L'événement Evt_{Loop} est alors déclenché $EtatLoop$ fois. Quand l'itération est terminée, Evt_0 se déclenche. Le variant $EtatLoop$ décroît de sa valeur arbitraire jusqu'à 0. La clause ASSERTIONS assure qu'au moins une garde d'un des événements est toujours respectée. La valeur initiale du variant est arbitrairement fixée par l'opérateur :∈. L'avantage d'une telle approche est la possibilité d'encoder un nombre arbitraire d'une boucle sans augmenter la complexité du processus de preuve. À l'inverse, les techniques formelles dites par vérification sur modèles souffriraient du problème d'explosion combinatoire des états lors de l'augmentation du nombre d'itérations.

RAFFINEMENT $RefParalleleT_0$
REFINE T_0
INVARIANT
$J(var_i, var_j) \wedge EtatConc_1 \in \{0,1\} \wedge EtatConc_2 \in \{0,1\} \wedge J'(var_j)$
ASSERTIONS
$G_0 \Rightarrow ((G_1 \wedge EtatConc_1 = 1) \vee (G_2 \wedge EtatConc_2 = 1) \vee (EtatConc_1 = 0 \wedge EtatConc_2 = 0 \wedge G'_0))$
VARIANT
$EtatConc_1 + EtatConc_2$
INITIALISATION
$EtatConc_1 := 1 \parallel EtatConc_2 := 1$
EVENTS

$EvtT_1 =$	$EvtT_2 =$	$EvtT_0 =$
SELECT	**SELECT**	**SELECT**
$G_1 \wedge EtatConc_1 = 1$	$G_2 \wedge EtatConc_2 = 1$	$EtatConc_1 = 0 \wedge$
THEN	**THEN**	$EtatConc_2 = 0 \wedge G'_0$
$EtatConc_1 = 0 \parallel$	$EtatConc_2 = 0 \parallel$	**THEN**
S_1	S_2	S'_0
END;	**END;**	**END;**

Fig. 11. Codage en B événementiel de l'opérateur parallèle

6.2.4 *Parallèle "||".* Considérons la tâche suivante $T_0 ::= T_1||T_2$. La sémantique du parallélisme par entrelacement impose de décrire tous les comportements possibles en énumérant donc toutes les traces. Cet opérateur exploite l'entrelacement mis en avant par la sémantique du B événementiel, i.e. si deux événements ont leur garde à vraie, ils sont déclenchés en parallèle d'une manière entrelacée (cf. figure 11). Deux événements $EvtT_1$ et $EvtT_2$ correspondant aux tâches T_1 et T_2 sont définis. Une fois que les événements $EvtT_1$ et $EvtT_2$ sont déclenchés, ils ne peuvent pas être déclenchés de nouveau. Leur variant (respectivement $EtatConc_1$ et $EtatConc_2$) décroît de un à zéro.

6.3 Traduction des autres opérateurs temporels

Les opérateurs de séquence, de parallélisme, de choix et la caractéristique itération finie ont été codés dans des modèles B événementiels. Il existe d'autres opérateurs (or-

dre indépendant, tâche optionnelle, désactivation). Leur traduction est obtenue en utilisant les opérateurs basiques définis précédemment en exploitant également la sémantique d'entrelacement de la méthode B événementiel. Nous invitons les lecteurs à parcourir la référence suivante de façon à trouver la description exhaustive de la traduction de la grammaire CTT [Baron 2003; Aït-Ameur et al. 2009].

6.4 Méthodologie de conception

La conception d'un modèle B événementiel à partir d'une expression de tâche avec CTT par l'application des règles proposées dans cet article permet de vérifier et de valider un système interactif d'une manière incrémentale. Notre objectif est de représenter chaque tâche abstraite de CTT réalisée par un ensemble de tâches concrètes ou abstraites (ajout de nouvelles tâches), ordonnées par les différents opérateurs CTT, par l'opération du raffinement de B événementiel (ajout de nouveaux événements).

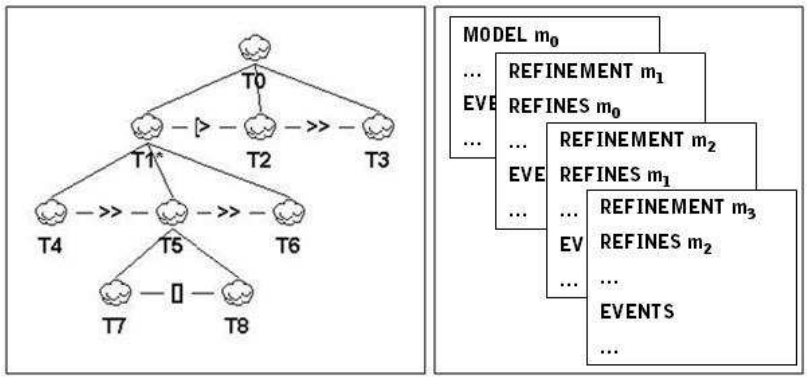

Fig. 12. Différents modèles B événementiel obtenus à partir d'une tâche CTT

Sur l'exemple de la figure 12, chaque tâche Ti est spécifiée par un évènement nommé Evt_{Ti}. L'application de notre méthodologie sur l'exemple aboutit au départ à un modèle abstrait nommé m_0 contenant l'événement Evt_{T0}. En appliquant les règles définies dans la section 6.2, l'évènement Evt_{T0} est raffiné par les évènements Evt_{T1}, Evt_{T2} et Evt_{T3} dans un modèle m_1 lié au modèle m_0 par raffinement. Par la suite, l'évènement Evt_{T1} est raffiné par les évènements Evt_{T4}, Evt_{T5} et Evt_{T6} dans le modèle m_2 lié également au modèle m_1 par raffinement. De la même manière, ce processus se déroulera jusqu'à ce qu'il n'y ait plus de tâche CTT qui se décompose en sous tâches.

