Validation et Vérification Formelles de Systèmes Interactifs Multi-Modaux Fondées sur la Preuve.

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

Validation et Vérification Formelles de Systèmes Interactifs Multi-Modaux Fondées sur la Preuve

Yamine AIT-AMEUR, Idir AIT-SADOUNE, Jean-Marc MOTA

LISI/ENSMA et Université de Poitiers
BP 40109, 86961 Futuroscope cedex, France
{yamine, aitsadoune, mota}@ensma.fr

Mickael BARON

INRIA - Rocquencourt (Projet MERLIN)
BP 105, 78153 Le Chesnay cedex, France
mickael.baron@inria.fr

RÉSUME
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 l'Automatisation du Test des Interfaces Multimodales. Ce projet s'intéresse, entre autres, à la mise en œuvre d'une technique formelle fondée sur la preuve : la méthode B événementiel. 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 et propose de formaliser des notations et techniques semi-formelles issues du domaine des IHM.

MOTS CLÉS : IHM multi-modales, techniques formelles fondées sur la preuve, vérification sur modèles, propriétés CARE.

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. It focuses on the application of a formal proof based technique, namely the event B method. 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.

CATEGORIES AND SUBJECT DESCRIPTORS H.5.m [Information Interfaces and Presentation (e.g, HCI)] : Miscellaneous; H.5.2 [User Interfaces] : User-centered design; D.2.4 [Software Engineering] : Software/Program Verification - formal methods, validation

GENERAL TERMS Human Factors, Verification

KEYWORDS: Multi-modal HCI, proof based technique, verification/validation, CARE properties.

INTRODUCTION

La diversité des interfaces homme-machine 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 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. Les propriétés CARE (Concurrence, Assignation, Redondance et Equivalence) [13] permettent de caractériser ce type d'IHM du point de vue de l'utilisation des différentes modalités aussi bien entrée qu'en sortie.

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 [15]. 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 [19]) ou bien sur la preuve [3]. Dans ce contexte, on peut citer les travaux autour des ICO (Interactive Cooperative Objects) [17] qui utilisent une approche fondée sur la vérification sur modèles en proposant une utilisation des réseaux de pétri.

Nous proposons d'utiliser une technique formelle fondée sur la preuve pour la conception, la validation et la vérification de propriétés d'utilisabilité, en l'occurrence les propriétés de la famille CARE, d'IHM multi-modales (IHM3). Cette technique permet 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. Cet article décrit une partie des résultats obtenus [5] dans le cadre du projet RNRT VERBA-

TIM s'intéressant à la validation formelle par test de la multimodalité en télécommunications mobiles[1].

Dans ce projet, nous utilisons la méthode B événementiel. Nous discutons les apports de cette technique pour la conception modulaire d'IHM3, en particulier, sa capacité à exprimer et à vérifier des propriétés de la famille CARE. Notre approche utilise et propose de formaliser des notations et techniques semi-formelles issues du domaine de l'IHM.

La section suivante décrit l'approche de validation des IHM3 fondée sur la preuve. Puis nous présentons brièvement la méthode B événementiel utilisée ainsi que l'étude de cas des pages jaunes développée au CLIPS-IMAG et mise en œuvre pour illustrer notre approche. La section *conception de l'IHM3* constitue le cœur de cet article. Elle décrit notre démarche et 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 codant un modèle de tâches utilisateurs. Enfin, nous terminons par un bilan de ce travail ainsi que quelques perspectives.

VALIDATION D'IHM3 FONDÉE SUR LA PREUVE

La démarche présentée dans cet article, décrite dans [5], consiste à modéliser toute IHM3 par un système de transitions étiquetées. Chaque transition est associée à un événement qui fait passer le système, i.e. l'IHM3, d'un état à un autre. La construction de ce système de transitions est réalisée par décomposition (raffinement) des événements et des états. La démarche descendante proposée permet de définir le système de façon incrémentale en introduisant à chaque décomposition (raffinement) de nouveaux événements et de nouvelles variables d'état. Les propriétés pertinentes sont exprimées sur ces systèmes dès lors que suffisamment d'informations permettant de les exprimer sont présentes dans le raffinement obtenu. Les propriétés sont vérifiées à des niveaux abstraits sans se soucier des détails qui ne les concernent pas.

