Validation directe de la conformité d'une application interactive à son modèle de tâches

Loé Sanou – Patrick Girard – Laurent Guittet

Laboratoire d'Informatique Scientifique et Industrielle (LISI / ENSMA)
Téléport 2 – 1, avenue Clément Ader BP 40109
86961, Futuroscope Chasseneuil, France
{sanou, girard, guittet}@ensma.fr

RESUME
Cet article présente une technique de développement permettant de vérifier la conformité avec le modèle de tâches d'une application interactive à travers l'enregistrement des interactions de l'utilisateur sur l'interface. En établissant un lien entre les actions élémentaires de l'interface et les tâches élémentaires du modèle de tâches (modèle de tâches K-MAD), on génère un scénario suivant la DTD d'un scénario de K-MADe. Ce scénario obtenu lors de l'exécution peut être chargé dans l'environnement de simulation de K-MADe pour animer le modèle de tâches de l'application interactive et vérifier ainsi sa conformité.

MOTS CLES : Programmation sur exemple, Boîte à outils, Interaction homme-machine, Interface homme machine, Swing, Test d'interface, Modèle de tâches.

ABSTRACT
This article presents a technique of development that allows to check the conformity of user interactions on an interactive application interface by its task model. By establishing a bond between the interface elementary actions and the task model elementary tasks (K-MAD tasks model), one generates scenarii according to the DTD of K-MADe scenarii. These scenarii, obtained during the execution, can be transferred in the K-MADe simulation environment for checking with the interactive application task model.

CATEGORIES AND SUBJECT DESCRIPTORS: D.2.3 [Software Engineering]: Coding Tools and Techniques; D.2.5 [Testing and Debugging] : Testing tools; H.5.2 [User Interface] : Evaluation/Methodology

GENERAL TERMS: Experimentation, Verification, Standardization

KEYWORDS: Programming by Demonstration, Toolkit, Human computer interaction, Human computer interface, Swing events, Interface testing, Task model.

INTRODUCTION

La validation des systèmes interactifs est et demeure un problème difficile. Au-delà des propriétés ergonomiques classiques, le point crucial consiste à établir la conformité de l'application livrée avec les besoins du client, ce qui se traduit, dans une approche centrée-utilisateur, par une évaluation de l'adéquation de l'application aux buts de l'utilisateur.

À la suite de Norman, de nombreux travaux ont été menés dans le domaine de l'analyse de l'activité, débouchant sur une importante littérature tournant autour du concept de modélisation des tâches. Plusieurs notations, à la sémantique plus ou moins précise, ont ainsi été développées. Souvent créées dans le seul but de recueillir l'activité existante et de modéliser l'activité demandée pour l'application à réaliser, elles souffrent le plus souvent d'un manque de formalisme. UAN en est l'exemple le plus frappant. Diverses tentatives pour incorporer une sémantique précise ont été menées, avec par exemple les travaux sur CTT, ou plus récemment sur K-MAD. Ces approches permettent d'envisager une utilisation plus rigoureuse de ces outils.

Cependant, utiliser les modèles de tâches pour aider à concevoir les applications est paradoxalement resté relativement peu développé. Quelques approches ont tenté de générer le contrôle des applications à partir des modèles de tâches, comme TERESA par exemple. Malheureusement, la méthode utilisée ne garantit pas le respect des propriétés lors des transformations.

Pourtant, si l'on pouvait établir un lien entre l'application réalisée et le modèle de tâches, il serait possible de vérifier des propriétés importantes comme la conformité de l'application aux tâches prescrites, ou la complétude de l'application par rapport aux tâches. C'est à ce problème que notre travail souhaite apporter un début de réponse.

Nous sommes partis de l'idée que la vérification de la conformité d'une application à son modèle de tâches prescrites était une activité qui, si elle ne pouvait s'appuyer sur une preuve formelle, pouvait aisément relever du domaine du test. Est-il envisageable de réaliser des tests dont les résultats pourront être confrontés au modèle de tâches ? Si oui, ces tests peuvent-ils être enregistrés, pour éventuellement servir dans une approche de non-régression ? En nous appuyant sur la boîte à outils de programmation sur exemple PbDToolkit [], qui s'intègre à la boîte à outils Java Swing pour permettre l'incorporation aisée de fonctionnalités d'espionnage/rejeu des interactions dans les applications interactives, nous avons défini les bases d'une méthode constructive de test basée sur l'enregistrement des interactions de l'utilisateur.

A travers cet article, et à l'aide d'un exemple, nous montrons comment un développeur peut outiller de façon simple son application interactive pour permettre de vérifier la conformité de l'IHM avec un modèle de tâches. À partir de l'interface applicative, en exécution, des scénarii sont générés. Ces scénarii peuvent ensuite être chargés dans l'environnement de simulation pour être testés. Au final, des réponses claires sont fournies sur la conformité de l'application par rapport à son modèle de tâches.

Dans un premier temps, nous présentons l'interface utilisateur et le modèle de tâches en faisant ressortir le rôle du dernier dans la construction de l'interface utilisateur. Dans une deuxième partie, nous montrons les liens que l'on peut définir entre les actions élémentaires de l'interface utilisateur et les tâches élémentaires du modèle de tâches. Et enfin, la troisième partie présente la technique d'implémentation par la génération de scénarios à partir de l'interface utilisateur en exécution. Nous terminons par la présentation d'un module interne permettant cette implémentation ainsi que les perspectives de notre travail.

