Résumé dans : Jambon F. & Boisdron Y. Vers une validation du dialogue homme-machine pour les Applications Graphiques Interactives de Conception Technique. ***Onzième conférence sur l'Interaction Homme-Machine (IHM'99), Montpellier, France***, 22-26 novembre 1999. p. 189.

Vers une validation du dialogue homme-machine pour les Applications Graphiques Interactives de Conception Technique

Francis JAMBON et Yohann BOISDRON

LISI/ENSMA
Téléport 2, 1 av. Clément Ader, B.P. 40109
86961 Futuroscope Chasseneuil Cedex, France
Francis.Jambon@ensma.fr, boisdryn@descartes.ensma.fr
Téléphone : (33/0) 5 49 49 80 70
Télécopie : (33/0) 5 49 49 80 64

RÉSUMÉ

Cet article s'intéresse à la validation du dialogue homme-machine d'une application interactive à partir d'une analyse de la tâche de l'utilisateur. Nous avons centré notre étude sur une classe particulière d'applications, les Applications Graphiques Interactives de Conception Technique (AGI-CT), réalisées selon le modèle d'architecture H^4. Notre processus de validation se base sur une génération automatique de vecteurs de tests à partir de l'analyse de la tâche. Ces vecteurs sont ensuite injectés à un simulateur qui réutilise le code exécutable du Contrôleur de Dialogue de l'application à tester. Le simulateur vérifie l'atteignabilité et la complétude de l'interaction, et permet ainsi de valider le dialogue.

MOTS CLÉS : Validation du dialogue, analyse de tâche, Applications Graphiques Interactives de Conception Technique, modèle d'architecture H^4.

INTRODUCTION

Les Applications Graphiques Interactives de Conception Technique (AGI-CT en abrégé) sont des logiciels graphiques interactifs très représentés en milieu industriel ; on pense naturellement à la mécanique, mais aussi à la conception de circuits pour l'électronique, ou encore à l'architecture, l'aménagement de bureaux… Le processus de conception de l'Interface Homme-Machine des AGI-CT s'effectue aujourd'hui en imitant le style d'interaction des systèmes existants, et non pas à partir d'une analyse de la tâche de l'utilisateur. À notre connaissance, aucune étude de la tâche des utilisateurs n'a été réalisée sur ces applications.

De multiples raisons peuvent expliquer cette pratique, mais il est vraisemblable que l'étendue des fonctions disponibles est encore un argument plus "commercial" que l'utilisabilité du système. De plus, les tâches et les objets manipulés dans un AGI-CT ont certaines particularités qui rendent l'analyse de tâche et la conception délicates [6]. Parmi les plus saillantes, citons :

- *Les tâches sont multi-objets* : une tâche peut faire référence à plusieurs objets du domaine sémantiquement distincts. Notons ici qu'il ne s'agit pas d'un simple mécanisme de groupement comme MacDraw™, qui permet d'effectuer une seule tâche sur un groupe d'objets sémantiquement considéré comme un seul objet. Par exemple, la tâche de créer un cercle peut nécessiter, comme paramètres, deux objets du domaine : un point et une distance.
- *Les tâches sont structurées* : l'exécution d'une tâche peut imposer l'exécution de sous-tâches, définissant ainsi une structure hiérarchique obligatoire de tâches/sous-tâches. Ceci a pour conséquence d'obliger le Contrôleur de Dialogue à mémoriser les tâches en cours. Par exemple, la tâche précédente de créer un cercle oblige la désignation d'un point et le calcul d'une distance.
- *Les objets du domaine sont relationnels* : la présentation des objets du domaine est non seulement dépendante de leur état, mais aussi de l'état d'autres objets. Par exemple, une vue 3D doit prendre en compte les faces cachées par d'autres objets.
- *Les objets du domaine sont structurés* : un objet peut être composé d'une hiérarchie d'objets liés. Par exemple, un cube est composé de faces, qui elles-mêmes sont composées d'arêtes que l'utilisateur peut être amené à désigner.

