6e Conférence Francophone de MOdélisation et SIMulation - MOSIM'06 - du 3 au 5 avril 2006 - Rabat - Maroc
«Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités»

# Etude et comparaison de scénarios de développements formels d'interfaces multi-modales fondés sur la preuve et le raffinement

**Yamine AIT AMEUR, Idir AIT SADOUNE**
LISI/ENSMA et Université de Poitiers
BP 40109, 86961 Futuroscope Cedex, France
{yamine, idir.aitsadoune}@ensma.fr

**Mickaël BARON**
INRIA - Rocquencourt (Projet MERLIN)
BP 105, 78153 Le Chesnay Cedex, France
{mickael.baron}@inria.fr

**RÉSUMÉ.** *Les architectures d'un système interactif reposent sur la séparation du noyau fonctionnel de l'interface utilisateur du logiciel. Le développement de ces deux modules implique l'utilisation de techniques et d'approches différentes. La validation du système interactif peut être une étape complexe puisque ces modules sont développés séparément.*
*Dans le cadre du projet RNRT Verbatim*, l'étude de différents scénarios de développement formels des systèmes interactifs multi-modaux, a été menée en utilisant la méthode B dans sa version "B événementiel". Cet article présente une partie des résultats de cette étude. Le raffinement est mis en œuvre pour structurer les développements et la preuve pour établir les propriétés. Il s'intéresse principalement à la liaison (lors de leur composition) entre les deux modules d'un système interactif que sont le noyau fonctionnel et l'interface utilisateur. Quatre scénarios de développement différents, représentant formellement cette liaison, sont étudiés et comparés. Cette comparaison est réalisée sur la base du nombre d'obligations de preuve générées et relatives aux propriétés décrites dans les spécifications. Une étude de cas décrivant un système interactif multi-modal, illustrant ces scénarios et leur comparaison est utilisée tout au long de cet article.*

**MOTS-CLÉS :** *Méthode "B événementiel", raffinements d'événements, IHM (Interfaces Homme-Machine), architectures logicielles, systèmes interactifs, méthodologie.*

## 1 INTRODUCTION

Les progrès réalisés par la technologie ont ouvert la voie à l'apparition de nouveaux types d'interfaces telles que les interfaces de type WIMP (Windows, Icônes, Menus, Pointing devices) ou les IHM multi-modales. Cette diversité et ces progrès 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 se sont intéressés à ce domaine. Toutefois leur application aux IHM multi-modales reste rare. 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 (Concurrence, Assignation, Redondance et Equivalence) que le système doit assurer. Reste aussi que dans la plupart des travaux, les besoins exprimant des principes d'utilisabilité sont souvent négligés dans la phase de conception. Dans cet article, nous proposons une démarche fondée sur l'utilisation de la technique formelle B pour la conception. La stratégie proposée comprend quatre scénarios prenant en compte la séparation du noyau fonctionnel de l'application et de son interface. Cette séparation est présente dans la plupart des architectures logicielles proposées pour le développement d'IHM telles que ARCH(Bass, Hardy, Hoyt, Little & Seacord 1988), MVC(Burbeck 1992) ou PAC(Coutaz 1987).

Les travaux réalisés dans le cadre du projet RNRT Verbatim visent à augmenter la part de l'utilisateur dans le processus de développement d'une IHM. La plupart des modèles de description de tâches utilisateurs existants sont informels ou semi-formels. Nous nous fondons sur les travaux développés au LISI (Aït-Ameur, Girard & Jambon 1998, Jambon, Girard & Aït-Ameur 2001, Aït-Ameur, Baron & Girard 2003, Baron 2003, Aït-Ameur, Bréholée, Girard, Guittet & Jambon 2004, Aït-Ameur, Baron & Nadjet 2005, Aït-Ameur & Baron 2005) afin de permettre l'exploitation de la formulation des différentes notations semi-formelles en modèles formels.

*Réseau National de Recherche en Télécommunications. VERification Bi-formelle d'Interfaces Multi-modales.

Notre article est structuré de la manière suivante. La section 2 est une revue des approches formelles utilisées dans le domaine des IHM. Nous introduisons ensuite la méthode B événementiel en section 3 avec un exemple illustratif de l'utilisation de cette dernière. En section 4 nous présentons les quatre scénarios de développement avec une mise en pratique sur l'étude de cas MATIS en section 5.

## 2 APPROCHES FORMELLES DANS LE DOMAINE DE L'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 de preuves. Les deux techniques ont été appliquées au domaine de l'IHM (Campos & Harrison 1997).

La première catégorie est basée sur l'évaluation de propriétés logiques d'un système à transition obtenue à partir des valeurs des variables. Parmi ces techniques, nous trouvons les logiques temporelles, les réseaux de Petri et bien d'autres encore. Dans le domaine de l'IHM, ces techniques ont été utilisées la première fois pour la vérification formelle de propriétés de systèmes interactifs (Campos & Harrison 1997). Par exemple, (Abowd, Wang & Monk 1995) vérifie des interfaces utilisateur avec SMV (Symbolic Model Verifier) utilisant CTL (Computational Tree Logic), tandis que (Paterno & Faconti 1992) utilise LOTOS pour écrire des spécifications d'interacteur. (Brun 1997) propose une logique temporelle, nommé XTL (eXtended Temporal Logic). La vérification sur modèles (model checking) est également utilisée par (Palanque, Bastide & Sengès 1995) où l'utilisateur et le système sont modélisés au moyen de réseaux de Petri orientés objets appelés ICO. Finalement, (D'Ausbourg 1998) a utilisé un langage à flot de donnés pour la validation automatique de systèmes interactifs.

