Gestion automatique de l'aide contextuelle multimedia d'une IHM par exécution de sa spécification formelle

Ph. A. Palanque†, D. Salber*, R. Bastide†

† L.I.S., Université Toulouse I,
Place Anatole France,
31042 Toulouse Cedex, France
E-mail: {palanque,bastide}@cict.fr

* LGI-IMAG
Université Joseph Fourier - B.P. 53
38041 Grenoble Cedex 9, France
E-mail: daniel.salber@imag.fr

Résumé
L'aide contextuelle est une fonctionnalité particulièrement coûteuse à mettre en œuvre dans les applications interactives modernes. Nous montrons dans cet article comment le fait de disposer d'une spécification formelle embarquée de l'application permet de simplifier la mise en œuvre de l'aide, et d'envisager des fonctionnalités étendues. Nous proposons dans ce but d'enrichir la spécification de l'application par des annotations spécialement destinées à être exploitées par le moteur d'aide, lors de l'exécution. L'approche proposée permet au système d'aide de combiner de manière multimodale des informations multimédia spécifiées au niveau de ces annotations.

Mots clés
Système d'aide, aide contextuelle, spécification formelle, multimodalité, multimedia.

1. INTRODUCTION

La plupart des applications interactives commercialisées aujourd'hui permettent à l'utilisateur d'accéder à une aide en ligne. Deux types d'aide en ligne peuvent être identifiés : **l'aide générale**, qui n'est souvent qu'une version électronique, éventuellement hypertexte, du manuel utilisateur et **l'aide contextuelle** qui, en revanche, assiste l'utilisateur en tenant compte du contexte de l'interaction. En termes d'utilisabilité, ce dernier type se révèle plus pertinent puisqu'il offre directement à l'utilisateur des informations relatives à la tâche qu'il est en train de réaliser. Cependant, l'aide contextuelle est difficile à mettre en œuvre, en particulier lorsque elle porte sur le domaine de la tâche : le système d'aide doit non seulement avoir accès au contexte d'interaction, mais plus encore être organisé en fonction de celui-ci. Pour satisfaire ces contraintes, le système d'aide doit être conçu comme une mini-application interactive. L'objet principal de cet article est de montrer qu'une spécification formelle du type Objets Coopératifs Interactifs à base de réseaux de Petri [Palanque 92] permet d'automatiser de façon générique la gestion de l'aide contextuelle. Nous montrons d'abord que l'utilisation du système d'aide induit un renversement du paradigme de la communication homme-machine. Nous détaillons ensuite le mécanisme d'aide contextuelle fondé sur l'utilisation d'une spécification d'application interactive par Objets Coopératifs Interactifs. Puis nous envisageons enfin l'organisation du système d'aide dans le cas d'une aide contextuelle multimédia et multimodale.

Un système d'aide contextuelle peut fournir des informations à l'utilisateur aux différents niveaux classiques de la communication homme-machine : lexical, syntaxique et sémantique. Nous considérons que l'aide porte sur le niveau lexical lorsque son contenu ne concerne qu'un seul interacteur indépendamment du contexte dans lequel l'aide a été activée. Nous dirons que l'aide porte sur le niveau syntaxique lorsque son contenu concerne ou décrit des enchaînements d'interactions offerts à l'utilisateur (par exemple, une aide de cet ordre pour la commande coller indiquerait qu'il est nécessaire d'avoir au préalable effectué une sélection, puis d'avoir choisi soit la commande couper soit la commande copier). Nous caractérisons par le terme sémantique une aide portant sur les fonctionnalités même de l'application ou exprimée en terme d'activité des

utilisateurs au niveau de la tâche qu'ils réalisent. On peut remarquer qu'une aide au niveau lexical ou syntaxique inclut souvent des informations d'ordre sémantique.