En conséquence, il n'est pas aujourd'hui possible de vérifier automatiquement, une fois le logiciel terminé, que les tâches prévues dans le cahier des charges sont bien réalisables par l'utilisateur. Ainsi, la recette d'un logiciel important peut-elle s'avérer fastidieuse. De même, il n'est pas aisé de vérifier que les modifications apportées lors d'une mise à jour n'ont pas altéré le dialogue homme-machine original, par exemple lors de tests de non-régression suite à une opération de maintenance. Des outils commerciaux comme *QC/Replay*™ ou *WinRunner*® apportent une solution de surface en permettant de rejouer des scripts au niveau Présentation de l'interface, mais ils ne se basent pas sur

Résumé dans : Jambon F. & Boisdron Y. Vers une validation du dialogue homme-machine pour les Applications Graphiques Interactives de Conception Technique. ***Onzième conférence sur l'Interaction Homme-Machine (IHM'99), Montpellier, France***, 22-26 novembre 1999. p. 189.

une analyse de la tâche de l'utilisateur. Ils ne valident ainsi qu'indirectement et partiellement le dialogue, on parle plutôt de vérification, car les scripts doivent être enregistrés un par un.

Face à ce constat, la motivation de notre travail est de disposer d'un formalisme adapté à la modélisation des tâches des utilisateurs, et d'outils permettant de vérifier a posteriori que le dialogue entre l'utilisateur et le système final respecte la tâche prévue dans le cahier des charges. Cet article s'intéresse tout d'abord à un exemple typique d'utilisation d'un système de CAO géométrique 2D, le dessin d'un cercle, puis il aborde notre modèle de tâche. Ensuite, la génération des vecteurs de test est détaillée. Enfin, le principe de validation du dialogue par l'utilisation d'un simulateur est exposé.

TÂCHE TYPIQUE EN CAO

Les tâches des AGI-CT utilisent de manière courante plusieurs objets du domaine comme paramètres. Ainsi, nous avons choisi d'utiliser comme exemple de travail la tâche, illustrée par la figure 1, qui consiste à «créer un cercle en utilisant comme centre le projeté du *Point_1* sur le *Segment*, et comme rayon la moitié de la distance entre le *Point_2* et l'extrémité du *Segment*». Afin de rendre l'interaction plus souple, la sélection du centre du cercle et de son rayon peuvent se faire dans n'importe quel ordre. En outre, le rayon peut être spécifié directement par un réel ou bien être un calcul de distance.

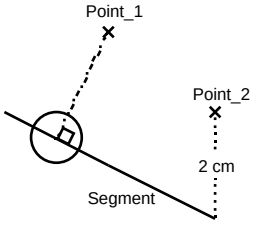

Figure 1 : Tâche de construction géométrique multi-objets.

PROPRIÉTÉS VÉRIFIÉES

Notre étude a pour but de vérifier deux des propriétés définies en Interaction Homme-Machine [1, 2]. Il s'agit de *l'atteignabilité* et de la *complétude de la tâche.* En outre, nous ne nous intéressons qu'aux actions de l'utilisateur sur le système, et non pas aux retours d'information vers l'utilisateur.

Atteignabilité («Reachability»)

L'atteignabilité est la «capacité du système à permettre à l'utilisateur de naviguer dans l'ensemble des états observables du système» [1]. Dans notre étude, nous voulons vérifier que le système accepte bien la suite des interactions de l'utilisateur, reflet des tâches prévues dans le cahier des charges.

Complétude de l'interaction («task completeness»)

La complétude de l'interaction est une propriété corollaire à l'atteignabilité. Nous voulons en effet vérifier que le système accepte non seulement les interactions prévues de l'utilisateur, mais aussi effectue les effets de bord attendu sur le Noyau Fonctionnel du système.

MODÈLE DE TÂCHES

Au cours de notre étude, nous nous sommes tout d'abord attachés à confronter les formalismes de modélisation de tâches actuels à des exemples d'interaction classiques dans des logiciels de CAO. Constatant que les formalismes disponibles étaient peu adaptés au cas particulier des AGI-CT, nous avons créé une grammaire de tâche simplifiée. Afin de faciliter le travail des ergonomes en amont, nous avons utilisé, pour sa représentation graphique, un sous-ensemble de MAD [7] qui sera traduit a posteriori dans notre grammaire. Le sous-ensemble de MAD utilisé limite les relations temporelles utilisables à la séquence (SEQ), l'alternative (OR), et l'ordre indépendant (AND) à l'exclusion de tout autre. La figure 2 illustre l'analyse de la tâche de création d'un cercle comme pourrait la faire un ergonome. Les rectangles symbolisent les commandes de l'utilisateur tandis que les textes en italique sont des désignations directes d'objets graphiques ou des entrées de données numériques.

