Développement incrémental prouvé de systèmes répartis : le cas Mondex

Nazim Benaissa, Dominique Méry

▶ **To cite this version:**

Nazim Benaissa, Dominique Méry. Développement incrémental prouvé de systèmes répartis : le cas Mondex. [Rapport de recherche] 2008, pp.13. inria-00336655

HAL Id: inria-00336655
https://inria.hal.science/inria-00336655

Submitted on 4 Nov 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Développement incrémental prouvé de systèmes répartis : le cas Mondex

Nazim Benaïssa[1,2,4] and Dominique Méry[1,3,4]

[1] Université Henri Poincaré Nancy 1
[2] benaissa@loria.fr
[3] mery@loria.fr
[4] LORIA
BP 239
54506 Vandœuvre-lè-Nancy
France

Résumé Notre travail se situe dans le cadre du développement incrémental prouvé de systèmes distribués communiquant par des canaux non fiables, en particulier des protocoles de communication utilisés dans les porte-monnaie électroniques. L'approche B événementielle est mise en application sur une étude de cas réputée du Grand Challenge et appelée Mondex. Ce travail met en avant une explication des mécanismes sous-jacents qui permettent à ce système de satisfaire les exigences initiales résumées sous la forme suivante : il n'y a pas de perte d'argent ni de création d'argent lors des transferts. Le développement permet de préserver cette propriété de sûreté attendue du système final et identifie des relations de communication spécifiques pour assurer la mise en œuvre du protocole. Ainsi, un premier modèle énonce la spécification globale de sûreté ; un second modèle raffiné décrit le comportement détaillé des différents protagonistes ; un troisième modèle localise les données ; un quatrième modèle élimine des variables utiles pour faciliter les preuves. Cette étude nous permet de vérifier la robustesse du protocole à la perte de messages.

1 Introduction

Notre démarche vise à faire des études de cas de systèmes distribués communiquant par des canaux non fiables et de voir quelles sont les difficultés inhérentes à ce genre de systèmes afin de pouvoir en déduire plus tard des patrons de développement de systèmes distribués à l'aide de la méthode B événementielle. Pour y parvenir on doit s'appuyer sur des études de cas suffisamment complexes afin de témoigner de l'utilité des techniques de preuve et de raffinement. Aussi, notre choix de développer Mondex [4,1] s'appuie sur la complexité inhérente du cas d'étude et sur l'intérêt de cette étude dans le cadre du Grand Challenge [3]. Michael Butler [2] a développé ce système en B événementiel et nous apporterons quelques éléments de comparaison dans la dernière partie. On peut donc considérer notre travail comme un exercice de modélisation et une première étape pour la mise en œuvre de patrons de développement pour ce genre de système.

1.1 Présentation de Mondex

Mondex [4,1] est un système de paiement électronique qui fut introduit pour la première fois par la *National Westminster Bank* en 1990 avant d'être repris par *MasterCard*. Le système est basé sur les cartes à puce servant de porte-monnaie électroniques contenant chacun un solde d'argent. Une des spécificités du système est que les transactions sont *hors-ligne*, ceci

signifie que les transactions se passent entre deux porte-monnaie sans l'intervention d'une autorité centralisée. Au cours d'une transaction, chaque porte-monnaie doit donc être capable de garantir lui-même les mesures de sécurité inhérente à ce genre de système, sans l'intervention d'une quelconque autorité de régulation.

Le protocole *Mondex* Chaque porte-monnaie est initialement crédité d'un solde. La transaction consiste en un transfert d'argent entre un porte-monnaie source et un porte-monnaie destination. La figure 1 montre le schéma global d'une transaction.