L'aide au niveau lexical doit être prise en compte au niveau présentation de l'architecture de Seeheim. Son rôle est d'expliciter à tout instant les commandes accessibles à l'utilisateur. Cette aide peut prendre la forme d'un feedback lexical tel que le changement de la forme du pointeur de souris en fonction de la zone de l'interface où il se trouve, l'animation d'un bouton lorsqu'il est enfoncé, … Dans les UIMS de haut niveau [Bass 91] cette aide est le plus souvent prise en compte par la boîte à outils avec une intervention minimale de la part du programmeur. Une autre forme d'aide au niveau lexical est **l'aide contextuelle locale** comme par exemple les bulles d'aides du système 7 du Macintosh. Les messages d'aide sont décrits statiquement dans les ressources de l'application et le développeur doit écrire un message d'aide pour chaque état possible de chaque élément d'interaction. Par exemple, deux messages d'aide doivent être écrits pour les deux états possibles d'un article de menu : grisé ou non. Lorsque l'utilisateur est en mode aide et pointe la souris sur l'article de menu, le système affiche dans une bulle d'aide le message correspondant à l'état de celui-ci. Les inconvénients d'un tel système sont d'une part qu'il est fastidieux à écrire pour le développeur, d'autre part que ce type d'aide ne fonctionne que pour le plus bas niveau d'interaction : il n'est pas possible avec ce mécanisme de donner de l'aide contextuelle relative au domaine de la tâche.

L'aide au niveau syntaxique doit être prise en compte au niveau du composant dialogue. Son rôle est d'informer l'utilisateur sur les séquences de commandes licites. Cette aide est beaucoup plus difficile à mettre en œuvre car elle nécessite de disposer au moment de l'exécution d'un modèle du dialogue accessible par le système d'aide lui-même. En effet, les seules propositions disponibles pour ce genre d'aide sont celle pour lesquelles l'application interactive fonctionne en exécutant une spécification du dialogue [Sukaviriya 92], [Palanque 93].

L'aide au niveau sémantique doit être prise en compte au niveau du composant noyau fonctionnel. Son rôle est d'informer l'utilisateur sur la signification et les effets possibles de ses actions. Elle peut revêtir deux aspects différents : l'aide fonctionnelle et l'aide au niveau de la tâche. L'aide fonctionnelle décrit la sémantique de chacun des services offerts par l'application interactive, par exemple la signification des éléments de menu, des boutons, des dialogues, … L'aide au niveau de la tâche est généralement proposée en mettant à disposition un manuel en ligne organisé selon les buts possibles de l'utilisateur. Cette aide ne tient en général pas compte du contexte de l'interaction. La prise en compte de l'état de l'interaction dans ce type d'aide nécessite l'intégration dans le système d'aide d'un modèle de tâche couplé avec le modèle de dialogue. Pour avoir à tout instant le contexte de la tâche en cours de l'utilisateur, il est nécessaire de répercuter les évolutions de l'état du dialogue dans le modèle de la tâche. Dans le cadre du formalisme des Objets Coopératifs Interactifs (ICO pour Interactive Coperative Objects), nous donnons un modèle de dialogue décrivant les séquences de commandes licites, et un modèle de tâche pour chacune des tâches pour lesquelles on désire offrir une aide. Ces modèles et leurs interactions sont décrits par un seul et même formalisme : les réseaux de Petri de haut niveau. Le caractère formel des réseaux de Petri nous permet de vérifier que chacune des tâches modélisée correspond bien à une séquence de commande licite dans le modèle du dialogue.

Les considérations ci-dessus mettent en évidence les difficultés spécifiques de la mise en œuvre de l'aide contextuelle aux différents niveaux de l'interaction. Nous allons maintenant détailler le contenu que devrait offrir à l'utilisateur un système d'aide contextuelle.

L'aide contextuelle dans l'interaction homme-machine doit idéalement répondre à trois questions de base que peut formuler l'utilisateur : **"Quoi ?"**, **"Pourquoi pas ?"** et **"Comment ?".**

La question **"Quoi ?"** correspond à l'interrogation "Que puis-je faire à partir de maintenant ?". Dans les interfaces dirigées par l'utilisateur, l'écran présenté à l'utilisateur

doit refléter l'état interne de l'application. De plus, il doit être possible de déterminer visuellement quelles sont les actions possibles à partir du contexte courant. Cette caractéristique est généralement présente dans les logiciels et est obtenue en "grisant" ou en inactivant les widgets correspondant aux opérations non disponibles.