INTERFACE UTILISATEUR ET MODELE DE TACHES

Dans cette section, nous définissons l'interface utilisateur d'une application interactive. Dans un premier temps, nous présentons les modèles de tâches, puis nous discutons de leur rôle dans la conception d'une IHM.

Interface utilisateur d'une application

La plupart des systèmes informatiques possèdent une interface homme-machine, encore appelée interface utilisateur. Ces interfaces utilisateur constituent en général le seul point d'entrée de l'utilisateur [4]. Elles ont pour but d'établir un lien entre les fonctionnalités du système et l'utilisateur. À ce titre, elles doivent satisfaire à des critères de qualité spécifiques, découlant de la notion d'utilisabilité. Dans [5], on trouve ainsi une classification des interfaces utilisateurs en fonction des critères de qualités. Le point le plus important est que l'interface utilisateur doit être en adéquation avec les besoins et les capacités de l'utilisateur [6] [7] afin de permettre à cet utilisateur d'atteindre ses objectifs à travers des trajectoires d'interactions intuitives et sûres.

D'un autre côté, l'interface utilisateur peut être vue comme un ensemble de composants graphiques permettant à l'utilisateur d'appréhender l'état du système et de transmettre à ce dernier ses directives ou ses commandes. Au cours des vingt dernières années, les interfaces utilisateurs sont devenues de plus en plus complexes. Aujourd'hui, on trouve couramment des interfaces évoluées (généralisation de la manipulation directe, interaction post-WIMP, …) intégrant de nouvelles techniques d'interaction [8]. La prise en compte de cette complexité se traduit par une part de plus en plus importante du code dévolu à la programmation de l'IHM. Les conséquencdes sont de deux ordres : (1) le risque d'erreurs lors des étapes de conception augmente, et (2) les procédures de tests d'interface sont de plus en plus coûteuses, car non automatisables.

Afin de prendre en compte cette complexité croissante et par conséquent d'assurer l'utilisabilité des IHM, le recours à des méthodes et des notations s'est avéré nécessaire. Ces méthodes et modèles proviennent de différentes communautés (informaticiens, ergonomes, psychologues, …) et constituent aujourd'hui un champ de recherche très actif.

Modèle de tâches

Pour exprimer le besoin des utilisateurs, souvent des non informaticiens, de nombreuses notations de description ont été proposées dans le domaine des IHM. Ces notations appelées couramment modèles de tâches sont, pour la plupart, centrées utilisateurs et sont souvent loin des implémentations informatiques. Les modèles de tâches possèdent des qualités et des finalités très différentes. Différentes classifications ont été Un effort salutaire de simplification a été proposé, au travers de leur regroupement en deux catégories [12] : les modèles d'analyse de tâches et les modèles de description de l'interface homme machine.

L'analyse de tâches consiste à collecter des informations sur la façon dont les utilisateurs accomplissent une activité. L'acquisition de ces informations est obtenue par les récits des utilisateurs au moyen de lectures de rapports, d'interviews ou de simulations [5]. Mais cette analyse doit être indépendante de toute idée d'interface à réaliser et ne doit pas comporter de sous-entendus sur les dispositifs d'interaction. Elle doit donc rester à un haut niveau d'abstraction. Ces modèles liés à l'analyse de tâches servent essentiellement de notations support pour la collecte et l'analyse des informations. En l'espèce, ils sont rarement formalisés. Rentrent dans cette catégorie les modèles HTA (Hierarchical Task Analysis) [13] et MAD (Méthode Analytique de Description) [14].

Les modèles de description de l'IHM définissent la vue que l'utilisateur aura du système interactif. Ces modèles s'inscrivent dans une logique de continuité par rapport à celle de l'analyse de la tâche. En plus, ils doivent intégrer le retour d'information en provenance de l'interface. Les modèles de description de l'interface améliorent la communication entre le concepteur ergonomique et le concepteur de logiciel. On ici citer parler des modèles comme UAN (User Action Notation) [15] et CTT (ConcurTaskTrees) [16].

Un modèle récent se distingue en se positionnant dans les deux catégories, selon son utilisation. Il s'agit de K-MAD (Kernel of Model for Activity Description, ou Noyau du Modèle de Description de l'Activité en français, N-MDA) [17]. Largement inspiré de MAD, K-MAD donne une définition précise des opérateurs utilisés, et approfondit la notion d'objets manipulés par les tâches. Son outil K-MADe[1] [18], à l'instar de CTTe, permet d'éditer les modèles en respectant les contraintes du formalisme, de construire des scénarios et de simuler l'exécution de ces modèles. Il va plus loin que ce même CTTe en permettant de prendre en compte les valeurs des objets concernés par les conditions définies dans le modèle. De plus, une interface d'échange a été prévue, basée sur une description XML, permettant d'exporter, et surtout d'importer des modèles K-MAD ou des scénarii. L'ensemble de ces caractéristiques nous a conduit à choisir le modèle K-MAD comme support de notre travail.