7. L'ARCHITECTURE LOGICIELLE

La grande majorité des modèles d'architecture de systèmes interactifs distingue trois composants essentiels : le noyau fonctionnel, la présentation et le contrôleur de dialogue. Ces modèles séparent ces trois composants pour des raisons de modularité et pour permettre une validation et vérification compositionnelles. Cette partie s'intéresse à la modélisation et à la vérification formelles du composant contrôleur de dialogue. Plusieurs scénarios de développement de ce composant sont possibles.

7.1 Architecture logicielle et B événementiel

Nous présentons en premier la modélisation, en B événementiel, des composants noyau fonctionnel et présentation. Chacun de ces composants est un système de transitions. Ce système est modélisé en B par un état déclaré dans la clause VARIABLES et un ensemble d'événements dans la clause EVENTS. Ces événements décrivent les changements d'état (les transitions). Les clauses INVARIANT et ASSERTIONS comportent les propriétés pertinentes de ces systèmes. Enfin, ces systèmes sont construits par une suite de raffinements qui préservent les propriétés. On notera $MODEL_{NF}$ le modèle B décrivant le noyau fonctionnel et $MODEL_{IHM}$ celui qui décrit la présentation (cf. figure 13).

MODEL	**MODEL**
$MODEL_{NF}$	$MODEL_{IHM}$
VARIABLES	**VARIABLES**
Var_{NF}	Var_{IHM}
INVARIANT	**INVARIANT**
$I(Var_{NF})$	$I(Var_{IHM})$
ASSERTIONS	**ASSERTIONS**
$A(Var_{NF})$	$A(Var_{IHM})$
INITIALISATION	**INITIALISATION**
$Var_{NF} := \ldots$	$Var_{IHM} := \ldots$
EVENTS	**EVENTS**
$Evt =$	$Evt =$
SELECT G_{NF}	**SELECT** G_{IHM}
THEN	**THEN**
S_{NF}	S_{IHM}
END	**END**
END	**END**

Fig. 13. Modèles décrivant respectivement le noyau fonctionnel et la présentation

Le contrôleur de dialogue est représenté par le produit synchronisé des deux systèmes de transitions représentant respectivement le noyau fonctionnel et la présentation. L'état du composant obtenu est une paire Var = (Var_{NF}, Var_{IHM}) composée des variables du noyau fonctionnel et de celles de la présentation et les événements sont de la forme présentée sur la figure 14.

G_{NF} et S_{NF} correspondent respectivement à la garde et à la substitution de la partie noyau fonctionnel. À l'inverse, G_{IHM} et S_{IHM} correspondent à la garde et à la substitution de la partie interface utilisateur. Notons que des événements d'attente peuvent apparaître dans le cas ou S_{NF} et S_{IHM} valent SKIP.

La preuve des modèles B obtenus dépend de la façon de construire ce produit. Cette construction exploite le raffinement. Nous avons identifié quatre scénarios de conception d'IHM. Chacun décrit une implémentation de la composition des composants noyau fonctionnel et de la présentation pour définir le contrôleur de dialogue :

(1) description des raffinements B du noyau fonctionnel puis introduction des événements de la présentation par raffinement ;

(2) description des raffinements B de la présentation puis introduction des événements du noyau fonctionnel par raffinement ;

(3) composition des événements du noyau fonctionnel et de la présentation dans le même modèle B ;
(4) description des raffinements B de la présentation avec un noyau fonctionnel abstrait.

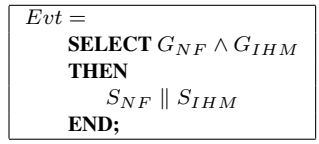

Fig. 14. Forme d'un événement du contrôleur de dialogue

Dans la suite nous montrons uniquement le scénario quatre que nous avons appliqué à notre étude de cas. Une description exhaustive des scénarios peut être trouvée dans les références suivantes [Aït-Ameur et al. 2006] et [Aït-Ameur et al. 2008].

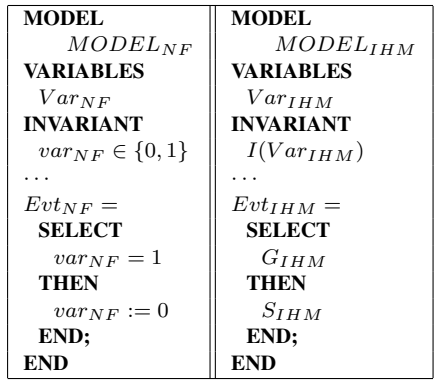

Fig. 15. Modèles décrivant respectivement l'abstraction du noyau fonctionnel et la présentation pour le scénario 4

7.2 Abstraction du noyau fonctionnel

Ce scénario est obtenu par une description des raffinements B de la présentation. Nous considérons la partie noyau fonctionnel comme étant abstraite. Nous ne nous intéressons pas aux activités effectuées par le noyau fonctionnel ni à leur spécification. Nous enregistrons, au niveau du contrôleur de dialogue, le fait qu'un événement du noyau fonctionnel a été déclenché. Pour cela, nous employons une variable entière avec un comportement booléen, qui détermine si l'état du noyau fonctionnel a été modifié ou pas par le déclenchement d'un événement.

Nous donnons sur la figure 15 deux modèles concernant les parties noyau fonctionnel et interface utilisateur. Sur la gauche, nous utilisons la variable Var_{NF} pour abstraire l'état du noyau fonctionnel. Si l'événement Evt_{NF} est déclenché, c'est-à-dire quand Var_{NF} est égal à un, l'état du noyau fonctionnel est modifié à zéro.

Nous montrons sur la figure 16 le résultat de la composition les deux modèles précédents obtenus après raffinement.

MODEL
$MODEL_{CD}$
VARIABLES
Var_{NF}, Var_{IHM}
INVARIANT
$Var_{NF} \in \{0,1\} \wedge I(Var_{IHM})$
...
$Evt =$
SELECT $\mathbf{G_{IHM}} \wedge \mathbf{Var_{NF} = 1}$
THEN
$\mathbf{S_{IHM}} \parallel \mathbf{Var_{NF} := 0}$
END;
END

Fig. 16. Modèle décrivant la composition de l'abstraction du noyau fonctionnel et de la présentation pour le scénario 4

8. APPLICATIONS AUX IHM3

La modélisation d'une IHM3 en B événementiel suit une démarche de conception descendante fondée sur le raffinement. Des événements de haut niveau sont déclarés dans le modèle abstrait racine. Ce dernier est raffiné par l'introduction de nouveaux événements qui précisent le modèle abstrait initial. Nous avons appliqué nos travaux à l'étude de cas "Pages Jaunes", décrite ci-après, et avons retenu le scénario 4. Nous avons utilisé une abstraction du noyau fonctionnel permettant une vérification formelle modulaire.