De plus il est possible de décomposer certaines parties du système en conservant abstraites les autres parties. Les modèles d'architectures tels que ARCH [8] ou MVC [10] peuvent être utilisés pour décrire cette décomposition. Par exemple, dans MVC, on peut s'intéresser à la décomposition du module C utilisant des abstractions des modules M et V. On obtient ainsi un développement modulaire et intégré. Cette approche a été mise en œuvre dans cet article où nous nous intéressons à la validation de l'interaction multimodale en entrée tout en décrivant le système interactif dans sa globalité. Les autres parties du système seront abstraites (noyau fonctionnel et interactions en sortie).

Enfin, l'utilisation d'une technique fondée sur la preuve comme la méthode B événementiel, permet d'éviter le problème de l'explosion combinatoire souvent présent en model checking. En effet, la possibilité de choisir des valeurs arbitraires et d'effectuer des preuves par induction permet d'éviter de choisir des valeurs fixées et d'explorer toutes les possibilités de manière combinatoire et exhaustive comme en model checking. Toutefois, nous ne préconisons pas d'abandonner les approches fondées sur le model checking mais préconisons une utilisation conjointe de ces deux approches [7].

[1]Site du projet : http://www.telecom.gouv.fr/rnrt/rnrt/projets/VERBATIM.htm

LA MÉTHODE B ÉVÉNEMENTIEL

Nous utilisons la méthode formelle B [2, 3] dans sa version événementielle, pour modéliser formellement un système interactif multi-modal. Plusieurs raffinements sont nécessaires avant d'aboutir au modèle représentant les différentes composantes du système interactif décrit ainsi que les propriétés pertinentes. Plusieurs scénarios de raffinement pour les systèmes interactifs ont été étudiés et comparés dans [6]. À l'image des systèmes de transitions, un modèle B événementiel décrit un système réactif par un ensemble d'événements (clause **EVENTS**) qui modifient un état (clause **VARIABLES**). Le raffinement permet de bâtir ce modèle par étapes en introduisant de nouveaux événements tout en préservant les propriétés préalablement établies (clauses **INVARIANT** et **ASSERTIONS**).

Description

Un modèle B événementiel est composé d'un ensemble d'événements atomiques décrits par des substitutions généralisées fondées sur le calcul de la plus faible pré-condition de Dijkstra [14]. Pour une substitution S et un prédicat P (post-condition), alors $[S]P$ représente la plus faible pré-condition qui établit P après l'exécution de S. Les substitutions intervenant dans les modèles B événementiel sont définies par les expressions suivantes [2, 3] :

$$[\mathbf{SKIP}]\,P \quad \Leftrightarrow \quad P \qquad (1)$$

$$[S_1 \mid\mid S_2]\,P \quad \Leftrightarrow \quad [S_1]\,P \wedge [S_2]\,P \qquad (2)$$

$$[\mathbf{ANY}\ v\ \mathbf{WHERE}\ G\ \mathbf{THEN}\ S\ \mathbf{END}]\,P \quad \Leftrightarrow \quad \forall v(G \Rightarrow [S]\,P) \qquad (3)$$

$$[\mathbf{SELECT}\ G\ \mathbf{THEN}\ S\ \mathbf{END}]\,P \quad \Leftrightarrow \quad G \Rightarrow [S]P \qquad (4)$$

$$[\mathbf{BEGIN}\ S\ \mathbf{END}]\,P \quad \Leftrightarrow \quad [S]\,P \qquad (5)$$

$$[x := E]\,P \quad \Leftrightarrow \quad P(x/E) \qquad (6)$$

$P(x/E)$ représente le prédicat P où toutes les occurrences libre de x sont remplacées par l'expression E. Les substitutions 1, 2, 5 et 6 représentent respectivement l'événement nul, la substitution parallèle exprimant que les substitutions S_1 et S_2 sont réalisées en parallèles, la substitution bloc et l'affectation. Les substitutions 3 et 4 sont les substitutions gardées où S est réalisée sous couvert de la garde G.