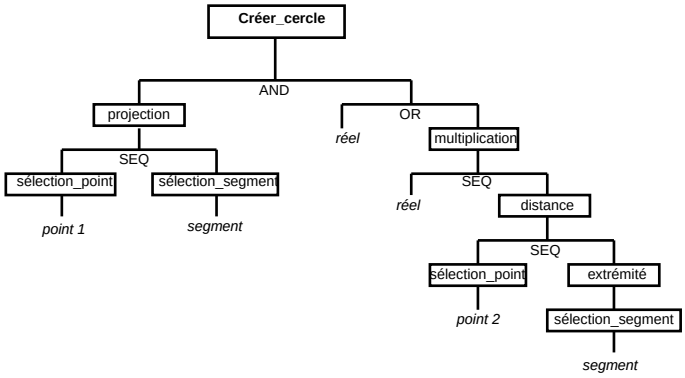

Figure 2 : description graphique de la tâche de création de cercle.

À l'heure actuelle, la transcription de la forme graphique à la grammaire s'effectue manuellement. Dans notre grammaire, chaque tâche ou sous-tâche est numérotée par identificateur unique, commence par les mots-clés *##tâche*, respectivement *#sous-tâche*, et se termine par *##fin tâche*, respectivement *#fin sous-tâche*. En outre, le corps de chaque tâche contient trois champs :

Le champ *commande* indique quelle commande est activée par l'utilisateur, par exemple l'item de menu utilisé. Ce champ est utilisé par le simulateur pour initier le dialogue.

Résumé dans : Jambon F. & Boisdron Y. Vers une validation du dialogue homme-machine pour les Applications Graphiques Interactives de Conception Technique. ***Onzième conférence sur l'Interaction Homme-Machine (IHM'99), Montpellier, France***, 22-26 novembre 1999. p. 189.

Le champ *résultat* indique le type d'objet du domaine que la tâche est supposée retourner, dans notre exemple c'est une entité *cercle*. Ce champ est utilisé par le simulateur pour vérifier que la tâche a donné le résultat escompté.

Le champ *paramètre* représente le dialogue structuré. Il indique quelles sont les entités dont a besoin la tâche pour arriver à son terme, et les moyens de l'obtenir. Il se compose de trois types d'information :

- Le type d'objet requis, par exemple une position ou un réel.
- Le moyen d'obtenir cet objet, ce peut être une sous-tâche, dans ce cas elle est indiquée par son numéro entre crochets, par exemple <11>, ou une tâche atomique, elle est alors directement indiquée, une_position ou un_réel par exemple. Bien entendu, le type d'objet retourné par une sous-tâche (champ *résultat*) doit être identique au type d'objet attendu.
- Un ou plusieurs opérateurs temporels, ils sont équivalents à ceux de MAD. Dans notre grammaire, la séquence est l'opérateur par défaut, l'opérateur AND est représenté par le symbole «&» et l'opérateur OR par le symbole «/».

```
## tache 1

       commande : creer_cercle
       resultat : cercle
       parametre : position=<11>
       & parametre : reel=un_reel / <13>

       # sous-tache 11
              commande : projection
              resultat : position
              parametre : point=<111>
              parametre : segment=<112>
       # fin sous-tache

       # sous-tache 13
              commande : multiplication
              resultat : réel
              parametre : reel=un_reel
              parametre : reel=<12>
       # fin sous-tache

       # sous-tache 12
              commande : distance
              resultat : réel
              parametre : point=<121>
              parametre : position=<122>
       # fin sous-tache

       # sous-tache 122
              commande : extremite
              resultat : position
              parametre: segment=<1221>
       # fin sous-tache

       …

## fin tache
```

Figure 3 : extrait de la description textuelle de la tâche de création de cercle.