Nous détaillons dans la section qui suit une modélisation de l'IHM3 représentant un modèle de l'étude de cas présentée ci-dessus correspondant au scénario 4 qui ne prend en compte que la multi-modalité en entrée.

8.1 Étude de cas : "Pages Jaunes CLIPS"

L'étude de cas "Pages Jaunes CLIPS", développée par l'équipe IIHM du laboratoire CLIPS-IMAG, est une application multi-modale implémentée en utilisant la plate-forme de composants d'interactions multi-modales ICARE (Interaction-Care [Bouchet et al. 2004]). Cette application (cf. figure 17) permet de rechercher un contact en spécifiant le nom (Name) d'une personne et une adresse (Address) de localisation. Suite à l'envoi de la requête de recherche de la personne (Search), un plan est affiché. Sur ce plan, l'utilisateur peut naviguer et cibler sa recherche. L'application "Pages Jaunes CLIPS" regroupe un ensemble de commandes permettant à l'utilisateur d'accomplir la tâche (spécifier nom, spécifier adresse, envoyer requête, déplacement haut/bas/gauche/droite sur la carte, agrandissement avant/arrière, ...). Il peut déclencher ces commandes en utilisant différentes modalités d'interaction : la voix, la souris et le clavier, ou une combinaison de ces modalités, en s'appuyant sur l'utilisation individuelle et synergique des différentes modalités d'entrées.

De cette manière l'utilisateur peut remplir ces champs en utilisant uniquement la voix en prononçant le nom de la personne et le lieu, ou en combinant de façon complémentaire les modalités de la voix, de la souris et du clavier (sélection du champs puis prononciation du nom ou alors en choisissant oralement le champs et en saisissant la valeur par le clavier). Les modalités peuvent également être utilisées de manière redondante dans le cas par exemple où l'utilisateur spécifie par la voix la commande "agrandissement" et en

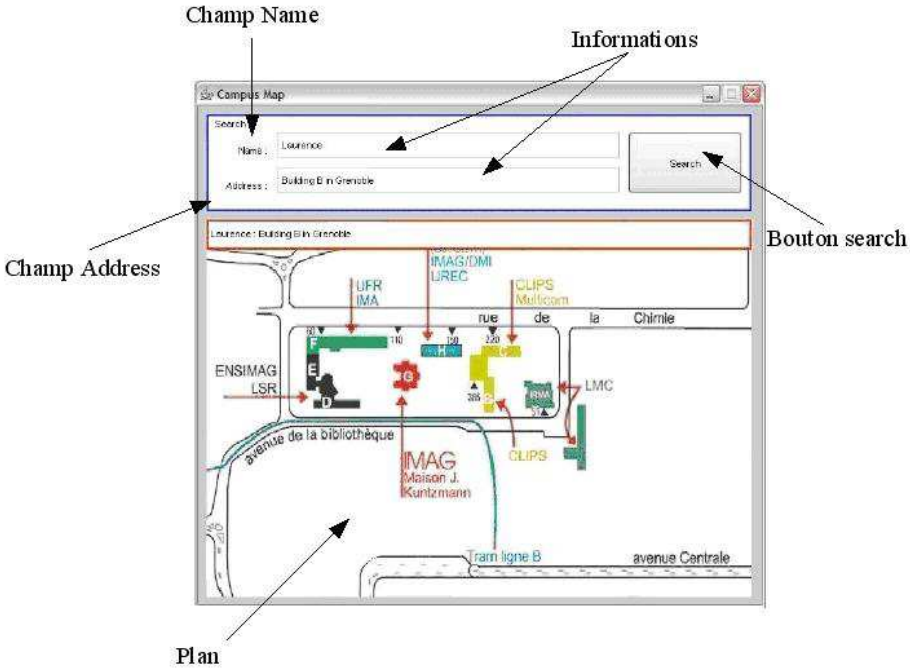

Fig. 17. IHM de l'étude de cas "Pages Jaunes Clips".

parallèle appuie sur la touche du clavier correspondante à cet effet, une seule commande d'agrandissement sera alors transmise au système.

Notons que nous ne nous intéressons dans cet article qu'à la multi-modalité en entrée (de l'utilisateur vers le système "Pages Jaunes CLIPS"), c'est pourquoi, nous ne détaillerons pas la multi-modalité en sortie.

8.2 Modélisation par raffinement

Trois modèles ont été définis. Ils sont reliés par une relation de raffinement. Le premier introduit les événements de haut niveau. Le deuxième présente les événements de la présentation ainsi que leur synchronisation avec le CD. Enfin, le troisième introduit les interactions multi-modales avec les différentes possibilités d'interaction. Ces modèles font tous abstraction du noyau fonctionnel. Ils ne font que rendre compte du fait que des événements abstraits du noyau fonctionnel ont été déclenchés.

Dans la suite, nous décrivons une représentation simplifié du développement de l'IHM3 associée à l'étude de cas "Pages Jaunes CLIPS". Nous avons volontairement réduit les modèles B événementiel afin d'en faciliter la lecture. Le code complet peut être demandé directement aux auteurs.