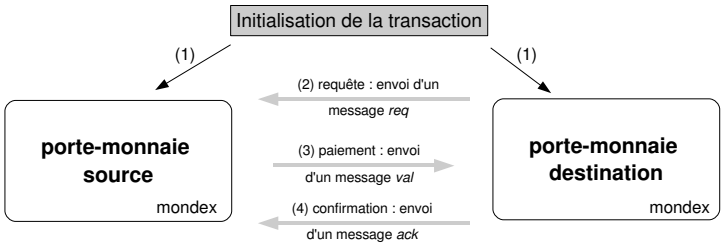

Fig. 1. Les différentes étapes d'une transaction

Le schéma de la figure 1 montre le cas d'une transaction où tout se passe bien. Mais dans la réalité plusieurs événements peuvent perturber le déroulement d'une transaction. Il peut s'agir par exemple d'une défaillance du canal de communication ou encore d'un retrait prématuré de la carte à puce du lecteur de carte, avant que la transaction n'arrive à son terme. Pour prévenir ce genre d'erreurs, un certain nombre de dispositifs ont été prévus. En plus de son identifiant et de son solde, un porte-monnaie contient un journal d'erreurs dans lequel les transactions qui ont échoué sont transcrites. Il contient également le numéro de séquence de la prochaine transaction à effectuer. Un porte-monnaie a aussi un état qui varie au fur et à mesure que la transaction avance. Les différents états possibles sont *eaFrom, eaTo, epr, epv, epa* (voir figure 2).

Lorsqu'un porte-monnaie source abandonne la transaction prématurément alors qu'il a déjà débité son solde (*état epa*), il inscrit la transaction en cours sur son propre journal d'erreurs. De même pour un porte-monnaie cible qui a envoyé un message de type *Req* et qui n'a pas encore crédité son solde (*état epv*). Grâce à ce procédé, le croisement des informations contenues dans les journaux d'erreurs des différents porte-monnaie permettra de récupérer l'argent apparemment perdu localement.

2 Développement du Mondex

Le développement du Mondex se fera en plusieurs étapes successives : un premier modèle abstrait suivi de dix raffinements qui rendront le modèle de plus en plus concret jusqu'à ce

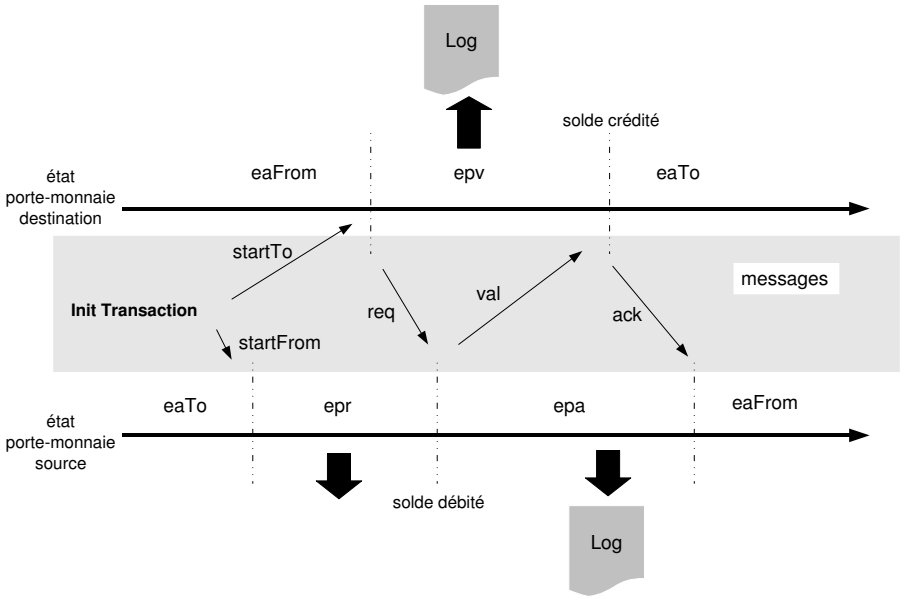

Fig. 2. Le protocole Mondex

que tous les aspects du protocole soient pris en considération.
Dans le premier modèle abstrait, la transaction entre deux porte-monnaie est atomique, le transfert d'argent se fait en un coup. Le solde du porte-monnaie source est débité au même temps que le solde du porte-monnaie cible est crédité (figure 3). Dans ce premier modèle sont

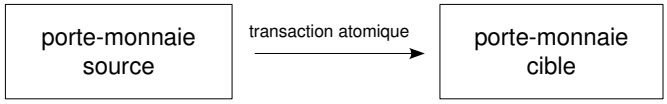

Fig. 3. Une transaction dans le modèle abstrait

exprimées les principales propriétés de sûreté que le système doit garantir, à savoir :
- Pas d'argent créé dans le système.
- Pas d'argent perdu dans le système.