La question **"Pourquoi pas ?"** se pose à l'utilisateur dès qu'il désire déclencher une action pour laquelle l'interacteur associé est inactivable. Dans les logiciels actuels, non seulement la question reste sans réponse, mais elle ne peut même pas être posée. En effet, une action de l'utilisateur sur un widget "grisé" est simplement ignorée par le système de gestion d'interface.

La question **"Comment ?"** signifie "Comment rendre telle action disponible ?". La réponse à cette question devrait compléter naturellement la question "Pourquoi pas ?" en fournissant la séquence des commandes à déclencher dans le but de rendre activable (disponible pour l'utilisateur) la commande désirée.

Certains travaux ont déjà été réalisés dans le domaine de la génération de l'aide contextuelle parmi lesquels on peut citer [Sukaviriya 92], [Carroll 88] et [Aaronson 87]. Toutefois, il est à noter que ces travaux et en particulier ceux décrits dans [Sukaviriya 92] ne se fondent pas sur une spécification formelle de l'interaction, rendant de ce fait l'aide générée peu fiable. En effet, dans ce cas, le système n'est pas vérifiable, c'est à dire que son fonctionnement ne peut pas être garanti même après implantation. Pour pallier à ces problèmes, des techniques de validation (à base de tests) ont été utilisées mais ont aussi montré leurs limites dans le domaine des interfaces homme machine dirigées par l'utilisateur. Dans le cas de l'aide générique construite à partir de spécification non formelles, les erreurs seront répercutées au niveau de l'aide, diminuant d'autant la fiabilité du système dans son ensemble. Cependant, ces travaux fondés sur une représentation du dialogue à l'aide de pré- et de post-conditions ont le mérite de reposer sur un système de construction d'interfaces opérationnel tel UIDE [Gieskens 92].

2. COMMUNICATION HOMME MACHINE ET SYSTÈME D'AIDE

Si l'on compare les systèmes interactifs et les systèmes classiques "directifs", on note que l'utilisateur et le système jouent dans chacun des deux cas des rôles opposés. Dans un système interactif à manipulation directe, l'utilisateur a l'initiative de la communication : à un instant donné, un certain nombre de services du système sont disponibles et l'utilisateur a le choix d'activer n'importe lequel de ces services. Dans une application traditionnelle directive en revanche, le système va dérouler par exemple un certain nombre de pages de menus afin de faire entrer à l'utilisateur une commande et ses paramètres. En fait, c'est le paradigme même de la communication qui est envisagé de deux façons symétriques : dans l'application à manipulation directe, l'utilisateur effectue des requêtes vis-à-vis de l'application. Dans un dialogue directif, c'est le système qui requiert à l'utilisateur les informations manquantes à l'exécution d'une commande.

Ces deux types de communication peuvent être appréhendés dans le cadre du protocole client-serveur, qui est à la base de nombreux langages à objets : on peut assimiler l'utilisateur au client et le système au serveur dans le premier cas, et l'on a dans le second cas une relation symétrique avec un utilisateur qui joue un rôle de serveur en fournissant à l'application-client les informations nécessaires.

Lorsque l'utilisateur invoque l'aide du système, le sens de la communication est profondément modifié, et ceci dans les deux cas des applications directives et à manipulation directe (cf. Figure 1).

Dans les applications directives en effet, le système d'aide laisse une grande latitude décisionnelle à l'utilisateur et le dialogue avec le système d'aide n'est pas directif, au contraire de l'application. Au lieu d'être contraint à répondre aux demandes de l'application (choix dans des menus ou saisie de formulaires par exemple), l'utilisateur peut en général naviguer dans le système d'aide qui est souvent présenté sous la forme d'un hypertexte ou d'un "manuel électronique". En d'autres termes, dans le dialogue avec l'application, l'utilisateur est contraint de passer séquentiellement dans une suite d'états

du dialogue qui est définie par l'application. En revanche, dans le système d'aide d'une application directive, tous les états du dialogue d'aide sont toujours accessibles à l'utilisateur. Il y a donc renversement du paradigme de communication : d'un dialogue contraint avec l'application, on passe à un dialogue non-contraint avec le système d'aide.