Chaque événement Ev est déclenché si la garde G associée à cet événement est vraie. Dans cet article les événements sont de la forme : Evt = **SELECT** G **THEN** S **END**. Evt est déclenché et est exécuté instantanément quand G est vraie. Les clauses **INVARIANTS** et **ASSERTIONS** permettent de représenter des propriétés d'atteignabilité, de sûreté et d'abscence de blocage du système prouvées pendant le développement au moyen du système de preuve associé à la méthode B. Un modèle B peut être raffiné et enrichi par de nouveaux événements

et de nouvelles propriétés. Le processus de raffinement conduit le développeur à la conception finale de l'interface utilisateur. Celle-ci est réalisée à la suite de plusieurs étapes de raffinement qui fournissent différents niveaux d'abstraction. La sémantique associée est une sémantique à base de traces d'événements (entrelacement). Un modèle est caractérisé par l'ensemble des séquences (traces) d'événements autorisés sous couvert des propriétés énoncées. La structure d'un modèle B événementiel est :

MODEL *Nom*	- Déclaration du modèle
...	
VARIABLES	- Décl. de l'ensemble
Var	des variables du système.
INVARIANT	- Décl. de propriétés invariants
I(*Var*)	(sûreté) pour tout événements
ASSERTIONS	- Décl. de propriétés résultant de
A(*Var*)	l'invariant (ex. pas de blocage)
INITIALISATION	- Init. des variables du système
Var := ...	
EVENTS	- Décl. des événements du système
Evt_Name =	- Événement déclenché si G= TRUE
SELECT	
G	- Garde de l'événement
THEN	
S	- Action ou corps de l'événement
END;	
...	
END.	

Un exemple

Considérons ci-dessous, la spécification d'une horloge [11]. *Clock* décrit une variable h pour les heures. L'événement *incr* permet d'augmenter la variable *hour*. L'événement *zero* est déclenché quand $h = 23$ pour initialiser à nouveau la variable *hour*.

MODEL *Clock*
VARIABLES
h
INVARIANT
$h \in 0..23$
ASSERTIONS
$h < 100$
INITIALISATION
$h := 13$
EVENTS
incr = **SELECT** $h \neq 23$ **THEN** $h := h + 1$ **END;**
zero = **SELECT** $h = 23$ **THEN** $h := 0$ **END**
END

Le raffinement *ClockWMinute* introduit une nouvelle variable m et un nouvel événement *ticTac* pour les minutes. Les gardes des événements *incr* et *zero* tiennent compte désormais des minutes. La clause **ASSERTIONS** assure que les nouveaux événements du système sont correctement déclenchés et n'engendrent pas de blocage.

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

Notons que des règles de génération à partir du système de transition permettent de produire le code associé à ce système [4].

Etude de cas

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 ICARE (Interaction-Care) [9].

Elle 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.

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 champ puis prononciation du nom ou alors en choisissant oralement le champ 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 parallèle appuie sur la touche du clavier correspondante, une seule commande d'agrandissement sera alors transmise au système.

TECHNIQUE DE MODÉLISATION DES IHM3

La grande majorité des modèles d'architecture de systèmes interactifs (PAC [12], ARCH [8]) distingue essentiellement trois composants essentiels :

- le *noyau fonctionnel (NF)* représente l'ensemble des fonctions et services offerts par le système avec lequel l'utilisateur interagit. Dans le cas précis de l'étude de cas il peut s'agir d'effectuer la recherche d'un nom et d'une localisation dans une base de données ;
- la *présentation (P)* est le composant disposant des différentes fonctions permettant de gérer la saisie et l'affichage d'informations consommées (le nom de la personne à rechercher) ou produites par le *NF*, ou bien de gérer les informations de présentation ne nécessitant pas de liaison avec le noyau fonctionnel (ex. déplacer une fenêtre) ;
- le *contrôleur de dialogue (CD)* organise la communication entre les deux composants précédents. Il **synchronise** les flots d'échanges entre le noyau fonctionnel et la présentation. Ce composant est essentiel dans un système interactif. Le contrôleur de dialogue résulte de la composition synchronisée des deux composants que sont le noyau et la 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**. Enfin, ces systèmes sont construits par une suite de raffinements qui préservent les propriétés. Le contrôleur de dialogue représente le produit synchronisé des deux systèmes correspondant respectivement au *NF* et la *P*. L'état du composant obtenu est une paire $Var = (Var_{NF}, Var_P)$ composée des variables du *NF* et de la *P*. En général, les événements apparaissant dans ce type de modèles B sont de la forme :

