Conception du dialogue d'une application interactive

Sybille Caffiau[1,2]

[1]LISI
Téléport 2 – 1 avenue Clément Ader
86961 Futuroscope Cedex, France
sybille.caffiau@ensma.fr

[2]INRIA
Domaine de Voluceau – Rocquencourt – B.P. 105
78153, Le Chesnay, France

RESUME
Le dialogue est une partie cruciale dans les logiciels interactifs. Concevoir cette partie nécessite du temps et est source d'erreurs. En conséquence, l'automatisation de la conception du dialogue est d'un grand intérêt pour l'industrie. Cependant, la génération du dialogue ne peut pas être complètement prise en charge par les outils existants. Le but de l'étude présentée dans cet article est d'identifier et d'analyser les difficultés d'une telle automatisation. Pour cela, nous avons dans un premier temps étudié les approches de génération d'interfaces (et par conséquent du dialogue). Puis, nous avons réalisé différentes études de cas afin de définir quels étaient les choix faits lors de la conception et qui imposent des transformations qui ne sont pas triviales. Cet article présente les conclusions de ces différentes études.

MOTS CLES : Modèles et formalismes, méthodes de conception, processus de développement centré-utilisateur/génie logiciel, modèles de tâches.

ABSTRACT
The dialog is a crucial part of interactive software. Designing this part is time-consuming and error-prone. As a consequence, the automation of the dialog part design is of major interest for the software industry. However the complete generation of the dialog can not be achieve with existing tools. The aim of the study presented in this paper is to identify and analyze the difficulties of such automation. Our results are first based on a study we have conducted on different approaches proposed to generate user interfaces (and thus the dialog). They are also based on case studies analysis aiming at highlighting choices during the dialog design, which impose non-trivial transformations. This paper exposes conclusions of these studies.

CATEGORIES AND SUBJECT DESCRIPTORS: H.5.2 User Interface : Theory and Method I.6.4 Model Validation and Analysis

GENERAL TERMS: Design, reliability, languages, theory, verification.

KEYWORDS: Models and formalisms, conception methods, user centered design/software engineering, task models.

INTRODUCTION
La conception et le développement des interfaces constituent une part importante, en temps et en nombre de lignes de code, de la réalisation d'une application interactive. C'est pourquoi des recherches sont menées afin de réaliser des interfaces plus rapidement et plus conformes aux besoins des utilisateurs finaux. Afin de recueillir ces besoins et de les formaliser, un certain nombre de formalismes de modèles de tâches, ont été proposés [6]. Malheureusement, ils ne sont que très rarement utilisés dans les outils de modélisation et de développement des applications interactives bien qu'ils soient nécessaires dans les phases de conception et de validation de ces applications. Un moyen d'intégrer ces modèles de tâches lors du développement du logiciel est de concevoir les applications à partir de ces modèles [6]. Certains systèmes basés sur modèles (model-based systems ou MBS) génèrent ou semi-génèrent des interfaces à partir d'un modèle de tâches. Ces approches ont certaines limitations comme le type d'interaction des interfaces ou la prise en compte des erreurs de l'utilisateur.

Les outils de simulation des éditeurs de modèles de tâches (comme CTTE[1] ou K-MADe[2]) semblent être proches d'une partie des interfaces ; le dialogue. Cependant, la conception du dialogue nécessite le respect de contraintes (conséquences des techniques d'interaction utilisée par exemple). Les modèles de tâches n'étant pas conçus pour prendre en compte ces données, le concepteur est amené à en tenir compte et à faire des choix pour les respecter.

[1] http://giove.cnuce.cnr.it/ctte.html
[2] http://www-rocq.inria.fr/merlin/kmade/

Nous proposons dans cet article une étude de la génération du dialogue des applications interactives. Nous la présentons en trois parties. Nous allons dans une première partie présenter les approches de génération d'interfaces et en donner quelques limitations. La seconde partie est consacrée au dialogue, nous le définirons et indiquerons quelques formalismes qui permettent de le modéliser. Enfin, avant de conclure, nous indiquerons quelques-unes des contraintes de conception du dialogue.