Rôle du modèle de tâches dans la conception des IHM

Les notations permettent, d'une part, la collecte des informations sur la façon dont les utilisateurs accomplissent une activité, et d'autre part, la description de l'IHM qui définit la vue que l'utilisateur aura du système interactif. Ces notations peuvent constituer la base de départ de la réalisation d'une IHM. D'après [19], le modèle de tâches issu du domaine de l'IHM est utile pour deux raisons principales. D'une part, il peut servir d'outil dans la conception et d'autre part, il s'adresse à l'utilisateur pour l'aider dans son travail via sa documentation, définie dans la phase de conception. Nous ajouterons à ces deux points que le modèle de tâches est un bon candidat pour servir de support à la validation du système. De par sa nature, il est permet de vérifier les propriétés liées aux tâches.

Cependant, l'utilisation de ces modèles dans les processus de développement n'est pas aisé. En effet, la modélisation de la tâche suit un processus purement informel ; elle aboutit souvent à des représentations conceptuelles erronées et/ou ambiguës, pour cause de manque d'outils pour vérifier la cohérence de la modélisation. Après le développement de ces représentations, comment peut-on s'assurer du respect de l'ensemble des tâches et de leur ordonnancement ? Comment s'assurer que l'interface utilisateur respecte le modèle de tâches ? La majorité des approches développées jusqu'ici s'appuient sur la vérification de propriétés sur des modèles de l'application. Malheureusement la conformité de l'application aux modèles censés la représenter ne peut être prouvée…

La solution que nous proposons suit une démarche totalement différente. Elle s'appuie sur la définition d'un lien direct entre l'interface et le modèle de tâche, qui permet, en exécutant l'application, de vérifier que cette exécution reste conforme au modèle. Nous explorons cette voie dans la section suivante.

LIENS ENTRE MODÈLE DE TÂCHES ET INTERFACE UTILISATEUR

Tester une interface pour vérifier sa conformité au modèle de tâches nécessite d'être capable d'établir le lien entre ces deux éléments. Concrètement, il s'agit d'établir une correspondance entre certains éléments observables du dialogue home-machine et les tâches du modèle de tâches.

Tâches élémentaires du modèle de tâches

K-MAD est un modèle hiérarchique. Il représente l'activité de l'utilisateur sous forme d'un arbre de tâches, de la plus générale (tâche-racine) à la plus détaillée (actions élémentaires), en passant par des tâches intermédiaires. À partir de la tâche racine, chaque tâche peut avoir, d'une part des liens hiérarchiques avec une tâche hiérarchiquement supérieure (tâche-mère) ou inférieure (tâche-fille, lors de la décomposition de la tâche considérée), et d'autre part des tâches-sœurs, de même niveau hiérarchique. La *figure 1* presente le modèle de tâche du « Convertisseur Franc/Euro » qui constitue notre étude de cas. Sur la *figure 1*, *Conversion_Franc_Euro* modélise la tâche liée à l'application du convertisseur. Elle décrit le fait que l'utilisateur doit choisir la valeur à convertir (*Valeur_Conversion*), puis saisir cette valeur (*Saisir_Valeur*) en continu (tâche itérative) jusqu'à ce qu'il choisisse la conversion soit par *Conversion_Franc* soit par *Conversion_Euro*. Une fois le choix de la conversion effectué, la valeur convertie est affichée (tâche *Valeur_Convertie*). A tout moment du déroulement de la tâche *Conversion_Franc_Euro*, l'utilisateur peut la désactiver à l'aide de la tâche *Quitter*.

Une tâche est composée d'un ensemble de caractéristiques qui représentent le pouvoir d'expression relatif à la tâche. Ces caractéristiques se divisent en trois catégories :

- les caractéristiques générales (numéro, nom, durée, état, …) ;
- les caractéristiques liées à l'ordonnancement (décomposition, nécessité, interruptible, …) ;
- les caractéristiques liées aux traitements des actions (événement, postcondition).

[1] *http://www-rocq.inria.fr/merlin/kmade/*

Dans l'environnement K-MADe, l'ordre des sous-tâches d'une même tâche mère (priorité des tâches ou contrainte de précédence) est défini par la position sur l'espace de tâches. Une tâche placée plus à gauche est considérée prioritaire par rapport à celles qui sont situées sur la droite. L'une des caractéristiques générales d'une tâche est son exécutant. Il peut être :

- « Utilisateur » : la tâche est réalisée par un utilisateur autre qu'une action sur le système ; exemple : tâche *Valeur_Conversion* (*Figure 1)* ;
- « Système » : la tâche est réalisée complètement par le système ; exemple tâche *Valeur_Convertie* (*Figure 1*) ;
- « Interactif » : la tâche est déclenchée par l'utilisateur et réalisée conjointement sur le système ; exemple : tâche *Conversion_Franc* (*Figure1*) ;
- « Abstrait » : soit la tâche n'est pas complètement décrite, soit la tâche contient des sous-tâches d'exécutants différents ; exemple : tâche *Conversion_Franc_Euro* (*Figure 1*) ;
- « Inconnu » : il s'agit d'un état non déterminé.