La seconde catégorie est basée sur des systèmes de preuves où le modèle est décrit par des variables, des opérations, des événements, des propriétés temporelles, des variants et des invariants. Les opérations doivent préserver ces invariants et une famille de propriétés (telles que sûreté, non-blocage, vivacité, etc). Pour assurer la conformité de ces spécifications, des obligations de preuve sont générées et doivent être prouvées. Notons que toutes ces techniques ne permettent la préservation de toutes les propriétés. Différents travaux connexes se sont intéressés à la préservation des propriétés dans les raffinements. Parmi ces techniques, nous trouvons Z(Spivey 1988), basée sur la théorie des ensembles, VDM(Bjorner 1987), basée sur le calcul de pré/postconditions (Hoare 1969), et B(Abrial 1996a, Abrial 1996b), basée sur le calcul de la plus faible précondition. Dans le domaine de l'IHM, VDM et Z ont été utilisées pour la définition de structures atomiques comme les interacteurs (Duke & Harrison 1993b, Duke & Harrison 1993a).

## 3 LA METHODE B EVENEMENTIEL

Nous utilisons la méthode formelle B pour montrer qu'il est possible de traiter un développement formel complet d'un système interactif. Cette approche permet la spécification, la vérification et le raffinement de spécifications formelles B. Plus particulièrement, des programmes écrits en ADA et TCL-TK ont été générés au moyen de cette technique, démontrant ainsi que notre approche peut être utilisée à tous 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).

### 3.1 Approche B

Nous utilisons la méthode B (Abrial 1996b, Abrial 1996a) dans sa définition événementielle. Elle permet la description de systèmes distribués, parallèles, multi-modales, réactifs et interactifs.

Un modèle B événementiel est composé d'un ensemble d'événements atomiques décrits par des substitutions généralisées particulières. Les substitutions généralisées sont basées sur le calcul de la plus faible précondition de Dijkstra(Dijkstra 1976). Si nous considérons une substitution $S$ et un prédicat $P$ représentant une post-condition, alors $[S]P$ représente la plus faible précondition qui établit $P$ après l'exécution de $S$. Les substitutions intervenant dans les modèles B événementiel sont définies par les expressions suivantes (Abrial 1996b, Abrial 1996a, Lano 1996) :

$$[\mathbf{SKIP}]\,P \quad \Leftrightarrow \quad P \tag{1}$$
$$[\mathbf{S1}\ ||\ \mathbf{S2}]\,P \quad \Leftrightarrow \quad [S1]\,P \wedge [S2]\,P \tag{2}$$
$$[\mathbf{ANY}v\mathbf{WHERE}E\mathbf{THEN}S\mathbf{END}]\,P \quad \Leftrightarrow \quad \forall\, v(P \Longrightarrow [S]\,P) \tag{3}$$
$$[\mathbf{SELECT}E\mathbf{THEN}S\mathbf{END}]\,P \quad \Leftrightarrow \quad E \Longrightarrow [S]P \tag{4}$$
$$[\mathbf{BEGIN}S\mathbf{END}]\,P \quad \Leftrightarrow \quad [S]\,P \tag{5}$$
$$[\mathbf{x{:=}E}]\,P \quad \Leftrightarrow \quad P(x/E) \tag{6}$$

$P(x/E)$ 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 $S1$ et $S2$ sont réalisées en parallèles, la substitution bloc et l'affectation. Les substitutions 3 and 4 sont les substitutions gardées où $S$ est réalisée sous couvert de la garde $E$.

Chaque événement $Ev$ est déclenché si la garde $P$ associée à cet événement est vraie. Dans cet article, seule la substitution $SELECT$ sera utilisée. $Evt = SELECT\ P\ THEN\ S\ END$. L'événement $Ev$ est déclenché et est exécuté en un temps nul quand $P$ est vraie. Par ailleurs, un modèle B contient un ensemble de propriétés, des invariants, des propriétés d'atteignabilité, de sûreté et de vivacité qui peuvent être prouvées pendant le développement au moyen du système de preuves 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, 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. Un modèle est caractérisé par l'ensemble des séquences (traces) d'événements autorisés sous couvert des propriétés énoncées.

### 3.2 Un exemple de modèle B événementiel

Considérons ci-dessous, la spécification d'une horloge présentée dans (Cansell 2003). 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 *hour*. Le second événement est *zero*. Il est déclenché quand $h = 23$ pour re-initialiser 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**

Dans la spécification du raffinement $ClockWMinute$ nous introduisons une nouvelle variable $m$ et un nouvel événement $ticTac$. Nous enrichissons les gardes des événements *incr* et *zero* en introduisant la description des minutes. La clause **ASSERTIONS** permet d'assurer que les nouveux événements du système peuvent être correctement déclenchés.