LES APPROCHES GENERATIVES

Les systèmes basés sur modèles (model-based systems ou MBS) réalisent des interfaces à partir de la spécification de modèles. Cependant, l'utilisation de ces systèmes impose souvent l'apprentissage de langages spécifiques et est réservée à des spécialistes en informatique. De plus, le développement des plateformes d'accueil a rendu nécessaire la conception d'interfaces adaptables au contexte d'utilisation [4].

Afin de concevoir de telles interfaces et de permettre la génération par les utilisateurs, des approches cherchent à générer des interfaces uniquement à partir de la spécification des besoins de l'utilisateur (spécifiés par un modèle de tâches). C'est cette approche qui est suivie par les outils ARTStudio [16], TERESA [10] et le framework Dygimes [9]. Après quelques étapes, ils produisent des interfaces spécifiques pour chaque plateforme. Pendant ces étapes, le modèle de tâches initial est soit complété, soit exploité pour produire d'autres modèles. Des informations sont ajoutées pour s'adapter aux contraintes des plateformes ou pour insérer d'autres informations. Ces ajouts peuvent modifier le modèle de tâche initial. D'autres parties de l'interface sont obtenues par l'exploitation des éléments des modèles de tâches, c'est le cas du modèle de dialogue [15]. Dans TERESA et Dygimes, le dialogue est centré sur l'enchaînement de fenêtres qui est induit directement des opérateurs du modèle de tâches. TERESA permet des optimisations grâce à l'utilisation d'heuristiques. De plus, l'utilisateur peut intervenir pendant le procédé de génération. Ces deux points peuvent avoir pour conséquences la modification du modèle de tâches initial alors que la conception d'une interface se fait de manière itérative. Enfin, toutes ces approches sont adaptées à la conception d'interfaces utilisant des techniques d'interaction de type WIMP. Cependant, les applications développées aujourd'hui ne se limitent plus aux techniques d'interaction de ce type. En plus des techniques traditionnelles elles doivent prendre en compte des techniques telles que la manipulation directe [14]. Suivre ces principes impose que les actions des utilisateurs et les réponses du système soient très liées et que les intermédiaires, comme les boîtes de dialogues, soient supprimés. De ce fait, l'ordre d'exécution des tâches peut être modifié en fonction de l'interaction choisie ou des instruments utilisés supprimant ou ajoutant des tâches comme l'illustre l'exemple dans [2]. Cependant, les modèles de tâches ne comportent pas d'informations sur les techniques d'interaction utilisées pour réaliser une tâche. C'est pourquoi générer de telles applications est impossible.

LES MODELES DE DIALOGUE

Des modèles ont été développés pour faciliter la conception des applications interactives. L'un de ces modèles est le modèle de dialogue. Il permet de spécifier la dynamique entre l'homme et la machine. En référence au modèle de Seeheim [13], les modèles de dialogue sont décrits dans [7] comme étant liés : d'une part avec la sémantique du système interactif pour que celui-ci sache ce qu'il doit faire, d'autre part avec la présentation pour donner la visualisation de ce système.

Le dialogue des applications réalisées avec TERESA ou Dygimes framework est centré sur la gestion de la succession des fenêtres. Dans chaque fenêtre, des widgets sont associés aux tâches, leur exécution permet le passage à la fenêtre suivante. Le dialogue n'est défini qu'une fois les fenêtres constituées. Le dialogue interne d'une fenêtre n'est pas déduit par ces approches génératrices. Les interfaces, le plus souvent, ne sont pas réalisées à l'aide de ces systèmes basés sur modèles mais codées directement. Il est alors à la charge du concepteur d'utiliser les modèles de dialogue adapté à l'application qu'il développe. Un nombre important de formalismes ont été étudiés [11]. Parmi eux, la catégorie de formalisme majoritairement utilisée est celle des formalismes à base d'états. Ce sont des formalismes basés sur des modèles mathématiques qui permettent le raisonnement, et donc, la validation et la vérification de propriétés du dialogue dans les IHM telles que les propriétés de sûreté ou de vivacité [1]. Parmi les formalismes à base d'états, quatre permettent d'être utilisés directement pour être intégrés dans l'implémentation : les machines à états (ou automates) [8], les interacteurs hiérarchisés [5], les Hierarchical State Machine (HSM) [3] et les Interactive Cooperative Objects (ICO) [12].