2.1 Modèle abstrait : Transfert atomique

Dans ce premier modèle, les porte-monnaie ont un identifiant propre à chacun. Tous les identifiants possibles sont contenus dans un ensemble *NAME*. Chaque porte-monnaie contient en plus de son nom deux paramètres :

– Solde courant du porte-monnaie.
– Argent perdu localement : quand une transaction échoue, l'argent qui n'est pas arrivé au porte-monnaie cible est stocké comme argent perdu localement sur le porte-monnaie source.

Un certain nombre de types sont introduits pour les besoins de la modélisation :

SETS

$NAME$ /* ensemble des identifiants des porte-monnaie authentiques */

CONSTANTS

$VALUE$

AXIOMS

$axm1 : VALUE = \mathbb{N}$

END

L'état du système à un moment donné est reflété par des variables qui contiennent le solde et l'argent perdu localement par chaque porte-monnaie :

VARIABLES

$abAuthB$ /* solde de chaque porte-monnaie authentique */
$abAuthB'$ /* sauvegarde de l'ancienne valeur de la variable abAuthB */
$abAuthL$ /*argent perdu localement par chaque porte-monnaie authentique*/
$abAuthL'$ /* sauvegarde de l'ancienne valeur de la variable abAuthL */

INVARIANTS

$inv1 : abAuthB \in NAME \rightarrow VALUE$
$inv2 : abAuthB1 \in NAME \rightarrow VALUE$
$inv3 : abAuthL \in NAME \rightarrow VALUE$
$inv4 : abAuthL1 \in NAME \rightarrow VALUE$

L'invariant quand à lui exprime les deux propriétés de sureté énnocée précédamment, pas d'argent créé frauduleusement :

$inv5 : SUM(abAuthB) \leq SUM(abAuthB')$

Par ailleurs, l'argent ne peut être perdu globalement, ceci est exprimé par le fait que la somme des soldes et de l'argent perdu localement par tous les porte-monnaie demeurent inchangés :

$inv6 : SUM(abAuthB) + SUM(abAuthL) = SUM(abAuthB') + SUM(abAuthL')$

Les transactions étant atomique dans ce premier modèle, il y a donc que deux événements $TransfertOk$ et $TransfertLost$. En un coup la transaction est soit réussie ou non et les

variables sont modifiées en conséquence :

EVENT TransfertOk
ANY
$from$
to
v
WHERE
$grd1 : from \in NAME \wedge to \in NAME \wedge v \in VALUE$
$grd2 : abAuthB(from) \geq v$
$grd3 : from \neq to$
THEN
$act1 : abAuthB' := abAuthB$
$act2 : abAuthL' := abAuthL$
$act3 : abAuthB := abAuthB \Leftarrow \{from \mapsto (abAuthB(from) - v), to \mapsto (abAuthB(to) + v)\}$
END

Contrairement à l'événement $TransfertOk$, dans l'événement $TransfertLost$ l'argent n'est pas transféré vers le solde du porte-monnaie destination mais il est stocké comme argent perdu localement par le porte-monnaie source dans la variable $abAuthL$.

EVENT TransfertLost
ANY
$from$
to
v
WHERE
$grd1 : from \in NAME \wedge to \in NAME \wedge v \in VALUE$
$grd2 : abAuthB(from) \geq v$
$grd3 : from \neq to$
THEN
$act1 : abAuthB' := abAuthB$
$act2 : abAuthL' := abAuthL$
$act3 : abAuthB(from) := (abAuthB(from) - v)$
$act4 : abAuthL(from) := (abAuthL(from) + v)$
END

2.2 Premier raffinement

Pour implémenter le protocole Mondex (voir figure 2), chaque porte-monnaie électronique a une structure complexe comme le montre la figure 4.