**REFINEMENT**
$ClockWMinute$
**REFINES**
$Clock$
**VARIABLES**
$h, m$
**INVARIANT**
$m \in 0..59$
**ASSERTIONS**
$(h \neq 23) \vee (h = 23) \Rightarrow (h \neq 23 \wedge m = 59) \vee$
$(h = 23 \wedge m = 59) \vee (m \neq 59)$
**VARIANT**
$59 - m$
**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.**

Une condition de non blocage est exprimée dans la clause **ASSERTIONS**.

## 4 LA METHODE B EVENEMENTIEL ET LE DEVELOPPEMENT DE SYSTEMES INTERACTIFS

Nous étudions le cas particulier de la conception de systèmes interactifs et des interfaces utilisateur. Après une rapide présentation des principes des modèles d'architectures développés dans le domaine des interfaces utilisateur nous montrons la mise en œuvre de la méthode B événementiel dans ce cas.

### 4.1 Développement de systèmes interactifs

La grande majorité des modèles d'architecture de systèmes interactifs, tels que ARCH(Bass et al. 1988), MVC(Burbeck 1992) ou PAC(Coutaz 1987), distinguent 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.

- le *noyau fonctionnel* représente l'ensemble des fonctions et services offerts par le système avec lequel l'utilisateur interagit ;
- la *présentation* est le composant disposant des différentes fonctions permettant de gérer la saisie et l'affichage d'informations consommées ou produites par le noyau fonctionnel, ou bien de gérer les informations de présentation ne nécessitant pas de liaison avec le noyau fonctionnel (ex. déplacer un fenêtre) ;

le *contrôleur de dialogue* organise la communication entre les deux composants précédents. Il **synchronise** les flots d'échanges entre noyau fonctionnel 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 fonctionnel et la présentation. Cette opération n'est pas disponible explicitement dans la méthode B événementiel.

Nous nous intéressons dans cet article à la modélisation et à la vérification formelles du composant contrôleur de dialogue. Plusieurs scénarios de développement de ce composants sont possibles.

### 4.2 Description de la modélisation

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.

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

La construction de ces modèles dépend du scénario de développement adopté. Nous abordons cet aspect dans la section suivante.

Le contrôleur de dialogue est représenté par le produit synchronisé des deux systèmes qui représentent 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 la présentation et les événements sont de la forme :

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

où $G_{NF}$ et $S_{NF}$ correspondent respectivement à la garde et à la substitution de la partie noyau fonctionnel. A 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 $S_{NF}$ ou $S_{IHM}$ valent $Skip$.

La preuve des modèles B obtenus dépend de la façon de construire ce produit. Cette construction exploite le raffinement.

Dans la suite nous montrons quatre scénarios de développement différents qui exploitent le raffinement supporté par la méthode B événementiel.

## 5 Modélisation de la composition : quatre scénarios de développement

Cette section présente quatre scénarios différents de formalisation de l'opération de composition des parties noyau fonctionnel et interface utilisateur d'un système interactif. Cette opération de composition n'existe pas dans la méthode B événementiel. Nous exploitons la technique de raffinement pour composer les deux parties qui caractérisent la structure d'un système interactif.

### 5.1 Scénario 1 : raffinement du noyau fonctionnel

Le développement débute par la spécification de la partie noyau fonctionnel. Ci-dessous, nous présentons le modèle $MODEL_{NF}$ avec l'événement $Evt$ spécifié par des attributs du noyau fonctionnel. L'invariant $I(Var_{NF})$ du modèle ne concerne actuellement que les variables de la partie noyau fonctionnel. Par ailleurs, quand la garde de l'événement $Evt$ est vraie ($G_{NF}$ est vraie), cet événement est déclenché et la substitution $S_{NF}$ est alors exécutée.

$MODEL_{NF}$
**VARIABLES**
$Var_{NF}$
**INVARIANT**
$I(Var_{NF})$
...
$Evt =$
**SELECT**
$G_{NF}$
**THEN**
$S_{NF}$
**END;**

Dans la spécification du raffinement de l'abstraction précédente, nous introduisons de nouveaux attributs décrivant des éléments de la partie présentation. Ainsi nous composons la partie noyau fonctionnel avec la partie présentation au moyen de la technique de raffinement de manière à construire le contrôleur de dialogue. D'autres événements relatifs à la partie présentation peuvent être également introduits.

Ce raffinement impose l'ajout de l'invariant de collage $JCol(Var_{NF}, Var'_{NF})$ traduisant la correspondance entre les variables concrètes ($Var'_{NF}$) et abstraites ($Var_{NF}$) du noyau fonctionnel. Un invariant $I(Var_{IHM})$ relatif aux attributs de la partie présentation caractérise les propriétés de l'IHM. Ainsi la complexité des preuves du modèle $MODEL_{CD}$ dépend essentiellement de la complexité de l'invariant résultant. Par ailleurs, l'événement $Evt$ a été enrichi par les éléments de la présentation $G_{IHM}$ et $S_{IHM}$.