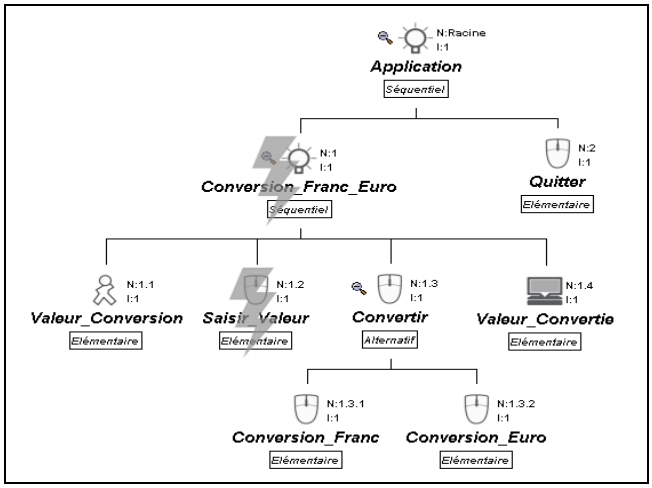

Figure 1 : Modèle de tâches du Convertisseur avec K-MADe.

Une tâche élémentaire du modèle de tâches correspond à une tâche plus détaillée d'une branche de l'arbre de tâches. Une tâche élémentaire du modèle de tâche est une action élémentaire non décomposable ou non décomposée du modèle de tâches. Une tâche élémentaire ne peut être de type « Abstrait » et est donc de type « Utilisateur », « Interactif », ou « Système ».

Les tâches élémentaires du modèle de tâches sont des tâches feuilles du modèle de tâches abstrait. Du point de vue de la décomposition hiérarchique des tâches, seules les tâches élémentaires ou feuilles sont susceptibles d'effectuer une action sur l'IHM. Toutes les tâches élémentaires du modèle de tâches sont des tâches feuilles. Du modèle de tâches de notre étude de cas, représenté par la *figure 1* ci-dessus, on déduit que les actions élémentaires sont les tâches *Saisir_Valeur*, *Conversion_Franc*, *Conversion_Euro*, *Valeur_Convertie* et *Quitter*.

Actions élémentaires de l'interface utilisateur

Une interface utilisateur d'une application est une couche applicative qui permet à l'utilisateur d'interagir avec le noyau de l'application. L'IHM est donc un élément médiateur. C'est à travers l'interface que l'utilisateur exerce les différentes actions lui permettant d'exécuter une tâche. Du point de vue informatique, l'interface utilisateur reflete exactement et fidèlement le comportement de l'application [6]. L'IHM, ensemble de composant graphique de présentation réagissant aux actions de l'utilisateur, peut prendre en compte plusieurs techniques d'interaction comme nous l'avons présenté plus haut dans la section précédente. Les actions élémentaires sont définies comme les interactions unitaires de l'utilisateur ou du système. Une interaction unitaire peut-être soit du type d'interaction utilisé, soit du type système. Le type d'interaction le plus répandu fait intervenir les dispositifs d'entrée standard, le clavier ou la souris. L'interaction unitaire du type système est en général la mise à jour de l'affichage suite à un changement de l'état interne du système. Le changement de l'état interne peut être aussi vu comme une action élémentaire. L'action élémentaire est une fonction de réalisation de l'action de l'utilisateur. Elle peut être le fruit d'une interaction élémentaire ou d'une suite d'interactions élémentaires dans la réalisation d'une tâche.

Dans l'interface utilisateur proposée à la figure 2 ci-dessous, les actions élémentaires se résument à la saisie d'une valeur dans la zone de champ « *Valeur à convertir* » et au clic sur le bouton « *Convertir en Euros* », ou sur le bouton « *Convertir en Franc* ». Toute autre action élémentaire issue d'une interaction quelconque de l'utilisateur ne sera pas considérée et par conséquent n'entraînera aucun changement de l'état du système.

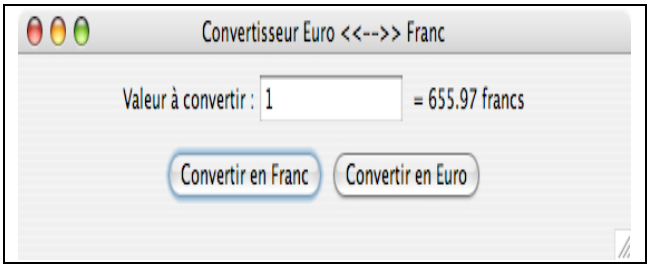

Figure 2 : Interface du Convertisseur Franc/Euro.

Liaisons entre les actions élémentaires du modèle de tâches et celles de l'interface utilisateur

Comment établir un lien entre l'application et le modèle de tâches ? Il est nécessaire, pour apporter une réponse à cette question, de savoir s'il existe une correspondance directe ou naturelle entre les tâches et l'interface.

L'interface étant caractérisée par un ensemble d'états accessibles par un ensemble de transitions, comment établir le lien entre ces transitions ou ces états avec les tâches du modèle ? Les différents types de tâches peuvent nous aider dans ce travail.