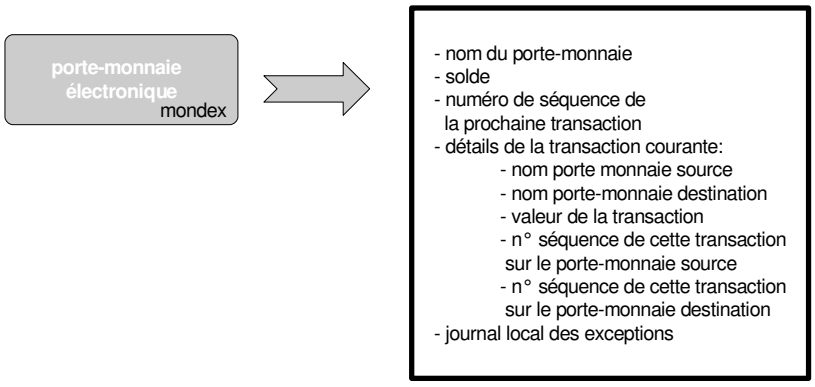

Fig. 4. La structure d'un porte-monnaie électronique

Il est donc nécessaire d'introduire les détails du protocole étape par étape pour réduire l'effort de preuve. Nous utiliserons pour cela la notion de transactions qui sera très abstraite dans le premier raffinement avant d'arriver aux transactions sous leur forme concrète de Mondex au bout du dixième raffinement. Une transaction concrète telle que montrée dans la figure 2 se déroule au niveau abstrait comme le montre la figure 5

Deux nouveaux ensembles porteurs sont introduits, *transactions* qui contient l'ensemble des toutes les transactions possibles. L'ensemble $STATUS$ contient quand à lui les états possibles d'une transaction. Chaque transaction a un porte-monnaie source, un porte-monnaie destination et une valeur.

SETS

$transactions$
$STATUS = \{ready, progress, success, fail\}$

CONSTANTS

t_src
t_dst
t_value

AXIOMS

$axm1 : t_src \in transactions \rightarrow NAME$
$axm2 : t_dst \in transactions \rightarrow NAME$
$axm3 : t_value \in transactions \rightarrow VALUE$

END

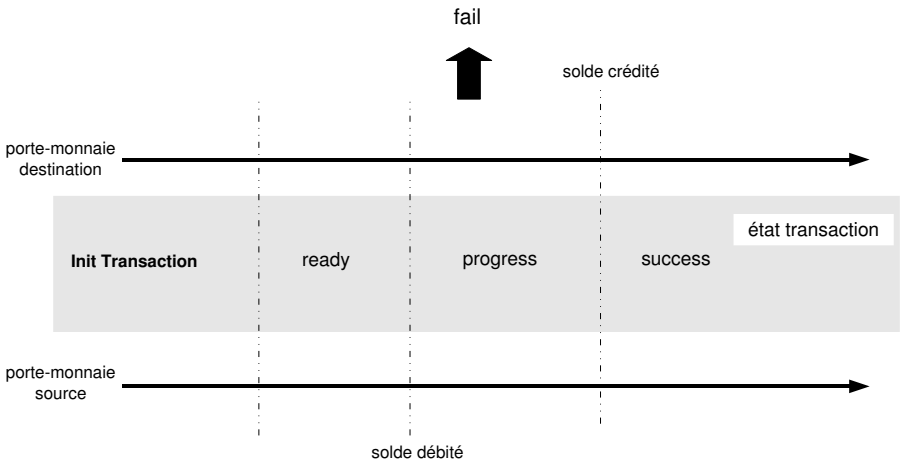

Fig. 5. Une transaction abstraite

L'état d'une transaction est donné par une variable *status* dont le domaine est un ensemble *trans* qui contient les transaction déjà terminées ou qui sont encore en cours :

$inv1 : trans \subseteq transactions$
$inv2 : status \in trans \rightarrow STATUS$

Un nouvel événement *initTrans* initialisant les transactions est introduit :

EVENT InitTrans
ANY
t
WHERE
$grd1 : t \in transactions$
$grd2 : t_src(t) \neq t_dst(t)$
$grd3 : t \notin trans$
THEN
$act1 : trans := trans \cup \{t\}$
$act2 : status := status \lhd\!\!\!\!- \{t \mapsto ready\}$
END

Dans ce premier raffinement, les transactions ne sont plus atomiques. Entre le moment où l'argent est débité du porte-monnaie source et le moment où il est recrédité dans le porte-monnaie destination, il s'écoule un temps où l'issue de la transaction est incertaine. Nous utilisons une variable *maybelost* qui contiendra pour un porte-monnaie donné les transactions dont l'issue est incertaine :

$inv3 : maybelost \in NAME \rightarrow \mathbb{P}(transactions)$

Le solde concret des porte-monnaie est donné par une variable *conBalance* :

$inv4 : conBalance \in NAME \rightarrow VALUE$

Comme le montre la figure 6, la relation entre le solde abstrait $abAuthB$ de la spécification et les deux variable *maybelost* et *conBalance* est exprimée dans l'invariant comme suit :

. $inv5 : \forall from \cdot (from \in NAME \Rightarrow$
$abAuthB(from) = conBalance(from) + SUMT(maybelost(from)))$

$SUMT$ étant une constante servant à calculer la somme d'argent contenu dans un ensemble de transactions donné.