8.2.1 *Le modèle racine.* Le modèle racine est un modèle abstrait représentant la synchronisation entre les événements de saisie et de déclenchement du noyau fonctionnel (cf. figure 18). Deux événements $Query$ et $Result\_query$ décrivent ce modèle. Le premier indique ($Query\_sending := 1$) que la requête est prête à être envoyée au noyau fonction-

MODEL IMAPPY
VARIABLES
$Query\_sending, Query\_ready$
INVARIANT
$Query\_sending \in 0..1 \wedge Query\_ready \in 0..1 \wedge$
$Query\_sending \neq Query\_ready$
ASSERTIONS
$(Query\_ready = 1) \vee (Query\_sending = 1)$
INITIALISATION
$Query\_sending \parallel Query\_ready := 0, 1$

EVENTS
$Query =$
$SELECT\ Query\_ready = 1$
$THEN$
$Query\_sending := 1 \parallel Query\_ready := 0$
$END;$

$Result\_query =$
$SELECT\ Query\_sending = 1$
$THEN$
$Query\_sending := 0 \parallel Query\_ready := 1$
END
END

Fig. 18. Modèle racine représentant la synchronisation entre les événements de saisis et déclenchement du noyau fonctionnel

nel alors que le second indique que le résultat de la requête est prêt ($Query\_ready := 1$). Les deux variables $Query\_sending$ et $Query\_ready$ décrivent la synchronisation de ces deux événements. Notons que les événements décrits ici sont abstraits (aucun élément relatif aux différentes modalités d'interaction) et n'indiquent que l'ordre correct de déclenchement. Une première propriété est établie à ce niveau : le noyau fonctionnel n'est sollicité qu'après avoir déclenché l'événement de l'interface.

La suite de cet article s'intéresse au raffinement de l'événement Query qui décrit la saisie du nom et de l'adresse selon différentes modalités.

8.2.2 *Identification des interactions abstraites.* L'événement $Query$ est décomposé de sorte à prendre en compte les différentes possibilités de saisie du nom et de l'adresse de la personne à rechercher. Ainsi, on pourra :

– saisir le nom un nombre arbitraire de fois : $Input\_Name^*$;
– saisir l'adresse un nombre arbitraire de fois : $Input\_Address^*$;
– lancer ensuite la recherche : $Search$.

Notons que l'on peut entrelacer la saisie du nom et de l'adresse (saisir l'un puis l'autre dans n'importe quel ordre et retour). Pour décrire les différentes tâches interactives que l'utilisateur peut effectuer sur le système des "Pages Jaunes CLIPS", nous utilisons les opérateurs temporels de la notation CTT. Ainsi la tâche CTT qui décrit les différentes possibilités de saisie est définie par :

$$Query = (Input\_Name^* || Input\_Address^*) >> Search >> Result\_query \quad (1)$$

La tâche $Query$ est implantée dans le raffinement $IMAPPY\_Ref\_1$ décrit sur les figures 19 et 20. Nous nous concentrons sur les événements introduits pour raffiner les deux événements principaux du niveau abstrait.

REFINEMENT IMAPPY_Ref_1
REFINES IMAPPY
SETS
$string$
VARIABLES
$Query\_sending, EV1, Nn, Na,$
$map, name, address$
INVARIANT
$EV1 \in 0..3 \wedge Nn \in N \wedge Na \in N \wedge$
$map \in BOOL \wedge name \subseteq string \wedge address \subseteq string \wedge$
$(Query\_ready = 1 \Rightarrow EV1 \neq 0)$
INITIALISATION
$Query\_sending := 0 \parallel map, EV1 := FALSE, 3 \parallel name, address, Nn, Na := 0, 0, 1, 1$

Fig. 19. Raffinement de la tâche Query

Nous avons introduit de nouvelles variables pour la description de l'interaction en entrée. Tout d'abord, nous décrivons les variables $name$ et $address$ de type $string$ qui récupèrent respectivement le nom et l'adresse. Les variables entières Nn et Na désignent le nombre arbitraire d'itérations * de CTT. Elles sont initialisées, dans l'événement $Name\_Address$, par un nombre entier arbitraire (grâce à l'opérateur $:\in$) qui indique le nombre de fois que cet événement est déclenché. Enfin, la variable $EV1$ (variable entière décroissante) initialisée à 3 est un variant qui décrit l'ordre de déclenchement de chaque événement.

Les événements de la figure 20 décrivent la tâche CTT précédente. L'événement $Name\_Address$ initialise les variants Nn et Na alors que les trois événements $Input\_Name$, I$nput_Address$ et $Search$ décomposent (raffinent) l'événement abstrait $Query$. Lorsque ces trois événements ont été déclenchés, ils rendent le contrôle à l'événement abstrait $Query$ qui ne fait que le transmettre aux autres événements.

Notons qu'à ce niveau, nous avons pris en compte les déclenchements des interactions sans entrer dans les détails de leurs définitions. Cela sera abordé dans les prochains raffinements et nous permet de nous concentrer sur la correction des synchronisations et des ordres de déclenchement.

Comme nous l'avions déclaré précédemment, nous ne nous intéressons pas à la multimodalité en sortie ni au noyau fonctionnel. C'est pour cette raison que les événements associés ($Result\_Query$) n'ont pas été raffinés. Cette démarche montre qu'il est possible de développer les différents aspects d'une IHM3 de façon modulaire afin de simplifier le processus de preuve. En effet, les obligations de preuve engendrées sont allégées du fait que certains événements restent abstraits.

8.2.3 *Identification des modalités.* La dernière étape de raffinement consiste à identifier et à introduire les différentes interactions multi-modales en entrée mises en jeu (cf. figure 21 et figure 22). Ainsi, on décrit les ensembles $SPEECH$, $KEYBOARD$ et $MOUSE$ indiquant et typant les différentes interactions multi-modales possibles. Notons que ces ensembles peuvent être enrichis ou réduits sans remettre en cause la modélisation

EVENTS
Name_Address =
SELECT $EV1 = 3$
THEN
$EV1 := EV1 - 1 \parallel Nn :\in N \parallel Na :\in N$
END;

Input_Name =
SELECT $EV1 = 2 \wedge Nn \neq 0$
THEN
$Nn := Nn - 1 \parallel name :\in P(string)$
END;

Input_Address =
SELECT $EV1 = 2 \wedge Na \neq 0$
THEN
$Na := Na - 1 \parallel address :\in P(string)$
END;

Search =
SELECT $EV1 = 2 \wedge Nn = 0 \wedge Na = 0$
THEN
$EV1 := 1$
END;

Query =
SELECT $EV1 = 1$
THEN
$Query\_sending, EV1 := 1, 0$
END;

Result_query = ...