$MODEL_{CD}$
**VARIABLES**
$Var_{NF}, Var'_{NF}, Var_{IHM}$
**INVARIANT**
$JCol(Var_{NF}, Var'_{NF}) \wedge I(Var_{IHM})$
...
$Evt =$
**SELECT**
$G_{NF} \wedge \mathbf{G_{IHM}}$
**THEN**
$S_{NF} \| \mathbf{S_{IHM}}$
**END;**

### 5.2 Scénario 2 : raffinement de l'interface utilisateur

Ce scénario est dual du précédent. Le développement débute par la spécification abstraite de la partie présentation. Ci-dessous, nous présentons la spécification du raffinement $MODEL_{IHM}$. L'invariant $I(Var_{IHM})$ du modèle ne concerne que les variables de la partie présentation. L'événement $Evt$ est décrit par la garde $G_{IHM}$ et la substitution $S_{IHM}$ exprimées sur les variables de la présentation.

$MODEL_{IHM}$
**VARIABLES**
$Var_{IHM}$
**INVARIANT**
$I(Var_{IHM})$
...
$Evt =$
**SELECT**
$G_{IHM}$
**THEN**
$S_{IHM}$
**END;**

Dans le raffinement de l'abstraction de la partie présentation, nous introduisons de nouveaux attributs issus de la partie noyau fonctionnel. Dans le code présenté ci-dessous nous raffinons l'événement $Evt$ pour composer la partie interface utilisateur avec celle du noyau fonctionnel ainsi qu'éventuellement de nouveaux événements liés au noyau fonctionnel. Nous enrichissons la garde par le prédicat $G_{NF}$ et la post-condition par la substitution $S_{NF}$.

Nous retrouvons par ailleurs un invariant de collage $JCol(Var_{IHM}, Var'_{IHM})$ qui traduit la correspondance entre les variables abstraites ($Var_{IHM}$) et concrètes ($Var'_{IHM}$) de la présentation. Un invariant $I(Var_{NF})$ caractérise les propriétés du noyau fonctionnel.

Cet événement se déclenche quand les gardes de l'interface utilisateur et du noyau fonctionnel sont vraies et modifie ainsi les valeurs des attributs du système interactif complet.

$MODEL_{CD}$
**VARIABLES**
$Var_{IHM}, Var'_{IHM}, Var_{NF}$
**INVARIANT**
$JCol(Var_{IHM}, Var'_{IHM}) \wedge I(Var_{NF})$
...
$Evt =$
**SELECT**
$G_{IHM} \wedge \mathbf{G_{NF}}$
**THEN**
$S_{IHM} \wedge \mathbf{S_{NF}}$
**END;**

Notons que la différence entre les deux scénarios précédents est minime du point de vue conceptuel. Il s'agit d'une approche où on ajoute à un noyau fonctionnel (resp. une IHM) le composant IHM (resp. noyau fonctionnel). Du point de vue de la preuve, la différence est détectée au niveau des invariants à prouver. Cela dépend de la nature des applications à développer (une IHM complexe et un noyau fonctionnel simple par exemple).

### 5.3 Scénario 3 : produit de l'interface utilisateur et du noyau fonctionnel

A partir des deux modèles développés séparément correspondant respectivement aux parties noyau fonctionnel et présentation, un modèle résultant modélisant le contrôleur de dialogue du système interactif est dérivé.

Nous illustrons sur le code ci-dessous deux modèles indépendants. La spécification de la partie noyau fonctionnel est décrite par le modèle de gauche. La spécification de l'interface utilisateur est décrite sur le second modèle de droite. Chaque modèle possède des attributs, un invariant et un événement nommé respectivement $Evt_{NF}$ et $Evt_{IHM}$.

| $MODEL_{NF}$ | $MODEL_{IHM}$ |
|---|---|
| **VARIABLES** | **VARIABLES** |
| $Var_{NF}$ | $Var_{IHM}$ |
| **INVARIANT** | **INVARIANT** |
| $I(Var_{NF})$ | $I(Var_{IHM})$ |
| ... | ... |
| $Evt_{NF} =$ | $Evt_{IHM} =$ |
| **SELECT** | **SELECT** |
| $G_{NF}$ | $G_{IHM}$ |
| **THEN** | **THEN** |
| $S_{NF}$ | $S_{IHM}$ |
| **END;** | **END;** |

Dans le raffinement qui suit, nous composons les deux modèles précédents pour en spécifier un seul. Plus précisément, nous fusionnons les deux événements $Evt_{NF}$ et $Evt_{IHM}$ en un seul $Evt$. La garde résultante de l'événement $Evt$ est obtenue par une conjonction des prédicats des gardes de $Evt_{NF}$ et $Evt_{IHM}$. De même la postcondition résultante est établie par une exécution parallèle des substitutions $S_{NF}$ et $S_{IHM}$. Par ailleurs, l'invariant résultant est obtenu par la conjonction des invariants des modèles composés.