$Evt =$ **SELECT**
$G_{NF} \wedge G_P$
THEN
$S_{NF} \parallel S_P$
END;

où G_{NF} et S_{NF} correspondent respectivement à la garde et à la substitution d'un sous-événement correspondant à une action issue du *NF*, alors que, G_P et S_P correspondent à la garde et à la substitution d'un sous événement issu de la *P*.

Conception de l'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.

Dans le cadre du projet VERBATIM, nous avons identifié quatre scénarios de conception d'IHM3 fondée sur la preuve. Quatre scénarios sont détaillés dans [6]. Chacun décrit une implantation de la composition des composants noyau fonctionnel et de la présentation pour définir le *CD* :

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.

À partir de cette étude, nous avons retenu le scénario 4. En effet, vu que nous sommes intéressés par la validation d'une IHM3, il ne nous est pas apparu pertinent de vérifier et de valider également le *NF*. Nous avons utilisé une abstraction de celui-ci permettant une vérification formelle modulaire.

Modélisation de "pages jaunes CLIPS" par raffinement

Quatre modèles ont été définis suivant le scénario 4 et chacun raffine le précédent. Le premier introduit les événements de haut niveau. Le deuxième présente les événements de la présentation ainsi que leurs synchronisations avec le *CD*. Le troisième introduit les interactions multi-modales avec les différentes possibilités d'interaction. Enfin, le quatrième introduit concrètement les modalités et décompose les interactions multi-modales en événements d'interaction atomiques de base. Ces modèles font tous abstraction du noyau fonctionnel et ne font que rendre compte du fait que des événements abstraits du noyau fonctionnel ont été déclenchés.

Dans la suite, nous décrivons une représentation simplifiée du développement de l'IHM3 associée à l'étude de cas "Pages Jaunes CLIPS". Nous avons volontairement réduit les modèles B événementiel afin de respecter les limites de taille de cet article. Le code complet peut être demandé directement aux auteurs.

Le modèle racine. Le modèle racine est un modèle abstrait représentant la synchronisation entre les événements de saisie et de déclenchement du noyau fonctionnel.

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
$Query\_sending, Query\_ready := 0, 1$
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

Deux événements abstraits (sans modalités) *Query* et *Result_query* décrivent ce modèle. Le premier indique ($Query\_sending := 1$) que la requête est prête à être envoyée au noyau fonctionnel 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.

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 et qui déclenchent les événements du noyau fonctionnel.

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). Ainsi la tâche CTT (Concur-

TaskTrees) [18] qui décrit les différentes possibilités de saisie est définie par :

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

où * désigne l'itération et $>>$ désigne l'opérateur de séquence. La tâche $Query$ est implantée dans le raffinement $IMAPPY\_Ref\_1$ décrit ci-dessous. 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
$EV1 \in 0..3 \wedge Nn \in \mathbb{N} \wedge Na \in \mathbb{N} \wedge$
$map \in BOOL \wedge name \subseteq string \wedge address \subseteq string \wedge$
$(Query\_ready = 1 \Leftrightarrow EV1 \neq 0)$
ASSERTIONS ...
INITIALISATION
$Query\_sending := 0 \parallel map, EV1 := FALSE, 3 \parallel$
$name, address, Nn, Na := \emptyset, \emptyset, 1, 1$

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 récupérent respectivement le nom et l'adresse. Les variables entières Nn et Na désignent les nombres arbitraires d'itérations * de CTT. Elles sont initialisées, dans l'événement $Name\_Address$, grâce à l'opérateur $:\in$ qui considère un élément quelconque dans un ensemble et indiquent le nombre de fois que cet événement est déclenché. Enfin, la variable $EV1$ (variable entière décroissante) initialisée à 3 (nombre d'événement à déclencher) est un variant qui décrit l'ordre de déclenchement de chaque événement de l'abstraction.