END

Fig. 20. Raffinement de la tâche Query

effectuée. La variable $modality$ décrit la modalité activée à un instant donné. Elle est initialisée à $NONE$. La variable $EV11$ quant à elle indique que le focus est sur la saisie du nom.

Nous procédons à la décomposition (raffinement) de l'événement $Input\_Name$ et $Input\_Address$ qui font intervenir les modalités voix et clavier/souris. Pour cela, il est nécessaire d'enrichir le modèle B par la déclaration de nouveaux événements.

Le nom est entré en deux étapes : saisie du champ à saisir (le nom) par la voix ou le clavier/souris puis entrée du nom (valeur du nom) par la voix ou bien par clavier/souris. Ces deux événements peuvent être itérés un nombre arbitraire de fois. Les suffixes $field$ et $value$ indiquent respectivement les événements associés au choix du champ à remplir et à la valeur entrée pour ce champ. Dans la suite, seuls les événements raffinant l'événement $Input\_Name$ sont présentés selon la tâche :

$$Input\_name^* = (Input\_Name\_Fied >> Input\_Name\_Value)^* \quad (2)$$

REFINEMENT IMAPPY_ref_2
REFINES IMAPPY_ref_1

SETS
$HCI = \{NONE, NAME, INPUT, SEARCH, ZOOM\_IN, ZOOM\_OUT, HERE, INPUT\_TEXT, ENTER\_KEY, LEFT\_ARROW, RIGHT\_ARROW, UP\_ARROW, DOWN\_ARROW, CLICK\_ON\_NAME, CLICK\_ON\_ADDRESS, CLICK\_ON\_SEARCH, CLICK\_ON\_MAP\}$

CONSTANTS
$SPEECH, KEYBOARD, MOUSE$

PROPERTIES
$SPEECH \subseteq HCI \wedge KEYBOARD \subseteq HCI \wedge MOUSE \subseteq HCI \wedge$
$SPEECH = \{NAME, INPUT, SEARCH, ZOOM\_IN, ZOOM\_OUT, HERE\} \wedge$
$MOUSE = \{CLICK\_ON\_NAME, CLICK\_ON\_ADDRESS, CLICK\_ON\_SEARCH, CLICK\_ON\_MAP\} \wedge$
$KEYBOARD = \{INPUT\_TEXT, ENTER\_KEY, LEFT\_ARROW, RIGHT\_ARROW, UP\_ARROW, DOWN\_ARROW\}$

VARIABLES
$modality, EV11, ...$

INVARIANT
$modality \in HCI \wedge$
$EV11 \in 0..2 \wedge ...$

INITIALISATION
$modality := NONE \parallel EV11 := 2 \parallel ...$

Fig. 21. Introduction des différentes interactions multi-modales

À ce niveau, il faut observer que tous les événements peuvent être déclenchés, c'est-à-dire, qu'il est possible de saisir le nom par la voix ou par le clavier/souris uniquement, ou bien par une combinaison des deux. On peut restreindre ces choix en fonction des propriétés CARE que l'on souhaite établir. Ce point est discuté dans la section suivante. Nous avons donc une description de toutes les possibilités.

8.3 Vérification des propriétés CARE par construction

La démarche développée dans cet article est fondée sur la notion de tâche. Nous avons représenté la totalité de l'IHM3 par des systèmes de transitions. Nous avons veillé à ne pas représenter le système de transitions explicitement, mais nous nous sommes appuyés sur la définition de tâches utilisateurs en prenant pour langage de tâches, la notation CTT. Nous avons montré que le modèle de tâches peut être représenté par un modèle B événementiel et que le modèle de tâches dirige le développement de modèles B événementiel.

Les tâches implantées en B événementiel permettent de valider des propriétés CARE grâce à la construction des modèles B événementiel. La représentation d'une interaction multi-modale satisfaisant les propriétés CARE est possible par le codage d'une tâche CTT décrivant le type de propriété que l'on souhaite représenter. Si une tâche interactive multi-modale $T0$ est réalisée par l'expression $T1\_m_i >> T2\_m_j$ avec m_i et m_j qui représentent

EVENTS
Name_address = ...

Input_Name_Field =
SELECT $EV1 = 2 \wedge Nn \neq 0 \wedge EV11 = 2$
THEN
$EV11 := EV11 - 1 \parallel$
$modality :\in \{CLICK\_ON\_NAME, NAME\}$
END;

Input_Name_Value =
SELECT $EV1 = 2 \wedge Nn \neq 0 \wedge EV11 = 1$
THEN
$EV11 := EV11 - 1 \parallel$
$modality :\in \{INPUT, INPUT\_TEXT\} \parallel$
$name :\in P(string) \parallel$
$Nn := Nn - 1$
END;

Input_Name =
SELECT $EV1 = 2 \wedge Nn \neq 0 \wedge EV11 = 0$
THEN
$EV11 := 2$
END;

Input_address = ...
Search = ...
Query = ...
Result_query = ...
END

Fig. 22. Introduction des différentes interactions multi-modales

la même modalité si $i = j$, ou deux modalités différentes si $i \neq j$. Pour la tâche $T0$ on obtient :

(1) une équivalence de modalités entre m_1 et m_2 avec l'implantation de la tâche :
–$((T1\_m_1 >> T2\_m_1)$ []
$(T1\_m_2 >> T2\_m_2))^*$;

(2) la redondance avec l'implantation de la tâche :
–$((T1\_m_1 >> T2\_m_1) \parallel$
$(T1\_m_2 >> T2\_m_2))^*$;

(3) la complémentarité avec l'implantation de la tâche :
–$((T1\_m_1 >> T2\_m_2)$ []
$(T1\_m_2 >> T2\_m_1))^*$

Dans notre étude de cas, le modèle B événementiel $IMMAPY\_ref2$ offre toutes les combinaisons d'interactions possibles. Par exemple pour coder la redondance de modalités avec la voix exclusivement, il suffit dans un nouveau raffinement d'affecter à la variable $modality$ la valeur $NAME$ dans l'événement $Input\_Name\_Field$ et la valeur $INPUT$ dans l'événement $Input\_Name\_Value$. De même pour coder la redondance avec le clavier/souris exclusivement, il suffit dans un raffinement d'affecter à la variable

$modality$ la valeur $CLICK\_ON\_NAME$ dans l'événement $Input\_Name\_Field$ et la valeur $INPUT\_TEXT$ dans l'événement $Input\_Name$- $\_Value$. Enfin pour coder la complémentarité de la voix et du clavier/souris exclusivement, il suffit, dans l'événement $Input\_Name\_Field$, d'affecter à la variable $modality$ la valeur $NAME$, et la valeur $INPUT\_$ dans l'événement $Input\_Name$- $\_Value$.