Si l'on élimine les tâches de type « Inconnu », qui ne sont présentes que pour permettre de représenter des modèles intermédiaires de conception, où l'on n'a pas encore décidé de la nature exacte de la tâche, quatre types de tâches sont à considérer. Les tâches de type « Abstrait » sont des éléments structurants du modèle de tâches. Elles sont essentielles à la définition de la dynamique du modèle (et donc à la réalisaiton de la simulation) car elles portent la plupart des opérateurs et attributs nécessaire à définir cette dynamique (opérateur de décomposition, multiplicité, interruptilibité, etc.). Malgré cela, elles ne rentrent pas en ligne de compte dans l'écriture de scénarii. En effet, ces derniers ne sont constitués que d'enchaînement de tâches « feuilles » du modèle, qui ne peuvent être de type « Abstrait », comme nous l'avons dit plus haut.

Les tâches de type « Utilisateur » et de type « Système » ne donne pas lieu à une interaction entre l'utilisateur et le système. Les premières ne sont liées au système que par le fait qu'elles requièrent, de la part du système, la mise à disposition des informations nécessaires à leur accomplissement. Par exemple, pour une tâche cognitive consistant à faire un choix, il est souhaitable que les alternatives du choix soient directement accessibles à l'utilisateur au moment où cette tâche doit s'exécuter. Un lien entre ce type de tâches et l'interface ne pourrait se faire qu'à travers des états de l'interface. Les tâches de type « Système » correspondent à une action de l'application, qui engendre normalement (principe d'observabilité) un retour d'information vers l'utilisateur. Ce retour d'information, qui constitue une transition entre deux états de l'interface, est un bon candidat pour établir un lien avec la tâche.

Les tâches de type « Interactif » sont les candidats naturels au lien avec l'interface. Elles correspondent à une interaction entre l'utilisateur et le système, donc obligatoirement à au moins une action du premier sur le deuxième.

Dans le cadre de notre étude de cas, comment s'établit cette correspondance entre tâches élémentaires et transitions de l'interface ? La tâche « Saisir_Valeur » correspond sur l'interface à la saisie d'une valeur numérique dans le champ de texte saisissable prévu à cet effet. Les deux tâches « Conversion… » correspondent à l'appui sur l'un des boutons représentés sur l'interface. Enfin, la tâche système est représentée par l'affichage de la valeur convertie dans l'étiquette prévue. On peut donc établir une table de correspondance très simple (Tableau 1).

Actions élémentaires de l'interface utilisateur	*Actions élémentaires du modèle de tâches*
Une saisie dans « Valeur à convertir »	« Saisir_Valeur »
Un clic sur le bouton « Convertir en Franc »	« Conversion_Franc »
Un clic sur le bouton « Convertir en Euro »	« Conversion_Euro »
Mise à jour du label d'affichage de la valeur convertie	« Valeur_Convertie »

Tableau 1 : Correspondance entre les actions élémentaires de l'interface et celles du modèle de tâches.

Notons que la relation qui s'établit entre modèle de tâches et application n'est en aucun cas une bijection. Rien n'empêche de prévoir la réalisation de la tâche par deux dispositifs d'interaction. Ce serait par exemple le cas si l'on décidait de doubler les boutons servant à la conversion par des items de menu. De même, rien n'interdit théoriquement (mais est-ce une bonne idée ?) d'utiliser le même dispositif d'interaction pour des rôles différents en fonction du contexte. La conséquence de ce point est que le lien n'est pas déterministe. Nous discuterons ce point dans l'analyse finale.

Voyons maintenant comment nous pouvons réellement tester une application à partir d'un lien entre elle-même et son modèle de tâche.

GÉNÉRATION DE SCÉNARII À PARTIR DE L'INTERFACE EN EXÉCUTION

Un scénario peut être défini théoriquement comme une utilisation particulière du système dans un contexte précis. Le modèle de tâches, lui au contraire est une description globale. Du point de vue de ce dernier, un scénario est en fait une succession de tâches élémentaires sans structuration autre que la séquence. Vérifier si le système est conforme au modèle de tâche consiste à s'assurer que la succession des tâches respecte les conditions posées par les opérateurs du modèle de tâches. Cette vérification peut se faire en utilisant un outil existant dans un outil comme K-MADe (ce serait aussi le cas avec CTTe) : l'outil de simulation. Si les scénarii issus de l'utilisation du système peuvent être exécutés avec succès par le simulateur de K-MADe, cela signifie qu'ils sont conformes au modèle de tâches.

Dans la section suivante, nous expliquons la technique de définition et d'implémentation des liens que nous avons privilégiée.

Définition des liens

Le concepteur connaît les tâches de son modèle et il sait ou prévoit que telle action du programme donnera ou devrait donner tel ou tel résultat. Il est le mieux placé pour décrire l'état de son système à tout moment. L'IHM transmettant les interactions de l'utilisateur au noyau du système englobe les liaisons. La boîte à outils que nous avons utilisée se situe au niveau des composants élémen-

taires de l'interaction. Nous avons donc choisi de permettre la définition des liens entre interface et modèle de tâches au moment de la programmation de ces mêmes composants. Les tâches de type « Interactif » correspondent à des actions de l'utilisateur. Dans le cas de Swing, ces actions passent par le mécanisme de l'abonnement. Pour ce qui est tâches de type « Système », le retour d'information vers l'utilisateur se fait par un mécanisme de notification des composants de l'interface. La définition du lien entre interface et modèle de tâches consiste à profiter de ces deux mécanismes. Par exemple, dans le cas de l'exemple sur le « *Convertisseur Franc / Euro* », lors de l'abonnement de l'écoute du clic sur le bouton « *Convertir en Franc* » (cf. *Figure 2*), le développeur précisera la correspondance avec la tâche élémentaire du modèle de tâche « *Conversion_Franc* » (*Figure 1*) comme montré dans le *tableau 1* ci-dessus. Si tous les traitements sont pris en compte par le concepteur dans l'interface utilisateur, et la correspondance avec les objets concrets du modèle de tâches effectués, alors nous pouvons retrouver l'ensemble des interactions possibles autorisées sur le système par le modèle de tâches lors de l'analyse.