Les machines à états sont le premier formalisme qui a servi à la description d'une application. Elles permettent de décrire le dialogue sous la forme d'états (les différents états du système) et de transitions qui permettent le passage de l'un à l'autre. Une condition booléenne, nommée garde, peut être associée à une transition. Il faut alors que la garde ait pour valeur « vrai » pour que l'action de la transition puisse être réalisée. Les transitions sont déclenchées par les événements produits par l'utilisateur. Cependant, ce formalisme produit des dialogues avec un grand nombre d'états pour des applications de grande taille. Afin de factoriser ces états, des formalismes permettant la hiérarchisation des états ont été créés, par exemple les interacteurs

hiérarchisés ou les Hierarchical State Machine. Un autre formalisme à état, celui-ci orienté-objet, les ICOs, permet une description formelle du comportement dynamique des systèmes interactifs. Il repose sur les réseaux de Petri. Comme les machines à états, il permet la vérification de propriétés et une représentation graphique du dialogue.

CONTRAINTES DE CONCEPTION DU DIALOGUE

Lors de la conception du dialogue d'applications interactives, certaines contraintes doivent être prises en compte. Nous présentons dans la suite celles qui sont liées au respect des propriétés des interfaces et celles qui sont liées aux techniques d'interactions choisies. En plus de ces deux types de contraintes, les choix faits sur les méthodes de conception (comme choisir un formalisme à objets ou non pour modéliser le dialogue) ont également des conséquences sur la conception. Afin d'illustrer ce dernier type de contrainte, nous indiquerons dans une dernière partie quelques-unes de celles qu'impliquent le choix d'un type de conception et un formalisme de dialogue particuliers. Une méthode de conception très souvent utilisée en informatique, est la conception descendante. De plus, comme nous l'avons précisé précédemment, le formalisme des machines à états est un formalisme très largement utilisé. C'est pourquoi, nous présenterons les conséquences d'une conception descendante du dialogue formalisé par des machines à états.

Les Propriétés des Interfaces

Lorsque le dialogue est modélisé par un formalisme à base d'états, le concepteur essaie de minimiser le nombre d'états obtenus. Pour cela, les états sont regroupés et des gardes sont définies sur les transitions. Cependant, certains états doivent rester indépendants (ne sont pas regroupables), par exemple pour prévenir les erreurs de l'utilisateur. Lorsque l'utilisateur ne peut déclencher l'exécution d'une action qu'à certaines conditions, il ne faut pas qu'il puisse utiliser la commande correspondant avant. Par exemple, dans le cas d'un bouton utilisé pour valider, si on définit une garde pour veiller à ce que le traitement de la validation ne soit exécuté que lorsque les informations nécessaires sont entrées, cela signifie que le déclenchement de la transition est autorisé à tout moment mais que son exécution est soumise à vérification. De ce fait, le bouton reste actif et ne déclenche l'action que lorsque la garde a pour valeur « vrai ». Les états ne sont donc pas tous de même nature, il faut donc être capable de distinguer les états pouvant être regroupés des autres et déterminer ce qui les différencie. De plus, l'utilisateur doit avoir la possibilité d'annuler certaines de ces actions, par exemple la saisie d'une donnée. Les transitions correspondantes sont alors liées, l'une ayant pour raison d'être le retour à l'état précédant l'exécution de l'autre. Elles doivent donc être définies en même temps. Cependant, toutes les actions (transitions) ne sont pas annulables, certaines ont un effet non réversible comme l'envoi d'un email par exemple.