Notons que la spécification de l'événement $Evt$ est la même que celle obtenue après le raffinement des spécifications du premier et deuxième scénarios. Toutefois, l'invariant résultant est différent. Ce type de composition et notamment par la conjonction directe des invariants amène à une plus grande complexité des obligations de preuve puisque l'intérêt du raffinement de la méthode B n'est pas utilisé (absence de décomposition).

| $MODEL_{CD}$ |
|---|
| **VARIABLES** |
| $Var_{NF}, Var_{IHM}$ |
| **INVARIANT** |
| $I(Var_{NF}) \wedge I(Var_{IHM})$ |
| ... |
| $Evt =$ |
| **SELECT** |
| $\mathbf{G_{NF}} \wedge \mathbf{G_{IHM}}$ |
| **THEN** |
| $\mathbf{S_{NF}} \parallel \mathbf{S_{IHM}}$ |
| **END;** |

### 5.4 Scénario 4 : produit de l'interface utilisateur et d'une abstraction du noyau fonctionnel

Ce dernier scénario est une abstraction du précédent. Nous considérons la partie noyau fonctionnel comme étant abstraite. Ainsi, nous employons une simple variable entière qui détermine si l'état du noyau fonctionnel est modifié ou pas.

Nous donnons ci-dessous deux modèles concernant les parties noyau fonctionnel et interface utilisateur. Sur la gauche, nous utilisons la variable $var_{NF}$ pour sauvegarder l'état de la partie 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. Concernant l'événement $Evt_{IHM}$ il est identique à celui du troisième scénario. Il est à noter qu'à la différence du troisième scénario, l'invariant de la partie noyau fonctionnel est ici beaucoup plus simple (expressions logiques sur des variables booléennes entières).

| $MODEL_{NF}$ | $MODEL_{IHM}$ |
|---|---|
| **VARIABLES** | **VARIABLES** |
| $Var_{NF}$ | $Var_{IHM}$ |
| **INVARIANT** | **INVARIANT** |
| $var_{NF} \in \{0,1\}$ | $I(Var_{IHM})$ |
| ... | ... |
| $Evt_{NF} =$ | $Evt_{IHM} =$ |
| **SELECT** | **SELECT** |
| $var_{NF} = 1$ | $G_{IHM}$ |
| **THEN** | **THEN** |
| $var_{NF} = 0$ | $S_{IHM}$ |
| **END;** | **END;** |

Nous montrons ci-dessous le résultat de la composition obtenu par la technique de raffinement. En comparant avec le troisième scénario, $G_{NF}$ et $S_{NF}$ ont été remplacées par la modification des variables entières $var_{NF}$. L'invariant du raffinement reste du point de vue forme identique au troisième scénario. Toutefois la complexité des obligations de preuve du raffinement est moindre puisque l'invariant de la partie noyau fonctionnel est plus simple et ne s'applique qu'à des variables booléennes entières.

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

## 6 MISE EN PRATIQUE DE NOS SCENARIOS DE DEVELOPPEMENT : L'ETUDE DE CAS MATIS

Nous présentons dans cette section une application des scénarios de développement à l'étude de cas MATIS(MacColl & Carrington 19908, Nigay 1994). La comparaison des quatre scénarios de développement est réalisée sur la base du nombre d'obligations de preuve générées et qui sont elles relatives aux propriétés décrites dans les spécifications B.

### 6.1 Présentation générale de MATIS

*MATIS* (Multimodal Airline Travel Information System) (MacColl & Carrington 19908, Nigay 1994) est une application multimodale qui permet à un utilisateur d'extraire des informations sur les horaires de vols en utilisant la voix, la manipulation directe avec le clavier et la souris, ou une combinaison de ces modalités, en s'appuyant sur l'utilisation individuelle et synergique des nombreuses modalités d'entrées.

Pour chaque requête, une fenêtre est créée suite à un ordre de l'utilisateur. Elle contient des champs remplis avec les valeurs correspondants aux paramètres de la requête, tels que les noms de départ et d'arrivée du vol. L'utilisateur peut remplir ces champs en prononçant des noms de ville qui sont reconnus par le système, ou en cliquant sur les champs pour modifier le focus et sur la liste des noms de ville pour sélectionner les éléments à mettre dans les champs sélectionnés.

### 6.2 Quatre scénarios de développement : application à une étude de cas multimodale

Nous ne décrivons pas dans cet article le développement complet de l'étude de cas MATIS. Nous nous intéressons dans cette présentation à un seul événement appelé $Evt_{InputDeparture}$ permettant de choisir le nom de la ville de départ. Du point de vue de l'interface utilisateur, le choix du nom de la ville s'effectue au moyen de trois modalités (la voix, la souris et le clavier).

Les variables $MouseModal$, $KeyboardModal$ et $SpeechModel$ du modèle B événementiel sont respectivement associées à ces trois modalités. Par ailleurs la variable $PlaceDeparture$ est issue de la partie noyau fonctionnel et permet de sauvegarder le choix de la ville de départ (il y a 9 villes numérotées de 1 à 9). Enfin la variable $Ev1$ est utilisée pour jouer le rôle de variant assurant ainsi un déclenchement ordonné des différents événements que décrivent le modèle B événementiel. Nous montrons ci-dessus la spécification de l'événement $Evt_{InputDeparture}$ qui est identique à la fin de la modélisation pour chaque scénario.