Technique d'implémentation des liens

L'implémentation de la technique de correspondance entre les tâches élémentaires du modèle de tâches et les actions élémentaires de l'interface utilisateur peut se faire par un double abonnement ou par la modification d'une méthode par un ajout d'attribut. Pour faciliter cette définition, nous avons redéfini les composants graphiques de manière à leur ajouter, au niveau des interfaces d'écoute, un attribut de type String. Cet attribut correspond au nom de la tâche élémentaire du le modèle de tâches.

Suivre les écoutes de l'IHM est une forme d'espionnage du système. La question que nous nous sommes posée est de savoir à quel niveau du système ce suivi doit être effectué. Le choix du niveau est important en fonction du but recherché [20]. Plus l'enregistrement s'effectue à un niveau élémentaire dans le dialogue entre l'utilisateur et l'application, plus il sera dépendant des changements intervenant dans l'interface utilisateur de l'application. L'influence sur la notion de test de non-régression est immédiatement perceptible. Comme nous l'avons déjà montré plus haut, les actions élémentaires à prendre en compte sont les fonctions de réalisation des actions de l'utilisateur que nous devons considérer lors du suivi ou de l'enregistrement de ces mêmes interactions. Ce ne sont pas les actions de l'utilisateur sur l'interface applicative qui sont enregistrées en réalité, mais ce sont les différentes opérations ou fonctions liées aux objets de l'interface graphique. Il s'agit là d'un enregistrement au niveau syntaxique. Cela signifie que les actions élémentaires de l'interface utilisateur sont situées au niveau syntaxique de l'application. A quel niveau se situent donc les tâches élémentaires du modèle de tâches ? Dans le modèle de tâches, c'est l'ordonnancement des tâches qui définit leur disponibilité ou leur activation. Si une tâche s'active après une autre, alors une logique de raisonnement est respectée et vu qu'il existe en général des conditions, le résultat est nécessairement significatif. Les tâches élémentaires sont donc au niveau sémantique dans le modèle de tâches. On utilise un lien sémantique vers le modèle de tâches alors que l'espionnage se fait au niveau syntaxique dans l'application. La phase manuelle de l'opératino de lien se trouve parfatiement justifiée par cette augmentaiton de niveau d'abstraction.

Concrètement, pour l'étude de cas du « *Convertisseur Franc/Euro* », nous avons redéfini le composant JButton en un composant PbDButton qui hérite ainsi de toutes les propriétés du JButton[2]. Dans cette nouvelle classe, nous avons surchargé la méthode d'abonnement « *addActionListener* » en ajoutant un attribut pour permettre de préciser le nom de la tâche élémentaire correspondante. Sa structure est présentée comme suit :

```
public class PbDButton extends JButton {
        ------------------------------
        ------------------------------
        public void addActionLitener(
                ActionListener action, String nomtache) {
                super.addActionListener(action);
                Enregistrement de  la tâche « nomtache »
                ------------------------------
        }       ------------------------------ }
```

Il aurait été possible de ne pas redéfinir les composants. Au prix pour le développeur d'une instruction supplémentaire à ajouter à chaque abonnement, effort de programmation supplémentaire que nous avons souhaité éviter, pour éviter les risques d'erreur ou d'oubli.

Génération de scénarii

Nous avons déjà écrit que le scénario est en fait un enchaînement de tâches élementaires. Dans K-MADe, la manipulation de scénarii distingue deux aspects : l'enregistrement de scénarii et le rejeu de scénarii. L'enregistrement consiste à simuler un arbre de tâches en choisissant, tout au long de ses différents états, les actions à appliquer d'une part, et les valeurs particulières d'autre part. Le rejeu d'un scénario consiste à parcourir un chemin particulier donné par le scénario dans le modèle de tâches K-MAD. En fonction du but recherché par le test, plusieurs aspects peuvent être vérifiés : atteignabilité de la tâche dans le modèle ; atteignabilité d'un état particulier dans les objets du modèle concret ; finalisation complète de la suite de tâches dans l'arbre de tâches ; etc.

Pour générer un scénario à partir de l'application, nous nous sommes appuyés sur les capacités d'enregistrement/rejeu de la PbDToolkit. Les noms des tâches insérés par le développeur sont accumulées dans une liste au fur et à mesure de l'exécution de l'IHM. À

[2] *http://java.sun.com/j2se/1.5.0/docs/api/*

chaque interaction significative de l'utilisateur (i.e. instrumentée par le concepteur), le système engendre la tâche élémentaire correspondante que l'on insère dans la liste. L'ordre d'insertion dans la liste determine l'ordre d'ordonnancement des tâches. Pendant la durée d'une exécution, l'utilisateur réalise ainsi un certain nombre de tâches élémentaires pour atteindre son but. Le résultat de l'espionnage de cette exécution construit donc automatiquement un scénario. Cette phase est tout à fait analogue à la phase d'enregistrement d'un scénario dans l'environnement K-MADe. Seules les identités des tâches sont enregistrées. Le contexte n'est pas pris en compte ce qui fait qu'aucune contrainte n'est jointe à la tâche issue de l'IHM.