Les Interactions

Le dialogue des applications interactives doit évidemment prendre en compte les interactions utilisées. Le passage d'un dialogue logique à un dialogue concret (dialogue prenant en compte les interactions) entraîne quelques modifications. Lorsque l'application contient des objets interactifs, ceux-ci sont sources d'interaction mais qu'à certaines conditions. Ils peuvent donc passer de l'état d' « objets d'interaction » à celui d' « éléments de visualisation ». Cependant, ce passage n'est pas présenté dans un dialogue formalisé sans objet. Il en est de même lors de la création de tels objets. Certaines actions peuvent être déclenchées de plusieurs manières différentes (par exemple : clic et raccourci clavier). Cependant, il est impossible de factoriser ces transitions, car bien qu'elles réalisent la même action, les événements qui les déclenchent peuvent être produits par des instruments différents qui peuvent ne pas être disponibles au même moment. Il est donc indispensable de pouvoir les différencier afin de s'assurer que lorsque l'utilisateur peut exécuter une action, au moins un des instruments est disponible. L'exécution d'une transition étant liée à l'événement qui la déclenche (et donc à l'instrument qui produit l'événement), il peut arriver qu'une transition ne soit pas exécutable dans un état donné.

Choix de Conception

TERESA et Dygimes framework réalisent les applications (et donc le dialogue) avec cette même approche : l'approche descendante, cependant ils ne réalisent le dialogue qu'à un niveau très élevé (succession de fenêtres). Dans cette étude, nous prenons en compte la conception de dialogue de plus bas niveau, afin d'identifier les problèmes spécifiques engendrés. La conception du dialogue par raffinements successifs impose des modifications du modèle de dialogue d'un niveau d'abstraction au modèle de dialogue du niveau inférieur afin d'intégrer les besoins de chaque niveau d'abstraction. Ces modifications rendent difficile le retour au(x) niveau(x) supérieur(s) et ce, même lorsque la même notation est utilisée à tous les niveaux d'abstraction. Il est donc difficile d'assurer qu'une machine à états d'un niveau donné, soit équivalente à celui du niveau supérieur. De ce fait, rien ne permet d'assurer que des propriétés vérifiées à un niveau d'abstraction soient toujours vraies au(x) niveau(x) inférieur(s). Pour en avoir l'assurance, il est nécessaire de définir les transformations provoquées par le raffinement, de manière formelle.

De plus, lors des différentes étapes de raffinement, des opérations ont été réalisées sur les états et les transitions (regroupement d'états, fusion de transitions, éclatement de transitions…). Ces opérations ont été utili-

sées pour répondre à un besoin et en fonction du contexte (par exemple dans le cas du regroupement d'états). Ces opérations doivent être analysées dans une perspective de définition de règles de généralisation. Enfin, le formalisme des machines à états, s'il prend en compte les entrées des utilisateurs (par les événements des transitions), ne les associe pas avec les éléments dont ils dépendent. Ainsi, certaines transitions partant d'un état peuvent ne pas être déclenchables du fait que l'élément dont ils dépendent est inaccessible.

CONCLUSION

Concevoir le dialogue d'applications interactives à partir du modèle de tâches n'est pas automatique. Elle implique l'ajout d'information à prendre en compte pour intégrer l'interaction, les propriétés des interfaces. De plus, les choix faits lors de la conception ont eux aussi des conséquences sur les modifications appliquées lors de la conception. Nous avons étudié une formalisation à l'aide de machines à états, celle-ci engendre un certain nombre de limites et induit certaines solution. C'est pourquoi nous allons étudier d'autres formalismes de dialogue, en particulier l'utilisation d'un formalisme à objets : les ICO. Nous pourrons alors comparer la conception à l'aide de ces deux types de formalisme. Les dialogues déduits de modèles de tâches sont obtenus en s'appuyant principalement sur les opérateurs temporels, nous voulons étudier les autres données du modèle de tâches afin de déterminer celles qui peuvent être utilisées pour définir le dialogue. Cependant, du fait que la conception d'une application interactive est le plus souvent itérative, nous cherchons à étudier le lien existant entre les modèles de tâches et les systèmes interactifs (et en particulier le dialogue) afin de le conserver pour l'utiliser pendant sa conception et sa validation.