Cet événement est déclenché par une action de l'utilisateur au travers des différentes modalités (exprimées dans la garde de l'événement) et modifie la variable $PlaceDeparture$ de la partie noyau fonctionnel, et le variant $Ev1$ décroît vers zéro pour déclencher les autres événements de la modélisation.

**SETS**
$State = \{on, off\}$
**VARIABLES**
$PlaceDeparture, MouseModal,$
$KeyboardModal, SpeachModal, \ldots$
**INVARIANT**
$PlaceDeparture \in 1..9 \wedge MouseModal \in State \wedge$
$KeyboardModal \in State \wedge SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \wedge$
$(MouseModal = on \vee KeyboardModal = on \vee$
$SpeachModal = on)$
**THEN**
$PlaceDeparture :\in 1..9 \parallel Ev1 := 0$
**END;**

#### 6.2.1 Scenario 1 : raffinement du noyau fonctionnel

Ci-dessous nous décrivons la modélisation de l'étude de cas correspondant au premier scénario.

**VARIABLES**
$PlaceDeparture, \ldots$
**INVARIANT**
$PlaceDeparture \in 1..9 \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1$
**THEN**
$PlaceDeparture :\in 1..9 \parallel Ev1 := 0$
**END;**

Notons que seuls les attributs du noyau fonctionnel ($PlaceDeparture$) sont utilisés dans la postcondition conformément au scénario.

Le code raffiné ci-dessous montre la modélisation de la composition de la partie noyau fonctionnel avec la partie présentation.

**SETS**
$State = \{on, off\}$
**VARIABLES**
$PlaceDeparture, MouseModal,$
$KeyboardModal, SpeachModal, \ldots$
**INVARIANT**
$PlaceDeparture \in 1..9 \wedge MouseModal \in State \wedge$
$KeyboardModal \in State \wedge SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \wedge$
**(MouseModal = on** $\vee$
**KeyboardModal = on** $\vee$
**SpeechModal = on)**
**THEN**
$PlaceDeparture :\in 1..9 \parallel Ev1 := 0$
**END;**

Nous enrichissons la spécification précédente en ajoutant trois variables de la partie présentation. Ces variables expriment les différentes modalités qui sont utilisées par l'utilisateur. Nous remarquons également que l'invariant de collage est trivial puisque nous manipulons dans ce cas précis de simple variable entière.

#### 6.2.2 *Scenario 2 : raffinement de l'interface utilisateur*

A cette étape du second scénario, seuls les attributs de la partie présentation sont utilisés. Nous les trouvons dans la garde de l'événement $Evt_{InputDeparture}$.

**SETS**
$State = \{on, off\}$
**VARIABLES**
$MouseModal, KeyboardModal,$
$SpeachModal, \ldots$
**INVARIANT**
$MouseModal \in State \wedge KeyboardModal \in State \wedge$
$SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \wedge$
$(MouseModal = on \vee KeyboardModal = on \vee$
$SpeechModal = on)$
**THEN**
$Ev1 := 0$
**END;**

Nous raffinons la spécification précédente pour composer les deux parties. Nous introduisons la variable *PlaceDeparture* de la partie noyau fonctionnel modifiée pas l'événement.

**SETS**
$State = \{on, off\}$
**VARIABLES**
$PlaceDeparture, MouseModal,$
$KeyboardModal, SpeachModal, \ldots$
**INVARIANT**
$PlaceDeparture \in 1..9 \wedge MouseModal \in State \wedge$
$KeyboardModal \in State \wedge SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \wedge$
$(MouseModal = on \vee$
$KeyboardModal = on \vee$
$SpeechModal = on)$
**THEN**
**PlaceDeparture**$:\in$ **1..9** $\parallel Ev1 := 0$
**END;**

#### 6.2.3 *Scenario 3 : produit de l'interface utilisateur et du noyau fonctionnel*

Nous donnons ci-dessous deux modèles correspondants aux deux parties distinctes du système interactif MATIS. Ces parties spécifient le même événement. Nous donnons sur la gauche les variables de la présentation tandis que sur la partie droite les variables du noyau fonctionnel.

| | |
|---|---|
| **SETS** | |
| $State = \{on, off\}$ | |
| **VARIABLES** | **VARIABLES** |
| $MouseModal,$ | $PlaceDeparture$ |
| $SpeachModal,$ | |
| $KeyboardModal, \ldots$ | |
| **INVARIANT** | **INVARIANT** |
| $MouseModal \in State \wedge$ | $PlaceDeparture \in 1..9 \wedge$ |
| $KeyboardModal \in State \wedge$ | $\ldots$ |
| $SpeachModal \in State \wedge \ldots$ | |
| $\ldots$ | $\ldots$ |
| $Evt_{InputDepartureIHM} =$ | $Evt_{InputDepartureNF} =$ |
| **SELECT** | **SELECT** |
| $Ev1 = 1 \wedge$ | $Ev1 = 1 \wedge$ |
| $(MouseModal = on \vee$ | **THEN** |
| $KeyboardModal = on \vee$ | $Ev1 := 0 \parallel$ |
| $SpeechModal = on)$ | $PlaceDeparture :\in 1..9$ |
| **THEN** | **END** |
| $Ev1 := 0$ | |
| **END** | |

Nous composons ces deux modèles par l'intermédiaire de la technique de raffinement. La garde qui en résulte est obtenue par une conjonction des deux gardes des précédents modèles. La substitution résultat est obtenue par une exécution parallèle des deux substitutions qui composaient les événements initiaux ($Ev1 := 0 \parallel PlaceDeparture :\in 1..9$).