EVENTS
$Name\_Address =$ **SELECT**
$EV1 = 3$
THEN
$EV1 := EV1 - 1 \parallel Nn :\in \mathbb{N} \parallel Na :\in \mathbb{N}$
END ;

$Input\_Name =$ **SELECT**
$EV1 = 2 \wedge Nn \neq 0$
THEN
$Nn := Nn - 1 \parallel name :\in \mathbb{P}(string)$
END ;

$Input\_Address =$ **SELECT**
$EV1 = 2 \wedge Na \neq 0$
THEN
$Na := Na - 1 \parallel address :\in \mathbb{P}(string)$
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;

$Result\_query = \ldots$

END

Les événements ci-dessus décrivent la tâche CTT précédente. L'événement $Name\_Address$ initialise itérateurs Nn et Na alors que les trois événements $Input\_Name$, $Input\_Address$ 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 abordées dans les prochains raffinements.

Comme nous l'avions déclaré précédemment, nous ne nous intéressons pas à la multi-modalité 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 plus simples du fait que certains événements restent abstraits.

Identification des modalités. L'étape suivante consiste à identifier et à introduire les différentes interactions multi-modales en entrée mises en jeu. Une nouvelle variable booléenne *Lock* indique que le focus est sur la saisie du nom.

REFINEMENT $IMAPPY\_ref2$
REFINES $IMAPPY\_ref1$
VARIABLES
$lock, \ldots$
INVARIANT
$lock \in BOOL \wedge \ldots$
ASSERTIONS
...
INITIALISATION
$lock := TRUE \parallel \ldots$

Nous procédons à la décomposition (raffinement) des événements $Input\_Name$ et $Input\_Address$ faisant intervenir les modalités voix (V) et clavier/souris (CS) sans introduire explicitement les modalités, ainsi que leurs caractéristiques. Ceci est un autre aspect de la modularité puisqu'il est possible de valider l'entrelacement des événements sans s'intéresser à ce qu'ils font explicitement. Pour cela, il est nécessaire d'enrichir le modèle B par la déclaration de nouveaux événements.

Dans la suite, seuls les événements raffinant l'événement $Input\_Name$ sont présentés selon la tâche de décomposition suivante :

$$Input\_name^* = (Input\_Name\_V\_V[]Input\_Name\_V\_CS[] Input\_Name\_CS\_V[]Input\_Name\_CS\_CS)^*$$

où [] désigne l'opérateur de choix. 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. Cela donne quatre possibilités. Ici on peut saisir le nom par la V puis V, ou bien par V puis CS ou bien par CS puis V ou bien par CS puis CS. Tous ces événements peuvent être itérés un nombre arbitraire de fois.

EVENTS
$Name\_Address = \ldots$

$Input\_Name\_V\_V$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE$
THEN
$Nn := Nn - 1 \parallel name :\in \mathbb{P}(string) \parallel lock := FALSE$
END;

$Input\_Name\_V\_CS$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE$
THEN
$Nn := Nn - 1 \parallel name :\in \mathbb{P}(string) \ \parallel$
$lock := FALSE$
END;

$Input\_Name\_CS\_CS$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE$
THEN
$Nn := Nn - 1 \parallel name :\in \mathbb{P}(string) \ \parallel$
$lock := FALSE$
END;

$Input\_Name\_CS\_V$ = **SELECT**
$EV1 = 2 \wedge Nn \neq 0 \wedge lock = TRUE$
THEN
$Nn := Nn - 1 \parallel name :\in \mathbb{P}(string) \ \parallel$
$lock := FALSE$
END;

$Input\_Name$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = FALSE$
THEN
$lock := TRUE$
END ;

$Input\_Address = \ldots$

$Search = \ldots$

$Query = \ldots$

$Result\_query = \ldots$
END

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.

Identification des interactions concrètes. Le dernier raffinement présenté introduit explicitement les modalités utilisées ainsi que leurs valeurs (au sens de la modélisation B). 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 effectuée, ni le processus de preuve.