Figure 1: Les deux types de communication entre l'homme et la machine dans les applications interactives. Dans le cas a) l'utilisateur initie la communication et la machine répond (c'est le cas dans les applications dirigées par l'utilisateur). Dans le cas b) le système initie la communication et l'utilisateur donne des informations à l'application (c'est le cas dans les applications directives). Par contre, lorsque l'on parle du système d'aide correspondant aux deux applications, la figure a) correspond au type de communication dirigé par l'utilisateur dans les systèmes d'aide offerts par les application directives alors que la figure b) correspond au type de communication directif offert par les systèmes d'aide "évolués" offerts par les applications dirigées par l'utilisateur

Dans le cas des applications interactives à manipulation directe, on peut également observer un renversement similaire du paradigme de communication lorsque l'utilisateur invoque l'aide du système. Dans l'interaction avec l'application, l'utilisateur a le contrôle du dialogue et peut à partir de l'état courant atteindre quasiment n'importe quel autre état de l'application. En revanche, dans le système d'aide, le système "prend en charge" l'utilisateur et, en le guidant pas à pas, requiert de l'utilisateur les paramètres manquants à l'exécution des commandes. Le dialogue devient alors directif, et l'utilisateur n'a pas d'autre choix que de passer successivement par tous les états imposés par le système d'aide. Les exemples les plus représentatifs de ces systèmes d'aide sont les "assistants" que l'on trouve maintenant couramment dans certaines applications. Le "Chart Assistant" d'Excel ou le "Print Merge Helper" de Word en sont des exemples typiques.

Que ce soit dans des applications directives ou des applications à manipulation directe, l'invocation du système d'aide induit donc un renversement du sens de la communication homme-machine : d'un dialogue contraint ou libre, on passe généralement avec l'aide à un dialogue de type opposé. Cette constatation a des implications importantes sur la réalisation de l'aide, en particulier dans le cas d'une aide multimédia ou multimodale. Dans la section 3 nous montrerons comment l'aide peut être construite à partir de la spécification formelle de l'application. Cette aide peut être exploitée en multimodal et multimédia comme cela sera montré dans la section 4.

3. STRUCTURE DU SYSTÈME D'AIDE

Le système d'aide que nous proposons dans cet article est mis en œuvre en exploitant une spécification de l'application interactive fondée sur le formalisme des ICO [Palanque 92]. Ce formalisme s'attache à modéliser le dialogue homme-machine par des réseaux de Petri de haut niveau structurés suivant l'approche objet. Chaque fenêtre de l'application donne lieu à la définition d'une classe d'ICO. Cette classe contient quatre composants : les attributs, les services, la structure de contrôle et la présentation. Les attributs et les services sont définis comme dans un langage à objets traditionnel. La présentation est un ensemble d'interacteurs directement manipulables par l'utilisateur. La structure de

contrôle (ObCS pour Object Control Structure) est un réseau de Petri de haut niveau décrivant en fonction de l'état de l'application quelles sont à chaque instant les services exécutables. La présentation est mise en relation avec la structure de contrôle par une fonction d'activation qui associe à chaque service un ensemble de couples (interacteur, action sur l'interacteur) définissant quel service doit être exécuté lorsque l'utilisateur agit sur un élément donné de l'interface.

Le réseau de Petri permet d'avoir une vision synthétique à la fois de l'ensemble des séquences de commandes disponibles et de l'ensemble des états accessibles de l'interface. Les objectifs de cette description sont les mêmes que ceux de la technique par précondition et post-condition proposée dans UIDE [Gieskens 92] à savoir disposer d'un modèle embarqué du dialogue qui soit directement exploité par le moteur de l'application pour définir lors de l'exécution le comportement de l'interface. Par rapport à l'approche de UIDE les principaux avantages de l'utilisation des réseaux de Petri au lieu des Pré et Post conditions sont les suivants :

- la possibilité d'utiliser les fondements mathématiques des réseaux de Petri pour vérifier le modèle avant sa mise en œuvre ;
- la description aisée du parallélisme et de la synchronisation ;
- la prise en compte des aspects temporels, dont l'importance s'accroît dans les interfaces multimodales [Nigay 94] ;
- l'existence de mécanismes de structuration soit hiérarchique (macro places, macro transitions) soit orienté objets (classification, héritage, communication de type client-serveur).