Dans un premier temps, nous nous sommes contentés de récupérer les différentes tâches engendrées suite aux interactions de l'utilisateur dans une liste sans nous préoccuper de la transcription en fichier répondant au format d'un fichier de scénarii sous K-MADe. A l'issue de cet enregistrement, notre outil produit un document XML compatible avec K-MAD. Un fichier de scénario dans K-MADe est en effet un fichier XML respectant une DTD (Document Type Definition) particulère dont nous présentons un extrait dans la *figure 4* ci-dessous. Ce fichier XML pourra être rejoué dans l'espace de simulation de l'environnement K-MADe. Les valeurs des contraintes n'ayant pas été prises en compte sont nulles dans le fichier scénario. Dans l'état actuel de K-MADe, le déroulement du rejeu se fait suivant l'animation du modèle de tâches et les erreurs dans le scénario sont détectées ou présentées de façon simple dans l'environnement : ou la simulation se termine, ou la tâche suivante n'est pas atteignable.

```
<!ELEMENT scenario (execute | pass | interrupt | resume | cancel)* >
<!ATTLIST scenario name CDATA #REQUIRED >
<!ATTLIST scenario date CDATA #IMPLIED >
<!ATTLIST scenario comment CDATA #IMPLIED >
<!ATTLIST scenario idTask CDATA #REQUIRED >
<!ATTLIST scenario nameTask CDATA #REQUIRED >
<!ELEMENT execute (userExecutingConstraint?, userPreValue*, userPreConcreteObject*,
userPostValue*, userPostConcreteObject*, userIterValue*,
userIterConcreteObject*) >
<!ATTLIST execute idTask CDATA #REQUIRED >
<!ATTLIST execute nameTask CDATA #REQUIRED >
<!ELEMENT userExecutingConstraint EMPTY >
<!ATTLIST userExecutingConstraint idUser CDATA #REQUIRED >
<!ATTLIST userExecutingConstraint nameUser CDATA #REQUIRED >
```

Figure 4 : Extrait de la DTD[3] d'un scénario K-MAD.

[3] *Extrait du manuel utilisateur de K-MADe*

Schéma global de réalisation

Notre vision est de fournir un outil complet à partir de la boîte à outils Swing de Java permettant aux développeurs de pouvoir enrichir leurs applications en incluant les techniques de la programmation sur exemple et en permettant une vérification automatique de la conformité de l'interface utilisateur avec le modèle de tâches. Il ne s'agit pas d'inventer un système propriétaire, mais au contraire de s'appuyer sur l'extensibilité naturelle des boîtes à outils pour construire un système généralisable. Il faut permette au système d'être indépendant de la manière dont les actions sont crées, composées, et représentées.

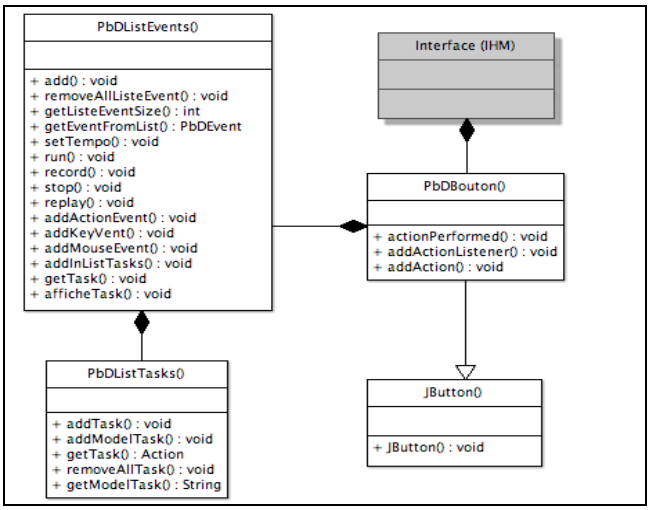

Figure 4 : Modèle UML du système pour l'enregistrement des actions utilisateur avec l'enrichissement du composant JButton.

ANALYSE DE LA SOLUTION

La solution que nous présentons ici, bien que très simple dans son principe, permet aisément de relier une application concrète avec un modèle de tâche censé la représenter. L'application directe qui consiste à vérifier que l'application est bien conforme au modèle de tâche permet de montrer de façon indubitable des erreurs de conception. Par exemple, la réalisation de la tâche de conversion suppose la saisie d'une valeur à convertir valide. Cette propriété est exprimable dans le modèle de K-MAD par l'intermédiaire d'une pré-condition de la tâche « Convertir », qui engendre une interdiction de cette tâche tant que l'on n'a pas commencé la tâche de Saisie. Si l'application n'implémente pas cette vérification, il sera possible de créer un scénario commençant par cette même tâche de conversion, ce qui est non conforme au modèle.

L'aspect non déterministe du lien établi n'est pas un problème majeur. Il engendre simplement une multiplication des scenarii à chaque choix dans le sens application vers modèle de tâches, alors qu'il n'a pas d'incidence dans l'autre sens.