Pour illustrer ceci, les tâches faisant intervenir les modalités voix sont post-fixées par V et les tâches faisant intervenir les modalités clavier/souris sont post-fixées par CS. Ainsi pour la tâche $Input\_Name$, on obtient :

(1) une équivalence de modalités entre clavier-souris et voix avec l'implantation de la tâche :

$$-((Input\_Name\_Field\_CS >> Input\_Name\_Value\_CS)\ [] \\ (Input\_Name\_Field\_V >> Input\_Name\_Value\_V))^* ;$$

(2) la redondance avec l'implantation de la tâche :

$$-((Input\_Name\_Field\_CS >> Input\_Name\_Field\_CS)\ \| \\ (Input\_Name\_Field\_V >> Input\_Name\_Value\_V))^* ;$$

(3) la complémentarité avec l'implantation de la tâche :

$$-((Input\_Name\_Field\_V >> Input\_Name\_Field\_CS)\ [] \\ (Input\_Name\_Field\_CS >> Input\_Name\_Field\_V))^*$$

8.4 Bilan des obligations de preuve

Le tableau II illustre les résultats obtenus sur l'étude de cas. 432 obligations de preuve sont automatiquement générées. Certaines d'entre elles sont dites triviales (Obv). En effet, ces dernières sont "si évidentes" qu'elles sont validées par l'outil dès leur génération. Toutefois, toutes les obligations de preuve ne sont pas aussi simples à démontrer. L'Atelier B propose deux modes pour les valider : un mode automatique et un mode interactif. Dans notre cas les 59 obligations de preuve "non triviales" n'ont pas toutes été démontrées dans le mode automatique. Seules 3 d'entre elles ont nécessité l'utilisation d'une preuve interactive i.e. le concepteur prend en main la preuve. Dans notre cas, le concepteur n'a fourni aucun effort puisque celui-ci s'est contenté de faire appel au démonstrateur de prédicats. Ce démonstrateur, uniquement disponible dans l'environnement de preuve interactive, est néanmoins entièrement automatique. Ces 3 obligations de preuve ayant été démontrées uniquement par l'utilisation du démonstrateur de prédicats, nous pouvons donc conclure que 100% des preuves on été démontrées automatiquement.

Modèle	Obv	nOP	Auto	Interactif	%Pr
IMappy	55	10	10	0	100
IMappy_ref1	176	22	22	0	100
IMappy_ref2	142	27	24	3	100
Total	373	59	56	3	100

Table II. Résultat du nombre de preuves effectuées pour l'étude de cas

8.5 Discussion entre techniques fondées sur la preuve et vérification sur modèles

Le raffinement utilisé par B événementiel permet une modélisation par décomposition qui offre l'avantage d'introduire de nouvelles descriptions à la spécification. La totalité du système est ainsi bâtie de proche en proche. Le raffinement permet d'une part de définir les propriétés à prouver et d'autre part de conserver les propriétés déjà établies.

Notre approche se démarque des techniques de Model Checking qui procèdent par composition (SMV [McMillian 1992; Abowd et al. 1995], réseaux de Petri [Palanque et al. 1995; Navarre et al. 2005], XTL [Brun 1997], flots de données [D'Ausbourg 1998]) où les propriétés sont validées sur des systèmes complexes (parcours exhaustifs, graphes de marquage).

De plus l'utilisation d'un nombre arbitraire permet de réaliser des preuves inductives dont la complexité ne dépend pas de la valeur de ce nombre. Cette démarche évite le problème de l'explosion combinatoire que rencontrent des techniques fondées sur les parcours exhaustifs où la construction de graphes de marquage. Cependant cette approche n'est pas complètement automatique du fait de la présence possible de preuves interactives. C'est pourquoi nous préconisons l'utilisation conjointe des deux types de technologies dans une démarche intégrée.

9. CONCLUSION

Cet article a montré qu'il était possible de conduire le développement formel d'une IHM3. Des modèles événementiels formels sont décrits et raffinés de proche en proche. Chaque raffinement introduit de nouvelles informations et de nouvelles propriétés. Ce travail est fondé sur la notion de tâches avec la notation CTT ainsi que les propriétés CARE permettant de qualifier une IHM3. Nous n'avons pas proposé de nouvelle notation ni de nouvelle technique. Nous nous sommes contentés de fournir une assistance formelle à un développeur d'IHM3 souhaitant établir a priori des propriétés de l'interface en cours de conception.

Par ailleurs, cet article a montré qu'à l'image des architectures logicielles proposées pour les IHM3, il est également possible de respecter le critère de modularité dans une conception utilisant une méthode formelle. Nous avons décrit le système abstrait dans sa globalité, à un haut niveau d'abstraction (deux événements seulement), ensuite nous avons dirigé nos développements vers la multi-modalité en entrée. Cette modularité permet d'alléger le processus de développement et de preuve. La définition de schémas de représentation d'opérations de CTT permet à des non spécialistes de produire ces modèles ; néanmoins la preuve interactive restera à la charge des concepteurs.

Actuellement, nous étudions la possibilité de tester et d'animer les modèles B obtenus afin de pouvoir conduire une expérimentation à un haut niveau d'abstraction. Cela permettrait de compléter la validation des tâches par une activité d'animation [Aït-Sadoune and Aït-Ameur 2008]. Nous nous intéressons également à la modélisation de la multi-modalité en sortie [Mohand-Oussaïd et al. 2010].