Un des avantages principaux de disposer d'un modèle de dialogue embarqué est que ce modèle peut être exploité non seulement pour l'exécution de l'application mais aussi pour proposer une aide contextuelle à l'utilisateur. Nous avons déjà proposé ce type d'approche dans [Palanque 93] où nous proposions de générer automatiquement l'aide contextuelle à partir du modèle de dialogue. L'inconvénient de cette approche est que l'aide générée exploite forcément les termes présents dans le modèle de dialogue. De ce fait, l'aide n'est exploitable par l'utilisateur que dans le cas où les termes du modèle de dialogue sont les mêmes que ceux utilisés par l'utilisateur. Pour palier à cet inconvénient, nous proposons une extension de l'approche ci-dessus visant à annoter le modèle du dialogue par des indications exploitées exclusivement par le moteur d'aide.

Cette extension permet d'améliorer la réponse fournie par le moteur d'aide pour répondre aux questions "Pourquoi pas ?" et "Comment ?".

En ce qui concerne la réponse à la questions “Quoi ?” l'ajout d'annotations n'apporte rien de plus car la réponse à cette question consiste simplement à montrer comme étant actifs ou inactifs les différents interacteurs. Ceci est calculable directement à partir du marquage du réseau et de la fonction d'activation (cf. [Palanque 93]).

La réponse à la question “Pourquoi pas ?” est calculée en fonction de l'état courant de l'application. Dans ce cas il faut mettre des annotations d'aide sur les places du réseaux car c'est la répartition des jetons dans ces places qui modélise l'état de l'application.

Dans le cas de la réponse à la question "comment ?" au contraire l'aide est fournie en termes de commandes à effectuer par l'utilisateur et les annotations doivent donc porter sur les transitions du réseau qui correspondent à ces commandes.

Dans le cas général, la construction d'un système d'aide contextuel est une tâche difficile puisqu'il est nécessaire d'avoir à disposition à la fois l'espace d'état de l'application et les différentes séquences de commandes licite à chaque instant. Dans le cas où l'application est conçue par ICO, la tâche du concepteur du système d'aide se limite à annoter les différents composants (places et transitions) du réseau de chacun des ICO. La structure de ces réseaux, qui détermine à la fois l'espace d'état et les séquence de commandes de l'application, est conservée et sera exploitée au moment de l'appel au système d'aide pour générer l'aide contextuelle.

Dans la section suivante, nous allons présenter comment le moteur d'aide va exploiter

cette spécification pour fournir à l'utilisateur une aide contextuelle multimodale et multimédia.

4. PRODUCTION DE L'AIDE CONTEXTUELLE MULTIMODALE ET MULTIMÉDIA

Nous avons montré dans la partie 2 que l'invocation par l'utilisateur de l'aide de l'application provoque une inversion du dialogue homme-machine. Ce résultat nous conduit à penser que l'aide est un bon terrain d'expérimentations pour les interfaces multimodales en sortie. En effet, les interfaces multimodales ont à ce jour démontré leur potentiel principalement en entrée d'un système informatique, pour la communication de l'utilisateur vers le système. Lors de l'utilisation d'un système d'aide, l'information est principalement transmise du système vers l'utilisateur ; le système présente à l'utilisateur des informations sur sa manipulation et sur les services qu'il offre. De ce fait, on peut envisager de générer les informations d'aide de façon multimodale. Dans cette section nous montrons comment l'utilisation d'une description formelle sous forme de réseau de Petri alliée au mécanisme de génération de l'aide exposé dans la partie 3 permet d'automatiser la génération automatique d'aide contextuelle multimédia et multimodale pour répondre à la question "Comment ?" (présentée dans la première partie). Nous montrons aussi que l'aide ainsi générée fait apparaître un type de présentation original, la combinaison multimodale d'éléments multimédia.