GÉNÉRATION DES VECTEURS DE TEST

À partir de la modélisation des tâches de l'utilisateur selon notre grammaire, nous générons automatiquement des séquences possibles d'interaction avec le système, appelées *vecteurs de test*. Ces séquences représentent toutes les suites d'actions possibles de l'utilisateur sur le système aboutissant à réalisation d'un but de l'utilisateur, dans notre exemple «créer un cercle». Notre système, en utilisant un parcours d'arbre classique, génère toutes les séquences d'interactions possibles, et assure ainsi une couverture totale.

Les vecteurs de test sont générés avec une syntaxe proche de celle des tâches. Chaque vecteur commence par le mot-clé *VECTEUR*, se termine par *FIN_VECTEUR*, et contient deux types d'actions : les commandes qui correspondent aux items de menu sélectionnés par l'utilisateur, et les entrées de données ou les sélections d'objets. Un des vecteurs généré est reproduit figure 4.

```
VECTEUR
       commande : creer_cercle
       commande : projection
       commande : designer_point
       une_position
       commande : designer_segment
       une_position
       commande : multiplication
       un_reel
       commande : distance
       commande : designer_point
       une_position
       commande : extremite
       commande : designer_segment
       une_position
FIN_VECTEUR
```

Figure 4 : un *vecteur de test* permettant la création d'un cercle.

SIMULATION DU DIALOGUE

L'application testée est un prototype de logiciel de CAO, permettant la création d'objets géométriques 2D [5]. Ce système permet en outre l'utilisation des tâches structurées multi-objets. Il a été réalisé selon une architecture logicielle, H^4 [4], adapté à ce type d'applications. Notre simulateur est une version "épurée" du logiciel réel, au sens où les appels à l'Adaptateur de Domaine de l'application (le modèle géométrique) ont été vidés de leur contenu. De même, toutes les interactions de l'utilisateur vers le système ont été remplacées par un module qui imite ces interactions à partir des *vecteurs de test*.

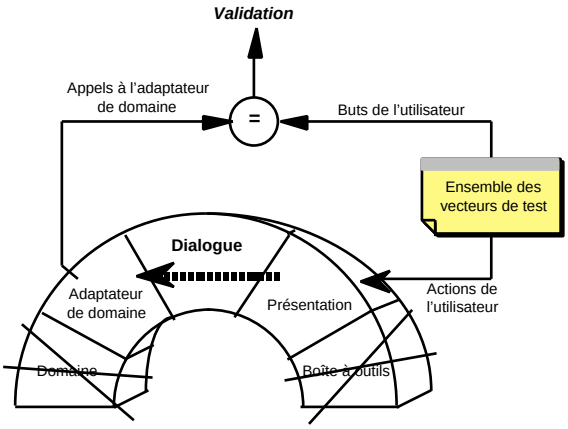

Figure 5 : principe de fonctionnement du simulateur.

Résumé dans : Jambon F. & Boisdron Y. Vers une validation du dialogue homme-machine pour les Applications Graphiques Interactives de Conception Technique. ***Onzième conférence sur l'Interaction Homme-Machine (IHM'99), Montpellier, France***, 22-26 novembre 1999. p. 189.

Ainsi, le Contrôleur de Dialogue du logiciel réel est sollicité comme si un utilisateur exécutait réellement la séquence spécifiée et s'il faisait appel à un véritable modèle géométrique. À l'issue de chaque *vecteur de test*, le simulateur vérifie que le noyau fonctionnel de l'application a bien reçu la commande attendue dans le modèle géométrique, pour notre exemple c'est la création d'une entité cercle (Cf. fig. 5).

Pour pouvoir effectuer la correspondance entre les buts de l'utilisateur et les appels à l'adaptateur de domaine, il est nécessaire de disposer d'une table de correspondance. Cette table symbolise la collaboration entre ergonomes et programmeurs. En effet, elle fait le lien entre les commandes de l'utilisateur issues de l'analyse de tâche (en partie gauche figure 6) et les méthodes de l'Adaptateur de Domaine (en partie droite figure 6).

```
commande : creer_cercle = COMMANDE :
       creation.cercle
commande : creer_rectangle = COMMANDE :
       creation.rectangle

un_reel : = REEL :
une_position : = POSITION :
un_segment : = SEGMENT :
un_point : = POINT :
un_cercle : = CERCLE :
un_rectangle : = RECTANGLE :
```

Figure 6 : Extrait de la table de correspondance.