RÉFÉRENCES

ABOWD, G., WANG, H.-M., AND MONK, A. 1995. A Formal Technique for Automated Dialogue Development. In *Proceedings of DIS'95*, G. Olsan and S. Schuon, Eds. 219–226.

ABRIAL, J.-R. 1996a. Extending B without changing it (for developing distributed systems). In *First B Conference, Putting Into Pratice Methods and Tools for Information System Design*, H. Habrias, Ed. Nantes, France.

ABRIAL, J.-R. 1996b. *The B Book. Assigning Programs to Meanings*. Cambridge University Press.

ABRIAL, J.-R. 2010. *Modeling in Event-B: System and Software Engineering*, Cambridge ed. Cambridge University Press.

ABRIAL, J.-R. AND HALLERSTEDE, S. 2007. Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B. *Fundamenta Informaticae 77*, 1–28.

AÏT-AMEUR, Y., AÏT-SADOUNE, I., AND BARON, M. 2006. Etude et comparaison de scénarios de développements formels d'interfaces multimodales fondés sur la preuve et le raffinement. In *6ème Conférence Francophone de Modélisation, Optimisation et SImulation des Systèmes : Défis et Opportunités (MOSIM'06)*, Lavoisier, Ed. Rabat, Maroc, 578–588.

AÏT-AMEUR, Y., AÏT-SADOUNE, I., BARON, M., AND MOTA, J.-M. 2006. Validation et vérification formelles de systèmes interactifs multimodaux fondées sur la preuve. In 18^e *Conférence Francophone sur l'Interaction Homme-Machine (IHM'2006)*. Vol. 1. ACM Press, Montréal, Canada, 123–130.

AÏT-AMEUR, Y., AÏT-SADOUNE, I., BARON, M., AND MOTA, J.-M. 2008. Développements formels d'interfaces multimodales fondés sur la preuve et le raffinement. Scénarios de développement. *RSTI série ISI Ingérnierie des Systèmes d'Information : Modélisation multiple, Formalisme et Modèles 13,* 2, 127–154.

AÏT-AMEUR, Y., BARON, M., AND GIRARD, P. 2003. Formal validation of hci user tasks. In *IEEE - The 2003 International Conference on Software Engineering Research and Practice - SERP 2003*, A.-A. Ban, A. H.R, and M. Youngsong, Eds. Vol. 2. CSREA Press, Las Vegas, Nevada USA, 732–738.

AÏT-AMEUR, Y., BARON, M., KAMEL, N., AND MOTA, J.-M. 2009. Encoding a process algebra using the Event B method. *Software Tools and Technology Transfer. Springer Verlag 11,* 3, 239–253.

AÏT-AMEUR, Y., BRÉHOLÉE, B., GIRARD, P., GUITTET, L., AND JAMBON, F. 2004. Formal verification and validation of interactive systems specifications. from informal speciticications to formal validation. In *Conference of Human Error, Safety and Systems Development, HESSD, Toulouse*.

AÏT-AMEUR, Y., GIRARD, P., AND JAMBON, F. 1998. Using the b formal approach for incremental specification design of interactive systems. In *Engineering for Human-Computer Interaction*, S. Chatty and P. Dewan, Eds. Vol. 22. Kluwer Academic Publishers, 91–108.

AÏT-SADOUNE, I. AND AÏT-AMEUR, Y. 2008. Animating Event B Models by Formal Data Models. In *ISOLA 2008 - 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*. Porto Sani, Greece.

BACK, R.-J., FAN, X., AND PREOTEASA, V. 2003. Reasoning about pointers in refinement calculus. In *Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference*. IEEE Computer Society, 425.

BACK, R.-J. AND SERE, K. 1989. Stepwise refinement of action systems. In *Proceedings of the International Conference on Mathematics of Program Construction, 375th Anniversary of the Groningen University*. Springer-Verlag, 115–138.

BARON, M. 2003. Vers une approche sure du développement des interfaces homme-machine. Ph.D. thesis, Université de Poitiers.

BARON, M., LUCQUIAUD, V., AUTARD, D., AND SCAPIN, D. 2006. K-MADe : un environement pour le noyau du modèle de description de l'activité. In *18ème Conférence Francophone sur l'Interaction Homme-Machine (IHM'2006)*, A. Press, Ed. Montréal, 287–288.

BASS, L., PELLEGRINO, R., REED, S., SHEPPARD, S., AND SZCZUR, M. 1991. The Arch Model : Seeheim Revisited. In *CHI 91 User Interface Developper's Workshop*.

BELLIK, Y. 1995. Interfaces Multimodales : concepts, modèles et architecture. Ph.D. thesis, LIMSI- Université d'Orsay.

BJORNER, D. 1987. Vdm a formal method at work. In *Proc. of VDM Europe Symposium'87*, S.-V. LNCS, Ed.

BOUCHET, J., NIGAY, L., AND GANILLE, T. 2004. Icare software components for rapidly developing multimodal interfaces. In *Proceedings of ACM-CHI 2004, Extended Abstracts*. ACM Press, Vienna, Austria, 1325–1328.

BRUN, P. 1997. *XTL: a temporal logic for the formal development of interactive systems*. Springer-Verlag, 121–139.

BUMBULIS, P., ALENCAR, P., COWAN, D., AND LUCENA, C. 1996. Validating Properties of Component-Based Graphical User Interfaces. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'96)*. Springer Verlag, 347–365.

BURBECK, S. 1992. Application programming in Smalltalk-80: How to use the Model-View-Controller (mvc). Report, University of Illinois in Urbana-Champaign (UIUC).

CAMPOS, J. AND HARRISON, M. 1997. Formally Verifying Interactive Systems: A Review. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'97)*. Springer Verlag, 109–124.

CANSELL, D. 2003. Assistance au développement incrémental et à sa preuve. Ph.D. thesis, Habilitation à diriger les recherches, Université Henri Poincaré.

COUTAZ, J. 1987. Pac, an implementation model for the user interface. In *IFIP TC13 Human-Computer Interaction (INTERACT'87)*. North-Holland, Stuttgart, 431–436.