REFINEMENT $IMAPPY\_ref3$
REFINES $IMAPPY\_ref2$
SETS
$HCI = \{NONE, NAME, INPUT, SEARCH,$
$ZOOM\_IN, INPUT\_TEXT, ENTER\_KEY,$
$LEFT\_ARROW, CLICK\_NAME,$
$CLICK\_SEARCH, \ \ldots\}$
CONSTANTS
$SPEECH, KEYBOARD, MOUSE$
PROPERTIES
$SPEECH \cup KEYBOARD \cup MOUSE \subseteq HCI \ \wedge$
$SPEECH = \{NAME, INPUT, SEARCH, ...\}$
$MOUSE = \{CLICK\_SEARCH, CLICK\_NAME, ...\}$
$KEYBOARD = \{INPUT\_TEXT, ENTER\_KEY, ...\}$
VARIABLES
$Query\_sending, map, EV1,$
$name, adress, Nn, Na,$
$modality, lock,$
$V\_speech\_speech, V\_speech\_keyboard,$
$V\_mouse\_speech, V\_mouse\_keyboard$

INVARIANT
$name \subseteq string \ \wedge$
$V\_speech\_speech \in 0..3 \ \wedge$
$V\_mouse\_speech \in 0..3 \ \wedge$
$V\_mouse\_keyboard \in 0..3 \ \wedge$
$V\_speech\_keyboard \in 0..3 \ \wedge$
$modality \in HCI \wedge \ldots$
ASSERTIONS
...
INITIALISATION
$Query\_sending := 0 \parallel map := FALSE \parallel EV1 := 3 \parallel$
$name, adress := \emptyset, \emptyset \parallel Nn, Na := 1, 1 \ \parallel$
$modality := NONE \parallel lock := TRUE \ \parallel$
$V\_speech\_speech, V\_speech\_keyboard := 2, 2 \parallel$
$V\_mouse\_speech, V\_mouse\_keyboard := 2, 2$

Les variables V_x_y, où x et y sont des modalités, définissent le type d'interaction en cours de déclenchement dans le modèle. La variable *modality* décrit la modalité activée à un instant donné. Elle est initialisée à $NONE$.

Les événements du raffinement précédent sont décomposés (par raffinement) en :

$$Input\_Name\_V\_V = Input\_Name\_V\_V\_field >> Input\_Name\_V\_V\_value$$

$$Input\_Name\_V\_CS = Input\_Name\_V\_CS\_field >> Input\_Name\_V\_CS\_value$$

$$Input\_Name\_CS\_V = Input\_Name\_CS\_V\_field >> Input\_Name\_CS\_V\_value$$

$$Input\_Name\_CS\_CS = Input\_Name\_CS\_CS\_field >> Input\_Name\_CS\_CS\_value$$

Les suffixes $field$ et $value$ indiquent respectivement les événements associés au choix du champ à remplir et à la valeur entrée pour ce champ. On obtient :

EVENTS
$Name\_Address$ = **SELECT**
$EV1 = 3$
THEN
...
END ;

$Input\_Name\_V\_V\_field$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE \ \wedge$
$V\_speech\_speech = 2$
THEN
$V\_speech\_speech := V\_speech\_speech - 1 \parallel$
$modality := NAME$
END;

$Input\_Name\_V\_V\_value$ = **SELECT**
$EV1 = 2 \wedge Nn \neq 0 \wedge lock = TRUE \ \wedge$
$V\_speech\_speech = 1$
THEN
$V\_speech\_speech := V\_speech\_speech - 1 \parallel$
$name :\in \mathbb{P}(string) \ \parallel$
$modality := INPUT$
END;

$Input\_Name\_V\_V$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE \ \wedge$
$V\_speech\_speech = 0$
THEN
$Nn := Nn - 1 \parallel lock := FALSE$
END;

$Input\_Name\_V\_CS\_field$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE \ \wedge$
$V\_speech\_keyboard = 2$
THEN
$V\_speech\_keyboard := V\_speech\_keyboard - 1 \parallel$
$modality := NAME$
END;

$Input\_Name\_V\_CS\_value$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE \wedge$
$V\_speech\_keyboard = 1$
THEN
$V\_speech\_keyboard := V\_speech\_keyboard - 1 \parallel$
$name :\in \mathbb{P}(string) \parallel$
$modality := INPUT\_TEXT$
END;