La prochaine version du système du Macintosh d'Apple [Apple 94] comprendra un système d'aide contextuelle original : l'utilisateur peut demander de l'aide à tout moment et poser une question de type "Comment (réaliser telle tâche ?)". En réponse, le système présente pas à pas les sous-tâches et actions élémentaires que l'utilisateur doit effectuer, à la fois sous forme textuelle et en entourant d'un trait rouge les éléments de l'interface sur lesquels l'utilisateur doit agir. Il y a bien dans ce cas génération multimodale au sens de [Coutaz 93]. En effet, les informations (texte et dessin d'un trait rouge des éléments d'interaction) sont générées par le système et utilisent deux médias différents. Par média, nous entendons la combinaison d'un dispositif d'entrée-sortie (en l'occurrence, l'écran) et d'un langage d'interaction utilisant ce dispositif (en l'occurrence, l'affichage de texte et le dessin d'un trait rouge autour des éléments d'interaction) comme décrit dans [Nigay 94]. Cependant, cette aide contextuelle multimodale n'est pas générée automatiquement et est à la charge du programmeur.

Si l'application repose sur une spécification formelle sous forme de réseau de Petri, nous avons montré dans la partie 3 que l'aide contextuelle pouvait être générée en ajoutant au réseau de Petri décrivant le dialogue des annotations complémentaires dédiées à l'aide. Chacune des transitions du réseau correspond à une action élémentaire possible de l'utilisateur et est annotée par l'aide correspondant à cette transition. Ce mécanisme permet de donner à l'utilisateur de l'aide pour chacune des actions élémentaires. Pour répondre à une question "Comment ?", le moteur d'aide va déterminer une suite de transitions dans le réseau. L'aide correspondante sera générée en combinant en séquence les informations d'aide de chacune des transitions présentes dans la suite de transitions déterminée. Ce mécanisme de base doit cependant être affiné : pour fournir de l'aide qui soit véritablement contextuelle, chaque information d'aide doit être paramétrable. Prenons l'exemple d'une application permettant le copier-coller de texte et d'images : les séquences d'actions à effectuer par l'utilisateur sont les mêmes dans les deux cas, donc dans le réseau, les mêmes séquences de transitions seront activées pour copier-coller aussi bien du texte que des images. Cependant, si l'utilisateur pose la question "Comment copier-coller cet élément ?", l'aide devra être présentée différemment suivant que l'élément à manipuler est du texte ou une image. Ceci est rendu possible par le fait que les jetons circulant dans le réseau de Petri sont des références d'objets et que les annotations présentes dans les transitions (et exécutées lors du franchissement) peuvent mettre en œuvre des méthodes polymorphes associées à ces objets. La même action donne alors des résultats différents en fonction du type de l'objet auquel elle est appliquée. On aura

donc si besoin est, dans chaque classe d'objet un ensemble de méthodes dédiées à la génération de l'aide.

Le mécanisme que nous venons d'exposer est générique : on peut annoter le réseau avec des informations d'aide de n'importe quel type. Des messages texte, du son, de la parole pré-enregistrée ou générée à partir de texte ainsi que des séquences graphiques animées ou de la vidéo peuvent être associées à chacune des transitions pour servir d'informations d'aide élémentaires. On peut aussi associer plusieurs informations à chacune des transitions. Par exemple, une même transition peut être annotée par un message texte, par un message audio (son ou parole), et par une séquence vidéo illustrant la façon d'effectuer l'action élémentaire correspondant à la transition. En réponse à une demande de l'utilisateur, le système d'aide exploitera ces informations pour générer une séquence d'aide. On pourra ainsi par exemple présenter à l'utilisateur un message texte expliquant l'action à réaliser en même temps qu'une séquence vidéo montrant la façon de la réaliser. La présentation des informations associées à une transition est multimédia au sens de [Coutaz 93] : les différentes informations sont pré-enregistrées et sont rejouées en parallèle par le système d'aide lors du franchissement de la transition. Lors de la réponse à une question "Comment ?", le système d'aide détermine une suite de transitions à activer dans le réseau selon la technique présentée dans [Palanque 93]. Le système génère alors une séquence constituée des informations d'aide de chacune des transitions. La séquence d'aide est générée par le système de façon multimodale : en tenant compte du contexte, le système va déterminer les informations d'aide à utiliser et les assembler en une séquence. On a donc une combinaison multimodale d'informations élémentaires multimédia.