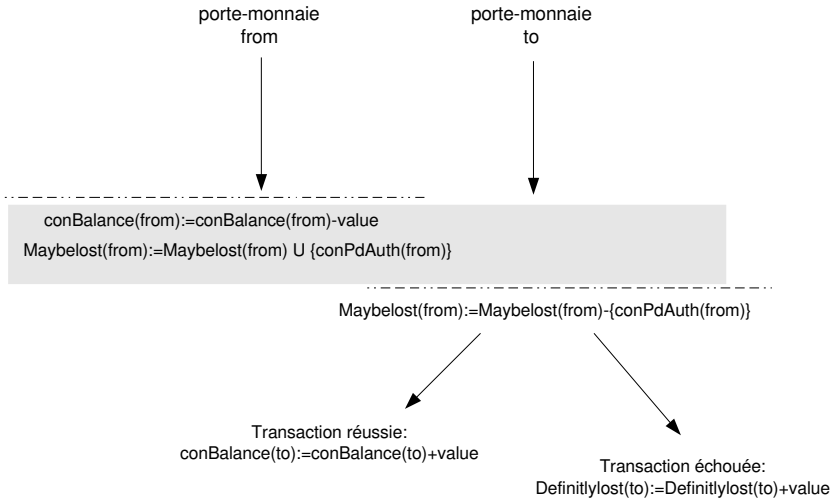

Fig. 6. La variable *maybeLost*

Un événement $transfert$ correspondant au débit du porte-monnaie source apparait dans ce raffinement :

EVENT transfert
ANY
t
WHERE
$grd1 : t \in transactions$
$grd2 : t_src(t) \neq t_dst(t)$
$grd3 : t \in trans$
$grd4 : t \notin maybelost(t_src(t))$
$grd5 : status(t) = ready$
$grd6 : conBalance(t_src(t)) \geq t_value(t)$
THEN
$act1 : maybelost(t_src(t)) := maybelost(t_src(t)) \cup \{t\}$
$act2 : status(t) := progress$
$act3 : conBalance(t_src(t)) := conBalance(t_src(t)) - t_value(t)$
END

Les deux événements$transfertOk$ et $transfertLost$ sont quand à eux raffinés comme suit :

EVENT TransfertOk
REFINES TransfertOk
ANY
t
WHERE
$grd1 : t \in transactions$
$grd2 : t_src(t) \neq t_dst(t)$
$grd3 : t \in trans$
$grd4 : t \in maybelost(t_src(t))$
$grd5 : status(t) = progress$
WITHNESSES
$from : from = t_src(t)$
$to : to = t_dst(t)$
$v : v = t_value(t)$
THEN
$act1 : maybelost(t_src(t)) := maybelost(t_src(t)) \setminus \{t\}$
$act2 : status(t) := success$
$act3 : conBalance(t_dst(t)) := conBalance(t_dst(t)) + t_value(t)$
END

EVENT TransfertLost
REFINES TransfertLost
ANY
t
WHERE
$grd1 : t \in transactions$
$grd2 : t_src(t) \neq t_dst(t)$
$grd3 : t \in trans$
$grd4 : t \in maybelost(t_src(t))$
$grd5 : status(t) = progress$
WITHNESSES
$from : from = t_src(t)$
$to : to = t_dst(t)$
$v : v = t_value(t)$
THEN
$act1 : maybelost(t_src(t)) := maybelost(t_src(t)) \setminus \{t\}$
$act2 : status(t) := fail$
$act3 : abAuthL(from) := (abAuthL(from) + v)$
END

2.3 Deuxième raffinement

Dans ce deuxième raffinement les variables *maybelost* et *abAuthL* sont supprimées et sont exprimée avec les autres variables à l'aide de l'invariant de collage :

$inv1 : \forall from \cdot (from \in NAME \Rightarrow$
$maybelost(from) = (dom(status \rhd \{progress\}) \cap dom(t_src \rhd \{from\})))$
$inv2 : \forall from \cdot (from \in NAME \Rightarrow$
$abAuthL(from) = (dom(status \rhd \{fail\}) \cap dom(t_src \rhd \{from\})))$