$Input\_Name\_V\_CS$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = TRUE \wedge$
$V\_speech\_keyboard = 0$
THEN
$Nn := Nn - 1 \parallel lock := FALSE$
END;

$Input\_Name\_CS\_V\_field = \ldots$

$Input\_Name\_CS\_V\_value = \ldots$

$Input\_Name\_CS\_V = \ldots$

$Input\_Name\_CS\_CS\_field = \ldots$

$Input\_Name\_CS\_CS\_value = \ldots$

$Input\_Name\_CS\_CS = \ldots$

$Input\_Name$ = **SELECT**
$EV1 = 2 \ \wedge Nn \neq 0 \ \wedge lock = FALSE$
THEN
$lock = TRUE \parallel modality = NONE$
END;

$Input\_Adress = \ldots$

$Search$ = **SELECT**
$EV1 = 2 \ \wedge Nn = 0 \ \wedge Na = 0$
THEN
$EV1 := 1 \parallel$
$modality :\in \{CLICK\_SEARCH, ENTER\_KEY, SEARCH\}$
END ;

$Query = \ldots$

$Result\_query = \ldots$
END

Vérification de propriétés CARE par construction

Le modèle B événementiel $IMAPPY\_ref2$ offrait toutes les combinaisons d'interactions possibles. Il permettait ainsi le codage de la équivalence de modalités avec la voix exclusivement (événement $Input\_Name\_V\_V$) ou bien avec le clavier/souris exclusivement (événement $Input\_Name\_CS\_CS$). Mais, ce modèle permettait également le codage de la complémentarité de la voix et du clavier/souris avec les événements ($Input\_Name\_V\_CS$ et $Input\_Name\_CS\_V$).

La représentation d'une interaction multi-modale satisfaisant d'autres types de propriétés CARE est possible par le codage d'une autre tâche CTT. Cette tâche doit décrire le type de propriété que l'on souhaite représenter. Par exemple,

- une équivalence de modalités entre clavier/souris et voix impliquerait l'implantation de la tâche :

$$(Input\_Name\_CS\_CS[]\ Input\_Name\_V\_V)^* || \ldots$$

- la redondance impliquerait l'implantation de la tâche :

$$(Input\_Name\_V\_V \ ||\ \ Input\_Name\_CS\_CS)^* || \ldots$$

- la complémentarité impliquerait l'implantation de la tâche :

$$Input\_Name\_V\_CS \ []\ Input\_Name\_CS\_V$$

Le raisonnement effectué ici s'applique également au dernier raffinement $IMAPPY\_ref3$. En effet, nous avons choisi d'identifier le champ puis de saisir sa valeur selon une tâche de la forme :

$$Input\_Name\_x\_y = Input\_Name\_x\_y\_field >> Input\_Name\_x\_y\_value$$

D'autres choix auraient pu être décrits, comme :

$$Input\_Name\_x\_y = Input\_Name\_x\_y\_field \parallel Input\_Name\_x\_y\_value$$

pour décrire entre le choix du champ et la saisie.

Validation de tâches utilisateurs

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 et que le modèle de tâches dirige la définition des modèles B. Enfin, il faut noter que les tâches implantées en B événementiel permettent de valider des propriétés CARE grâce à la construction des modèles B événementiel.

Bilan : preuves de propriétés

Le tableau ci-dessous illustre les résultats obtenus sur l'étude de cas. 219 obligations de preuve sont automatiquement générées. Seules deux d'entre elles ne sont pas prouvées automatiquement. La première dans le raffinement $IMAPPY\_ref2$ consiste à prouver un lemme. La seconde dans $IMAPPY\_ref3$, est une preuve par cas liée à la disjonction des gardes des événements (pour prouver l'absence de blocage). Elle consiste à prouver les différents cas obtenus dans cette disjonction en appelant le prouveur qui les établit.