**SETS**
$State = \{on, off\}$
**VARIABLES**
$PlaceDeparture, MouseModal,$
$KeyboardModal, SpeachModal, \ldots$
**INVARIANT**
$PlaceDeparture \in 1..9 \wedge MouseModal \in State \wedge$
$KeyboardModal \in State \wedge SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \wedge$
$(MouseModal = on \vee$
$KeyboardModal = on \vee$
$SpeechModal = on)$
**THEN**
**PlaceDeparture**$:\in$ **1..9** $\parallel Ev1 := 0$
**END;**

#### 6.2.4 *Scenario 4 : produit de l'interface utilisateur et d'une abstraction du noyau fonctionnel*

| | |
|---|---|
| **SETS** | |
| $State = \{on, off\}$ | |
| **VARIABLES** | **VARIABLES** |
| $MouseModal,$ | $var_{NF}$ |
| $SpeachModal, \ldots$ | |
| $KeyboardModal, \ldots$ | |
| **INVARIANT** | **INVARIANT** |
| $MouseModal \in State \wedge$ | $var_{NF} \in \{0, 1\} \wedge$ |
| $KeyboardModal \in State \wedge$ | $\ldots$ |
| $SpeachModal \in State \wedge \ldots$ | |
| $\ldots$ | $\ldots$ |
| $Evt_{InputDepartureIHM} =$ | $Evt_{InputDepartureNF} =$ |
| **SELECT** | **SELECT** |
| $Ev1 = 1 \wedge$ | $Ev1 = 1 \wedge var_{NF} = 1$ |
| $(MouseModal = on \vee$ | **THEN** |
| $KeyboardModal = on \vee$ | $Ev1 := 0 \parallel var_{NF} = 0$ |
| $SpeechModal = on)$ | **END** |
| **THEN** | |
| $Ev1 := 0$ | |
| **END** | |

Ci-dessus nous modifions la partie noyau fonctionnel du scénario 3 en substituant l'ensemble des attributs du noyau par une seule variable entière appelée $var_{NF}$.

**SETS**
$State = \{on, off\}$
**VARIABLES**
$var_{NF}, MouseModal,$
$KeyboardModal, SpeachModal, \ldots$
**INVARIANT**
$var_{NF} \in \{0,1\} \wedge MouseModal \in State \wedge$
$KeyboardModal \in State \wedge SpeachModal \in State \wedge \ldots$
$\ldots$
$Evt_{InputDeparture} =$
**SELECT**
$Ev1 = 1 \ \wedge var_{NF} = 1$
$(MouseModal = on \vee$
$KeyboardModal = on \vee$
$SpeechModal = on)$
**THEN**
$Ev1 := 0 \parallel var_{NF} = 0$
**END;**

Le modèle raffiné fusionne les deux précédentes spécifications. Ce modèle permet de vérifier d'une part l'IHM et d'autre part que le contrôleur de dialogue communique correctement avec le noyau fonctionnel.

### 6.3 Bilan

Le tableau ci-dessous illustre les résultats obtenus sur l'étude de cas MATIS. Les différents scénarios produisent un nombre d'Obligations de Preuve (OP) différents selon les scénarios. Il faut remarquer que le scénario 4 montre que la technique d'abstraction choisie réduit le nombre d'OP à prouver. Par ailleurs, les scénarios 1 et 2 produisent un nombre d'OP plus élevé du fait du nombre de raffinements nécessaires, mais plus simples que dans le cas de la composition parallèle du scénario 3 malgré le nombre plus faible d'OP de ce scénario.

| Scénario | OP Tri | OP | Nb Raf |
|---|---|---|---|
| 1 | 1133 | 127 | 3 |
| 2 | 1063 | 81 | 3 |
| 3 | 760 | 94 | 2 |
| 4 | 189 | 41 | 2 |

Table 1: Résumé des Obligations de Preuve (OP) des quatre scénarios de l'étude de cas MATIS (Scénario = Scénario de développement, OP Tri = Obligations de Preuve triviales, OP = Obligations de Preuve, Nb Raf = Nombre de Raffinements)

## 7 CONCLUSION

Le travail résumé dans cet article est une contribution à la modélisation et à la validation d'IHM multi-modales. Nous avons montré et évalué quatre approches différentes de développements formels de telles IHM. Ces différentes approches ont été mises en œuvre sur plusieurs cas d'études dans le cadre du projet RNRT-Verbatim.

Par ailleurs, nous avons abordé, pour le cas des IHM, la composition parallèle de deux modèles, exprimés en B événementiel, que sont noyau fonctionnel et la présentation pour modéliser le contrôleur de dialogue. Nous pensons que l'approche proposée pour la composition parallèle peut être étendue à toutes les compositions parallèles de deux modèles événementiels.