2.4 Troisième raffinement

Le but de ce raffinement est de remplacer l'état d'une transaction contenu dans la variable *s*tatus par l'état de chaqu'un des porte-monnaie impliqués dans cette transaction. L'état des porte-monnaie est contenu dans une nouvelle variable $c_$status. Dans Mondex, un porte-monnaie est dans l'un des états suivant :

- *eaFrom*, *eaTo* : le porte-monnaie n'est pas actif.
- *epr* : le porte-monnaie est source dans la transaction et il attend que la destination lui envoie la somme à verser.
- *epa* : le porte-monnaie est source de la transactions, il a versé l'argent à la destination et il attend l'acquittement.
- *epv* : le porte-monnaie est destination, il a fait la demande à la source et il attend le versement de la somme requise.

La figure 2 montre les différents états des porte-monnaie. La variable $c_$status est donc définie comme suit :

$axm1 : c_STATES = \{eaFrom, epr, epa, epv\}$
$inv1 : c_status \in NAME \rightarrow c_STATES$

Comme le montre la figure 4, chaque porte-monnaie connait à un moment donné les détails de la transaction dans laquelle il est partie prenante. Nous avons donc introduit une variable p_trans liant chaque porte-monnaie à la transaction dans laquelle il est impliqué :

$inv3 : p_trans \in NAME \rightarrow transactions$

Des invariants ont été rajoutés pour prouver ce raffinement. Ces invariants sont nécessaires pour décrire l'état d'une transaction avec l'état des porte-monnaie. Un exemple d'invariant est donné en ce qui suit :

$$\begin{aligned} inv6 : \forall t \cdot (t \in trans \wedge\ & p_trans(t_src(t)) = t \wedge \\ & p_trans(t_dst(t)) = t \wedge \\ & t_src(t) \neq t_dst(t) \wedge \\ & c_status(t_src(t)) = epa \wedge \\ & c_status(t_dst(t)) = epv \Rightarrow \\ & status(t) = progress) \end{aligned}$$

2.5 Quatrième raffinement

Ce quatrième raffinement vise à enlever la variable $trans$. Cette variable contient les transactions en cours ou les transactions déjà terminées. Dans le protocole Mondex chaque porte-monnaie a un numéro séquentiel qui sera associé à la prochaine transaction dans laquelle il sera partie prenante. Une variable p_seq est utilisée :

$inv1 : p_seq \in NAME \rightarrow \mathbb{N}$

A chaque transaction sont associés le numéro séquentiel du porte-monnaie source et celui du porte-monnaie destination au moment ou la transaction a commencé :

$axm1 : f_seq \in transactions \rightarrow \mathbb{N}$

$axm2 : t_seq \in transactions \rightarrow \mathbb{N}$

Pour enlever la variable $trans$, les invariants nécessaires sont les suivants :

$inv2 : \forall t \cdot (t \in transactions \wedge t \in trans \Rightarrow f_seq(t) < p_seq(t_src(t)))$

$inv3 : \forall t \cdot (t \in transactions \wedge t \in trans \Rightarrow t_seq(t) < p_seq(t_dst(t)))$

2.6 Cinquième, sixième et septième raffinements

Ces raffinements sont essentiellements des raffinements de données. A l'issue du septième raffinement, les porte-monnaie sont modélisés sous leur forme la plus concrète comme le montre la figure 4. Les variables suivantes sont introduites :

- identifiant du porte-monnaie source de la transaction en cours :
  $inv1 : p_from \in NAME \rightarrow NAME$
- identifiant du porte-monnaie destination de la transaction en cours :
  $inv2 : p_to \in NAME \rightarrow NAME$
- numéro séquentiel du porte-monnaie source :
  $inv3 : p_f_seq \in NAME \rightarrow \mathbb{N}$
- numéro séquentiel du porte-monnaie destination :
  $inv4 : p_t_seq \in NAME \rightarrow \mathbb{N}$
- valeur monnaitaire de la transaction en cours :
  $inv1 : p_value \in NAME \rightarrow \mathbb{N}$
- journal d'erreurs local :
  $inv1 : p_log \in NAME \rightarrow \mathbb{P}(NAME \times NAME \times \mathbb{N} \times \mathbb{N} \times \mathbb{N})$

2.7 Huitième raffinement

Un des principes essentiels des systèmes distribués est qu'une entité du système ne peut connaître l'état des autres entités. Une manière de synchroniser les différentes entités du système est l'utilisation des messages. Dans ce huitième raffinement divers types de messages sont introduits avec des variables contenant chacune un type de message (voir figure 2).