Modèle	Obv	nOP	Auto	Interactif	%Pr
$IMAPPY$	75	10	10	0	100
$IMAPPY\_ref1$	360	22	22	0	100
$IMAPPY\_ref2$	465	28	27	1	100
$IMAPPY\_ref3$	557	159	158	1	100
Total	145	219	217	2	100

Bilan : discussion entre techniques fondées sur la preuve et vérification sur modèles

Le raffinement utilisé par le B événementiel permet une modélisation par décomposition ce 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 [16], réseaux de petri, promela/spin [1]) et 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.

CONCLUSION ET PERSPECTIVES

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 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 de test. Nous nous intéressons également à la prise en compte des modalités en sortie.

REFERENCES

1. Spin - version 4.0.1, 1993.
2. J.-R. Abrial. Extending B without changing it (for Developing Distributed Systems). In H. Habrias, (Ed.), *First B Conference, Putting Into Pratice Methods and Tools for Information System Design*, page 21, Nantes, France, 1996.
3. J.-R. Abrial. *The B Book. Assigning Programs to Meanings*. Cambridge University Press, 1996.
4. J.-R. Abrial. Event Based Sequential Program Development: Application to Constructing a Pointer Program. In *FME 2003: Formal Methods Europe*, pages 51–74. Springer, September 2003.
5. Y. Aït-Ameur, I. Aït-Sadoune, and M. Baron. Modélisation et validation formelles d'IHM : Lot 1. Technical report, Projet RNRT Verbatim, LISI ENSMA, 2005.
6. Y. Aït-Ameur, I. Aït-Sadoune, and M. Baron. Etude et comparaison de scénarios de développements formels d'interfaces multimodales fondés sur la preuve et le raffinement. In *Proceedings of MOSIM 2006 - 6ème Conférence Francophone de Modélisation et Simulation*, Rabat, Maroc, 2006. To Appear.
7. Y. Aït-Ameur and N. Kamel. A generic formal specification of fusion of modalities in a multimodal HCI. In R. Jacquart, (Ed.), *IFIP World Computer Science*, pages 415–420, Toulouse, France, 2004. Kluwer Academic Publishers.
8. L. Bass, R. Faneuf, R. Little, N. Mayer, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard, and M. Szczur. A metamodel for the runtime architecture of an interactive system. *SIGCHI Bulletin*, 24(1):32–37, 1992.
9. J. Bouchet, L. Nigay, and T. Ganille. ICARE software components for rapidly developing multimodal interfaces. In *Proceedings of ACM-CHI 2004, Extended Abstracts*, pages 1325–1328, Vienna, Austria, 2004.
10. S. Burbeck. Application programming in smalltalk-80: How to use the Model-View-Controller (MVC). Report, 1992.
11. D. Cansell. *Assistance au développement incrémental et à sa preuve*. Habilitation à diriger les recherches, Université Henri Poincaré, 2003.
12. J. Coutaz. PAC, an Implementation Model for the User Interface. In *IFIP TC13 Human-Computer Interaction (INTERACT'87)*, pages 431–436, Stuttgart, 1987. North-Holland.
13. J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R.M. Young. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In *Proceedings of Human Computer Interaction - Interact'95*, pages 115–120. Chapman and Hall, 1995.
14. E.W. Dijkstra. A discipline of programming. Prentice-Hall, Englewood Cliffs, 1976.
15. F. Jambon, Philippe Brun, and Y. Aït-Ameur. *Spécifications des systèmes interactifs (Volume 1, chapitre 6)*, pages 175–206. In Kolski C. (Ed.), Interaction homme-machine pour les S.I. Hermès Science, Paris, France, 2001.
16. K. Mc Millan. The SMV System. Technical report, Carnegie Mellon University, 1992.
17. D. Navarre, P. Palanque, R. Bastide, A. Schyn, M. Winckler, L. Nedel, and C. Freitas. A formal description of multimodal interaction techniques for immersive virtual reality applications. In *Proceedings of INTERACT 2005*, Roma, Italy, 2005. Lecture Notes in Computer Science, Springer Verlag.
18. F. Paternò. *Model-Based Design and Evaluation of Interactive Applications*. Springer, 2001.
19. P. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie, and A. Petit. *Vérification de logiciels : techniques et outils du model-checking*. Vuibert, April 1999.