RÉSULTATS DE LA SIMULATION

Actuellement, nous avons identifié trois principaux types d'erreurs générés par le système :

- La commande appelée n'est pas implémentée dans l'API de l'Adaptateur de Domaine ;
- Le type d'objet retourné par l'Adaptateur de Domaine n'est pas celui attendu ;
- Aucune commande de l'Adaptateur de Domaine n'est appelée à l'issue d'une interaction.

Les résultats obtenus à l'issu de la simulation sont du type «Pass/Fail», c'est-à-dire que le concepteur peut s'assurer que tout va bien, ou bien qu'il y a au moins un problème. L'identification du problème est très liée à l'architecture H^4, et est pour cette raison de la compétence du seul programmeur, même si le problème rencontré provient de l'analyse de la tâche, domaine de l'ergonome.

CONCLUSION & PERSPECTIVES

Cette étude préliminaire sur la validation du dialogue homme-machine s'intègre dans une perspective de recherche plus vaste visant à promouvoir les principes de la validation des IHM en complément de la vérification [3]. Elle montre qu'il est possible de vérifier automatiquement et directement sur le système final la conformité entre une analyse de la tâche d'un utilisateur et son implémentation dans le système final. Cette validation présente deux intérêts majeurs : tout d'abord elle s'effectue entre les deux extrémités du cycle de conception du logiciel, d'un côté le cahier des charges, et de l'autre le logiciel final. En outre, elle peut s'effectuer a posteriori sur un logiciel déjà opérationnel dont on ne modifie qu'une faible partie du code exécutable.

Cependant cette validation présente des limitations dont nous désirons nous affranchir. La validation est actuellement limitée à un modèle de tâche très restrictif, ne comprenant que la séquence, l'alternative, et l'ordre indépendant. De plus, il impose une correspondance biunivoque entre les buts de l'utilisateur et l'API de l'Adaptateur de Domaine. Nous souhaitons dans un premier temps étendre ce modèle. En outre, le processus de validation impose une modification de deux modules de l'architecture logicielle H^4 dont nous sommes dépendants. C'est pourquoi, nous souhaitons dans un second temps nous affranchir de ce modèle pour étendre le processus de validation à tout type d'architecture logicielle.

BIBLIOGRAPHIE

1. Coutaz, J. et Nigay, L. *Cours d'Interfaces Homme-Machine*. Université Joseph Fourier, 1998.
2. Dix, A., Finlay, J., Abowd, G. et Beale, R. *Human-Computer Interaction*. Prentice Hall, 1998.
3. Fields, B., Merriam, N. et Dearden, A. DMVIS: Design, Modelling and Validation of Interactive Systems. In *Proceedings of Eurographics Workshop on Design, Specification, Verification of Interactive Systems* (June 4-6, Granada, Spain), Eurographics, Springer-Verlag, 1997, pp. 29-44.
4. Guittet, L. *Contribution à l'Ingéniérie des Interfaces Homme-Machine - Théorie des Interacteurs et Architecture H4 dans le système NODAOO.* Thèse de Doctorat : Université de Poitiers, 1995.
5. Patry, G. *Contribution a la conception du dialogue Homme Machine dans les applications graphiques interactives de conception technique : le systeme GIPSE*. Thèse de Doctorat : Université de Poitiers, 1999.
6. Pierra, G. Towards a taxonomy for interactive graphics systems. In *Proceedings of Eurographics Workshop on Design, Specification, Verification of Interactive Systems* (June 7-9, Bonas), Springer-Verlag, 1995, pp. 362-370.
7. Scapin, D.L. et Pierret-Golbreich, C. MAD : Une méthode analytique de description des tâches. In *Proceedings of Colloque sur l'ingénierie des Interfaces Homme-Machine (IHM'89)* (Mai, Sophia-Antipolis, France), 1989, pp. 131-148.