COUTAZ, J., NIGAY, L., SALBER, D., BLANDFORD, A., MAY, J., AND YOUNG, R. 1995. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In *Proceedings of Human Computer Interaction - Interact'95*. Chapman and Hall, 115–120.

D'AUSBOURG, B. 1998. Using Model Checking for the Automatic Validation of User Interface Systems. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'98)*. Springer Verlag, 242–260.

DEPAULIS, F., JAMBON, F., GIRARD, P., AND GUITTET, L. 2006. Le modèle d'architecture logicielle H4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. *Revue d'Interaction Homme-Machine (RIHM) 7,* 1, 93–129.

DIJKSTRA, E. W. 1976. *A Discipline of Programming*. Prentice-Hall, INC.

DIX, A., FINLAY, J., ABOWD, G., AND BEALE, R. 1993. *Human-Computer Interaction*. Prentice Hall.

DUKE, D. AND HARRISON, M. 1993a. Abstract Interaction Objects. In *Proceedings of Eurographics conference and computer graphics forum*. Vol. 12. 25–36.

DUKE, D. AND HARRISON, M. 1993b. Towards a Theory of Interactors. Tech. rep., Amodeus Esprit Basic Research Project 7040, System Modelling/WP6.

DUKE, D. AND HARRISON, M. D. 1995. Event Model of Human-System Interaction. *IEEE Software Engineering Journal 10,* 1, 3–10.

FACONTI, G. P., ZANI, N., AND PATERNÒ, F. 1992. The input model of standard graphics systems revisited by formal specification. *Comput. Graph. Forum 11*, 237–251.

GRAY, P., ENGLAND, D., AND MCGOWAN, S. 1994. XUAN: Enhancing the UAN to Capture Temporal Relation Among Actions. Tech. rep., Department of Computing Science, University of Glasgow.

GUITTET, L. 1995. Contribution à l'ingénierie des interfaces homme-machine - théorie des interacteurs et architecture h4 dans le système nodaoo. Ph.D. thesis, Université de Poitiers.

HARRISON, M. AND THIMBLEBY, H. 1990. *Formal Methods in Human-Computer Interaction*. Series on Human computer interaction. Cambridge University Press.

HIX, D. AND HARTSON, H. R. 1993. *Developing User Interfaces : Ensuring Usability Through Product and Process*. Wiley Professional Computing.

HOARE, C. 1969. An Axiomatic Basis for Computer Programming. *CACM 12,* 10, 576–583.

JAMBON, F., GIRARD, P., AND AÏT-AMEUR, Y. 2001. Interactive system safety and usability enforced with the development process. In *Engineering for Human-Computer Interaction (8th IFIP International Conference, EHCI'01, Toronto, Canada, May 2001)*, R. M. Little and L. Nigay, Eds. Lecture Notes in Computer Science, vol. 2254. Springer, Canada, 39–55.

KOLSKI, C., FORBRIG, P., DAVID, B., GIRARD, P., CHI DUNG, T., AND EZZEDINE, H. 2009. Agent-based architecture for interactive system design: current approaches, perspectives, evaluation. In *Human-Computer Interaction, Part I, HCII 2009 HCI International 2009*. Number LNCS 5610. Springer-Verlag, San Diego États-Unis d'Amérique, 806–815.

LUCQUIAUD, V. 2005. Sémantique et outil pour la modélisation des tâches utilisateur : N-MDA. Ph.D. thesis, Université de Poitiers / ENSMA / INRIA.

MCMILLIAN, K. 1992. The smv system. Tech. rep., Carnegie Mellon University.

MILNER, R. 1980. *A Calculus of Communicating Systems*. Springer Verlag.

MOHAND-OUSSAÏD, L., KAMEL, N., AÏT-SADOUNE, I., AÏT-AMEUR, Y., AND AHMED-NACER, M. 2010. *Un cadre formel pour la conception et la validation de systèmes interactifs multimodaux dans le secteur du transport, In IHM et Transports terrestres*, Kolski, C ed. Hermes, Paris., à paraitre.

NAVARRE, D., PALANQUE, P., BASTIDE, R., SCHYN, A., WINCKLER, M., NEDEL, L., AND FREITAS, C. 2005. A formal description of multimodal interaction techniques for immersive virtual reality applications. In *Proceedings of INTERACT 2005*. Lecture Notes in Computer Science, Springer Verlag, Roma, Italy.

NIGAY, L., F.JAMBON, AND J.COUTAZ. 1995. Formal specification of multimodality. In *CHI'95 Workshop on Formal Specifications of User Interfaces*. ACM Press, Denver (Colorado), USA.

PALANQUE, P., BASTIDE, R., AND SENGÈS, V. 1995. Validating interactive system design through the verification of formal task and system models. In *IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction (EHCI'95)*, L. J. Bass and C. Unger, Eds. Chapman & Hall, Grand Targhee Resort (Yellowstone Park), USA, 189–212.

PATERNÒ, F. 2001. *Model-Based Design and Evaluation of Interactive Applications*. Springer.

PATERNÒ, F., MORI, G., AND GALIMBERTI, R. 2001. Ctte: An environment for analysis and development of task models of cooperative applications. In *ACM CHI 2001*. Vol. 2. ACM/SIGCHI, Seattle.

PFAFF, G. 1985. *User Interface Managment Systems*. Eurographics seminars. Springer Verlag - Berlin.

ROCHE, P. 1998. Modélisation et validation d'interface homme-machine. Ph.D. thesis, École Nationale Supérieure de l'Aéronautique et de l'Espace.

SCAPIN, D. L. AND PIERRET-GOLBREICH, C. 1989. Mad : Une méthode analytique de description des tâches. In *Colloque sur l'Ingénierie des Interfaces Homme-Machine (IHM'89)*. Sophia-Antipolis, France, 131–148.

SCAPIN, D. L. AND PIERRET-GOLBREICH, C. 1990. Towards a Method for Task Description : MAD. In *Work with display units*. Elsevier Science Publishers, North-Holland.

SPIVEY, J. M. 1992. *The Z notation: A Reference Manual*. Prentice Hall International (UK) Ltd.