5. CONCLUSION

Dans cet article nous avons présenté comment la construction de l'aide contextuelle d'une application interactive peut être grandement simplifiée par l'exploitation de la spécification formelle de l'application au moyen du formalisme des ICO. Pour obtenir une gestion automatique de l'aide contextuelle par un moteur d'aide il suffit de rajouter des annotations dans le réseau de Petri décrivant le dialogue de l'application interactive. Suivant le type de ces annotations, nous avons montré que l'aide ainsi gérée par l'application peut être multimédia (dans le cas où les annotations exploitent plusieurs media différents) et même multimodale (dans le cas où la séquence des annotations est générée par le moteur d'aide).

Quand la même séquence de transition peut être activée par l'intermédiaire de différents interacteurs, on pourrait envisager d'intégrer au moteur d'aide un mécanisme intelligent permettant d'introduire une certaine adaptativité au niveau de l'aide contextuelle. Par exemple, pour copier, on peut choisir 'Copier' dans un menu ou bien taper l'équivalent-clavier. On pourrait alors envisager de s'appuyer sur un modèle d'utilisateur pour guider le moteur d'aide dans le choix de la séquence d'aide à présenter. De même, un modèle d'utilisateur peut être utile pour guider le choix des médias à utiliser lorsque chaque transition du réseau d'aide est annotée par des informations exprimées avec différents médias.

Nous travaillons actuellement à la construction d'un environnement de développement permetttant l'interprétation d'une application modélisée par ICO. Cet environnement intégre un éditeur graphique d'interface, un éditeur syntaxique permettant l'édition de réseau de Petri à objets, plusieurs modules d'analyse permettant de prouver certaines propriétés des modèles et un interprète d'ICO. Dans le cadre de l'aide contextuelle, nous prévoyons un interprète spécialisé pour la gestion des annotations de l'aide contextuelle.

7. RÉFÉRENCES

[Aaronson 87] A. Aaronson, J.M. Carroll. *The answer is in the question : A protocol study of intelligent help*. Behaviour and Information Technology n° 6, 1987. pp.

393-402.
[Apple 94] Système 7.5, version beta (logiciel), Apple Computer Inc., 1994.
[Bass 91] L. Bass, J. Coutaz. *Developing software for the user interface*. Addison-Wesley publishing 1991. ISBN 0-201-51046-4
[Carroll 88] J.M. Carroll, A. Aaronson. *Learning by doing with simulated help*. Communications of the ACM n°31, 1988. pp. 1054-1079.
[Coutaz 93] J. Coutaz, L. Nigay et D. Salber: *The MSM Framework: A Design Space for Multi-Sensori-Motor Systems*, EWHCI'93 Conference, Moscou, Russie, Août 1993.
[Gieskens 92] D.F. Gieskens, J.D. Foley. *Controlling user interface objects through pre- and postconditions*. Proceedings of the ACM CHI'92 conference, Monterey California 1992. pp. 189-194.
[Nigay 94] L. *Nigay: Conception et modélisation logicielle des systèmes interactifs : application aux interfaces multimodales*. Thèse de Doctorat de l'Université Joseph Fourier, Grenoble, 1994.
[Palanque 92] P. Palanque, *Modélisation par Objets Coopératifs Interactifs d'interfaces homme-machine dirigées par l'utilisateur*. Thèse de doctorat de l'Université Toulouse I (France), 1992.
[Palanque 93] P. Palanque, R. Bastide, L. Dourte. *Contextual help for free with formal dialogue design*. In HCI International 93 Orlando FL (USA) (8-13 August 1993).
[Peterson 81] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1981.
[Sukaviriya 92] P. Sukariviya, J. De Graaff. *Automatic generation of context-sensitive "Show & Tell" Help*. Technical Report GIT-GVU-92-18. Atlanta, Georgia. Graphics, Visualization and Usability center, Georgia Institute of Technology.