Enfin, cette étude est la première étape d'une étude plus générale portant sur la conception d'une application à partir du modèle de tâches. C'est pourquoi, nous étudions également les données pouvant être utilisées lors de la conception d'autres parties d'une l'application (comme la partie fonctionnelle).

Directeur de thèse : Patrick GIRARD (LISI/ENSMA)
Co-directeur de thèse : Dominique SCAPIN (INRIA)
Début de la thèse : Octobre 2006.

BIBLIOGRAPHIE

1. Aït-Ameur, Y., Aït-Sadoune, I., and Baron, M. 2005. Modélisation et Validation formelles d'IHM : LOT 1 (LISI/ENSMA), pp. 73. Délivrable pour le projet RNRT-VERBATIM.
2. Beaudouin-Lafon, M. 2000. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *CHI*, The Hague, Netherlands.
3. Blanch, R. 2005. *Architecture logicielle et outils pour les interfaces hommes-machines graphiques avancées*, (Doctorat d'Université (PhD Thesis)), Paris XI, Orsay.
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, L., Florins, M., and Vanderdonckt, J. 2002. "Plasticity of User Interfaces: A Revised Reference Framework", In *Tamodia*. Bucarest.
5. Depaulis, F., Jambon, F., Girard, P., and Guittet, L. 2006. Le modèle d'architecture logicielle H4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. *Revue d'Interaction Homme-Machine* 7(1): 93-129.
6. Diaper, D., and Stanton, N. A. 2004. *The Handbook of Task Analysis for Human Computer Interaction*. Lawrence Erlbaum Associates.
7. Dix, A., Finlay, J., Abowd, G., and Beale, R. 1993. *Human-Computer Interaction*. Prentice Hall.
8. Jacob, R. J. K. Using Formal Specification in the Design of Human-Computer Interface. Human Factors in computing systems, 1982, pp. 315-321.
9. Luyten, K., Clerckx, T., Coninx, K., and Vanderdonckt, J. Derivation of a Dialog Model from a Task Model by Activity Chain Extraction. DSV-IS'2003, Funchal, Madeira Island, Portugal, 2003.
10. Mori, G., Paternò, F., and Santoro, C. Tool Support for Designing Nomadic Applications. Intelligent User Interfaces (IUI'2003), Miami, Floride, 2003, pp. 141-148.
11. Olsen, D. R. 1992. *User Interface Management Systems: Models and Algorithms*. Morgan Kaufmann Publisher, San Mateo (CA), USA.
12. Palanque, P. 1992. *Modélisation par Objets Coopératifs Interactifs d'interfaces homme-machine dirigées par l'utilisateur*, (Doctorat d'Université (PhD Thesis)), Université de Toulouse I, Toulouse.
13. Pfaff, G. E. 1985. User Interface Management Systems, Proceedings of the Workshop on User Interface Management Systems held in Seeheim. In *Eurographic Seminars*. Springer-Verlag, Berlin.
14. Shneiderman, B. 1983. Direct Manipulation: a Step beyond Programming Languages. *IEEE Computer* 16: 57-69.
15. Tarby, J.C., *Gestion automatique du dialogue Homme-Machine à partir de spécifications conceptuelles*. Thèse de doctorat, Université Toulouse I - France, Septembre 1993.
16. Thévenin, D. 2001. Adaptation en Interaction Homme-Machine : le cas de la plasticité (Doctorat d'Université (PhD Thesis)), Université de Toulouse I, Toulouse, France.