On peut cependant reprocher à cette méthode d'être plus destructrice que constructrice. En effet, les résultats les plus faciles à mettre en évidence sont des non conformités au modèle de tâches : dès qu'un scénario n'est pas conforme, il arrête le simulateur. À l'inverse, il n'est possible de prouver la complétude de l'application par rapport aux tâches prescrites qu'en bâtissant un jeu de scénarii couvrant tous les chemins du modèle de tâche, ce qui est toujours très difficile en test. De plus, le diagnostic d'erreur est très pauvre : en l'état actuel du simulateur de K-MAD, qui, rappelons-le, n'a pas été conçu pour cet usage, il est très difficile de comprendre où se situe l'erreur détectée.

De nombreuses pistes s'ouvrent à nous pour enrichir les capacités de cette approche. K-MAD permet de définir des objets servant à construire des expressions entrant en ligne de compte dans l'évaluation des pré et post-conditions sur les tâches. Notre méthode d'établissement du lien entre application et modèle de tâches peut s'étendre à cet aspect. Afin d'obtenir un outil permettant d'effectuer des tests de non-régression, il faudrait être en mesure de rejouer un scénario validé sur l'application ; l'utilisation de l'approche sur exemple pour construire le scénario entrouvre de nombreuses portes. Enfin, il convient de déterminer une méthode pour obtenir d'autres résultats que la conformité aux tâches, comme la complétude des tâches.

CONCLUSION

Les techniques de développement des IHM doivent assurer que ces dernières satisfont les propriétés qui traduisent les exigences de l'utilisateur. La construction d'IHM à partir d'un modèle de tâches est un domaine en pleine expansion actuellement. La mise à disposition de techniques permettant la vérification de la conformité des actions élémentaires de l'IHM avec le modèle de tâches sera une avancée et facilitera la validation des IHM d'une manière générale.

Nous avons montré dans ce travail comment implémenter une méthode simple permettant de générer, directement à partir de l'application finale, des scénarii vérifiable sur le modèle de tâches censé représenter l'application. Cette méthode utilise une approche basée sur l'utilisation de Java Swing. Cette réalisation est une première étape dans la définition d'outils plus complet de vérification et validation des IHM.

BIBLIOGRAPHIE

1. Bass, L. and J. Coutaz, Developing Software for the User Interface. The SEI Series in Software Engineering. 1991: Addison-Wesley. 256.
2. Curtis, B. and B. Hefley, A Wimp No More: The Maturing of User Interface Engineering. Interactions, 1994: p. 23-34.
3. Paternò, F., Model-Based Design and Evaluation of Interactive Applications. 2001: Springer. 208.
4. Myers, B.A., User Interface Software Tools. ACM Transactions on Computer Human Interaction, 1995. 2(1): p. 64-103.
5. Dix, A., et al., Human-Computer Interaction. Second Edition ed. 1998: Prentice Hall. 638.
6. Coutaz, J. Interface Homme-Machine : un regard critique. in Journées d'Études AFCET : Interfaces Homme-Machine. 1992. Paris: AFCET.
7. Shneiderman, B., Designing the User Interface. 3 ed. 1998: Addison-Wesley. 639.
8. Myers, A.B. and M.B.S. Rosson. Survey on user interface programming. in Human Factors in Computing Systems (CHI'92). 1992. Monterey, USA: ACM/SIGCHI.
9. Brun, P., XTL : une logique temporelle pour le spécification formelle des systèmes interactifs, in LRI. 1998, Université Paris -Sud: Orsay.
10. Navarre, D., Contribution à l'ingénierie en Interaction Homme-Machine, in LIHS. 2001, Université Toulouse 3: Toulouse. p. 219.
11. Limbourg, Q. and J. Vanderdonckt, Comparing Task Models for User Interface Design (Chapter 6), in The Handbook of Task Analysis for Human-Computer Interaction, D. Diaper and N. Stanton, Editors. 2003, Lawrence Erlbaum Associates.
12. Baron, M., Vers une approche sûre du développement des Interfaces Homme-Machine (Thesis). 2003, Université de Poitiers: Poitiers. p. 255.
13. Shepherd, A., Analysis and training in information technology tasks, in Task Analysis for Human-Computer Interaction, D. Diaper, Editor. 1989, Ellis Horwood: Chichester, USA. p. 15-55.
14. Scapin, D. and J.-M.C. Bastien, Analyse des tâches et aide ergonomique à la conception : l'approche MAD* (chapitre 3), in Analyse et conception de l'I.H.M. / Interaction Homme-Machine pour les S.I. vol.1, C. Kolski, Editor. 2001, Hermès Science: Paris, France.
15. Hix, D. and H.R. Hartson, Developping user interfaces: Ensuring usability through product & process. Wiley professional computing. 1993, Newyork, USA: John Wiley & Sons, inc.
16. Paternò, F., G. Mori, and R. Galimberti. CTTE: An Environment for Analysis and Development of Task Models of Cooperative Applications. in ACM CHI 2001. 2001. Seattle: ACM Press.
17. Baron, M., et al. K-MADe : un environnement pour le noyau du modèle de description de l'activité. in IHM 2006. 2006. Montréal, Canada: ACM.