Enfin, notons que des travaux en cours sont menés pour permettre le test et l'animation de spécifications d'IHM multi-modales toujours dans le cadre de ce même projet.

## References

Abowd, G., Wang, H.-M. & Monk, A. (1995). A Formal Technique for Automated Dialogue Development, *in* G. Olsan & S. Schuon (eds), *Proceedings of DIS'95*, pp. 219–226.

Abrial, J. (1996a). *The B Book. Assigning Programs to Meanings*, Cambridge University Press.

Abrial, J.-R. (1996b). 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*, Nantes, France, p. 21.

Aït-Ameur, Y. & Baron, M. (2005). Bridging the gap between formal and experimental validation approaches in hci systems design : use of the event b proof based technique (revue), *Accepted in International Journal on Software Tools for Technology Transfer Special Section* .

Aït-Ameur, Y., Baron, M. & Girard, P. (2003). Formal validation of hci user tasks, *in* A.-A. Ban, A. H.R & M. Youngsong (eds), *The 2003 International Conference on Software Engineering Research and Practice - SERP 2003*, Vol. 2, CSREA Press, Las Vegas, Nevada USA, pp. 732–738.

Aït-Ameur, Y., Baron, M. & Nadjet, K. (2005). Encoding a process algebra using the event b method. application to the validation of user interfaces, *in* D. o. C. S. U. o. Loyola College

(ed.), *ISOLA 2005 - 2nd IEEE International Symposium on Leveraging Applications of Formal Methods*, Columbia, Maryland USA.

Aït-Ameur, Y., Bréholée, B., Girard, P., Guittet, L. & Jambon, F. (2004). Formal verification and validation of interactive systems specifications. from informal speciticications to formal validation, *Conference of Human Error, Safety and Systems Development, HESSD, Toulouse*.

Aït-Ameur, Y., Girard, P. & Jambon, F. (1998). Using the b formal approach for incremental specification design of interactive systems, *in* S. Chatty & P. Dewan (eds), *Engineering for Human-Computer Interaction*, Vol. 22, Kluwer Academic Publishers, pp. 91–108.

Baron, M. (2003). *Vers une approche sûre du développement des Interfaces Homme-Machine (Thesis)*, Thèse de doctorat, Université de Poitiers.

Bass, L., Hardy, E., Hoyt, K., Little, R. & Seacord, R. (1988). The arch model : Seeheim revisited, the serpent run time architecture and dialog model, *Technical Report CMU/SEI-88-TR-6*, Carnegie Melon University.

Bjorner, D. (1987). Vdm a formal method at work, *in* S.-V. LNCS (ed.), *Proc. of VDM Europe Symposium'87*.

Brun, P. (1997). XTL: a Temporal Logic for the Formal Development of Interactive Systems, *in* P. Palanque & F. Paterno (eds), *Formal Methods for Human-Computer Interaction*, Springer-Verlag, pp. 121–139.

Burbeck, S. (1992). Application programming in smalltalk-80: How to use the model-view-controller (mvc), *Report*.
*http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

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

Cansell, D. (2003). *Assistance au développement incrémental et à sa preuve*, Habilitation à diriger les recherches, Université Henri Poincaré.

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

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

Dijkstra, E. (1976). *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs.

Duke, D. & Harrison, M. (1993a). Towards a Theory of Interactors, *Technical report*, Amodeus Esprit Basic Research Project 7040, System Modelling/WP6.

Duke, D. & Harrison, M. D. (1993b). Abstract Interaction Objects, *Proceedings of Eurographics conference and computer graphics forum*, Vol. 12, pp. 25–36.

Hoare, C. (1969). An Axiomatic Basis for Computer Programming, *CACM* **12**(10): 576–583.

Jambon, F., Girard, P. & Aït-Ameur, Y. (2001). Interactive system safety and usability enforced with the development process, *in* R. M. Little & L. Nigay (eds), *Engineering for Human-Computer Interaction (8th IFIP International Conference, EHCI'01, Toronto, Canada, May 2001)*, Vol. 2254 of *Lecture Notes in Computer Science*, Springer, Canada, pp. 39–55.

Lano, K. (1996). *The B Language Method: A Guide to Practical Formal Development*, Springer Verlag.

MacColl & Carrington, D. (19908). *Testing MATIS: a case study on specfication based testing of interactive systems*, FAHCI ISBN 0-86339-7948.

Nigay, L. (1994). *Conception et Modélisation Logicielle des Systèmes Interactifs : Application aux Interfaces Multimodales*, Doctorat d'université (phd thesis), Université Joseph Fourier.

Palanque, P., Bastide, R. & Sengès, V. (1995). Validating Interactive System Design Through the Verification of Formal Task and System Models, *IFIP TC2/WG2.7 Engineering for Human-Computer Interaction*, pp. 189–212.

Paterno, F. & Faconti, G. (1992). On the LOTOS Use to Describe Graphical Interaction, *Proceedings of HCI, People and Computer*, Cambridge University Press, pp. 155–173.

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