Rejeu et perte de messages Afin de tester la robustesse du système, des événements simulant la perte des messages aléatoirement ont été introduits. Les preuves de raffinement ont pu être faites ce qui prouve qu'en cas de disparition d'un message, la transaction ne peut aller à son terme mais que l'argent ne peut être perdu car en abondonnant la transaction le porte-monnaie rajoute la transaction en cours dans son journal d'erreurs.

Pour le rejeu, il faut noter que dans les différents événements les messages ne sont pas effacés après leurs lecture d'où le risque de lire plusieurs fois le même message. Ceci peut, par exemple, générer de l'argent illégalement. Dans notre cas, grâce à la présence du numéro de la transaction courante sur le porte-monnaie et sur le message, on arrive à prouver que message d'une transaction précédente ne peut être confondu avec un message de la transaction courante.

3 Conclusion

Nous avons dans cette étude modélisé et prouvé le protocole Mondex dans la perspective de développer des patrons pour les protocoles de communications à travers des canaux non fiables. Cette étude de cas nous a permis de voir qu'elles étaient les difficultés liées à ce genre de système et la manière avec laquelle les gérer :

- Les propriétés de sûreté du système qui doivent être préservées sont exprimées dans le modèle abstrait dans lequel la transaction est atomique.
- Modéliser d'abord des transactions abstraites et raffiner ensuite le système pour que les transactions ne soient plus atomiques jusqu'à arriver au protocole sous sa forme concrète.
- Les exceptions (transaction interrompues prématurément) sont gérées avec une variable abstraite *abAuthLost* qui est ensuite concrétisée dans les raffinements suivants (avec les journaux d'erreurs dans le cas Mondex).
- La nécessité de distinguer les messages de même type afin de prévenir les risques qu'un même message soit utilisé plusieurs fois. Dans le cas Mondex un numéro séquentiel associé à la transaction courante figurant sur les messages et les porte-monnaie est utlisé.
- Lorsqu'il y a perte des messages, la transaction doit pouvoir être interrompue par la partie qui attend ce message sans que cela ne remette en cause les propriétés de sûreté du système. Dans le cas Mondex, ceci est garanti par le fait qu'à chaque fois qu'un porte-monnaie interrompt une transaction il la rajoute dans son journal d'erreurs.

L'ensemble du processus de modélisation a produit 557 obligations de preuves réparties sur les différents raffinements comme suit :

Modèle	Nombre Total de PO	Automatique	Interactives
Modèle abstrait	27	27 (100%)	0 (0%)
Premier raffinement	45	45 (100%)	0 (0%)
Deuxième raffinement	17	17 (100%)	0 (0%)
Troisième raffinement	263	191 (72%)	72 (28%)
Quatrième raffinement	114	114 (100%)	0 (0%)
5^e,6^eet 7^eraffinement	104	104 (100%)	0 (86%)
Total	570	498 (87%)	72 (13%)

Notre modélisation est fondée sur le modèle Z [4] et comprend des variables intermédiaires permettant d'exprimer que la somme des soldes ne peut pas augmenter ce qui n'est pas le cas de l'article de Butler [2] nous montre que cet élément n'est pas intégré à ses modèles. Enfin, Butler ajoute un événement qui transfère l'argent se trouvant perdu vers le solde, alors que cette opération est à réaliser par la banque dans le cadre de la mise à jour de la carte.

Comme perspective à notre travail, il est envisagé d'obtenir à partir de cette étude de cas des informations pour dériver un patron de modélisation de ce type de systèmes répartis reprenant la même démarche et pouvant être réutilisé pour différents cas.

Références

1. The mondex electronic purse system. http ://www.mondex.com.
2. M. Butler. Mondex in Event B. Technical report, ECS, 2007.
3. T. Hoare. Grand Challenge GC6, 2005.
4. Susan Stepney, David Cooper, and Jim Woodcock. An electronic purse : Specification, refinement, and proof. Technical monograph PRG-126, Oxford University Computing Laboratory, July 2000.