MODELES DE TACHE : ANALYSE COMPARATIVE, UTILITE ET LIMITATIONS

Sandrine BALBO, Joëlle COUTAZ

Laboratoire de Génie Informatique, LGI-IMAG
BP 53, 38041 Grenoble Cedex 9
tel. : (+33) 76.51.48.54
E-mail : Sandrine.Balbo@imag.fr, Joelle.Coutaz@imag.fr

Résumé
Cet article présente une revue simplifiée des modèles de tâche. L'analyse comprend deux axes de réflexion : la finalité d'un modèle de tâche et le formalisme support. Concernant la finalité, nous observons deux rôles complémentaires des modèles de tâche : 1) comme outil de conception intervenant dans le processus de développement d'un logiciel, 2) comme composant intégré au logiciel final en vue d'augmenter la robustesse de l'interaction. Concernant les formalismes, nous évoquons quelques critères d'évaluation parmi lesquels la rigueur sémantique, la description des relations temporelles et logiques entre les tâches, la richesse de description du comportement du système en sortie, et la lisibilité.

Mots clés : Modèles de tâche, Analyse de tâche, Interaction Homme-Machine, Formalismes, Spécification d'Interfaces Homme-Machine.

1. Introduction

La littérature fait état de recherches actives sur la modélisation de tâche. Il en résulte un espace de solutions riche qu'il convient de cerner si l'on veut délier le scepticisme d'une certaine communauté informatique. Cet article a pour objectif de clarifier la situation au moyen d'une analyse comparative entre des modèles de tâche représentatifs et notamment CLG [Moran 81], TAG [Payne 86], ETAG [Tauber 90], UAN [Hartson 92], TKS [Johnson 91], MAD [Pierret-Golbreich 89], GOMS [Card 83].

Pour cela, nous commencerons par quelques définitions introductives en relation avec le concept de tâche. Un modèle de tâche est un système de représentation de la notion de tâche. Comme tout système de représentation, le modèle adopté est fonction de sa finalité. L'utilité constitue notre premier élément de comparaison. Un système de représentation nécessite le support d'un formalisme d'expression. La nature du formalisme constitue notre second axe d'analyse. Ces deux dimensions, utilité et formalisme, font l'objet des paragraphes 3 et 4. Nous conclurons l'exposé par une synthèse et une courte revue des limites des modèles actuels. Cette dernière facette de notre analyse est susceptible d'alimenter un atelier aux journées IHM'93 qui pourrait se donner comme objectif la définition d'un agenda de recherche en modélisation de tâche.

2. Définitions

Une tâche peut se voir comme un objectif (ou état souhaité) associé à une ou plusieurs procédures (ou plans) dont l'exécution permet d'atteindre l'état visé [Normand 92a]. Une procédure recouvre un ensemble d'opérations organisées par des relations

temporelles et logiques. Une opération peut être décomposée à son tour en sous-tâches ou bien être considérée comme primitive.

Selon la perspective et le niveau d'affinement adoptés, une opération primitive dénote une action physique, une tâche élémentaire, une tâche simple, une tâche unitaire ou encore une tâche de base! Dans ETAG, une opération primitive est nommée "tâche de base" (basic task). Dans TAG, une "tâche simple" (simple task) représente la perspective cognitive : "any task that the user can routinely perform with no demand for a control structure, such as branching and iteration, that requires monitoring of plan progress" [Payne, 1986, pp. 121]. Dans ETIT [Moran 83], Moran fait la distinction entre "tâche interne" et "tâche externe", la première correspondant à la notion de tâche de base (perspective système), la seconde à celle de tâche simple (perspective psychologique).

Une action physique se définit le plus souvent comme une opération indivisible que l'utilisateur applique à un dispositif d'entrée physique et qui a pour effet immédiat de modifier l'état du dispositif. Par exemple, enfoncer et relâcher une touche du clavier sont deux actions distinctes. Par extension, une action physique émanant du système est une opération indivisible sur le changement d'état perceptible du système. Il est fréquent dans les modèles de tâche de décomposer une tâche de base en termes d'actions physiques. L'analyse comparative qui suit offre l'opportunité de faire référence à ces termes.

3. Modèle de tâche et utilité

Un modèle de tâche peut être utile comme outil de conception d'un système ou comme aide à la résolution de tâche. Dans le premier cas, le plus fréquent, le modèle a pour client le concepteur. Dans le second cas, il s'adresse à l'utilisateur final du système.

3.1. Modèle de tâche comme outil de conception

En génie logiciel, le processus de conception d'un système répond à des étapes bien identifiées. Parmi ces étapes, nous retenons celles où le modèle de tâche est pertinent : la définition des besoins et les spécifications externes. Nous observons que pour chacune de ces étapes correspondent des modèles de tâche privilégiés.

Modèle de tâche et définition des besoins. La définition des besoins, au sens du génie logiciel, identifie les services attendus du système. A cet effet, une méthode telle que Diane [Barthet 88] peut être adoptée. Cette méthode (et bien d'autres) incluent une analyse de tâche que concrétise un modèle de tâche exprimé dans un formalisme idoine.

Ce modèle s'appuie toujours sur une décomposition hiérarchique et conceptuelle des tâches. La hiérarchie traduit l'organisation structurelle de l'espace de travail et l'analyse conceptuelle décrit la nature des concepts manipulés dans le domaine. Le graphe orienté de tâches ainsi obtenu est le plus souvent décoré d'attributs traduisant des phénomènes importants observés sur le terrain comme l'ordre temporel, la fréquence d'utilisation ou la difficulté cognitive. Ces attributs pourront ensuite être considérés pour la définition des spécifications externes.

Le modèle de tâche conduit cependant à une double interprétation selon que l'on adopte le point de vue système ou le point de vue cognitif. Il peut représenter le fonctionnement effectif du système, c'est-à-dire l'ensemble des services que le système devra fournir ou bien il peut modéliser la connaissance que l'utilisateur devra posséder pour utiliser le système. Cette double interprétation vaut pour CLG. Pour ETAG au contraire, le modèle traduit uniquement le fonctionnement du système.

Pour Diane et dans une certaine mesure pour UAN, le modèle recouvre les tâches accomplies aussi bien par le système que par l'utilisateur. Dans CLG, le concepteur introduit des tâches "cognitives" telle la recherche d'information sur l'écran. De tels systèmes de représentation offrent une vue interactionnelle complète mettant en évidence les responsabilités et besoins des agents communicants.

L'intérêt des modèles de cette catégorie est double : une documentation formelle des besoins ouvre la voie à la génération automatique d'interface et à l'évaluation prédictive. Par exemple, le projet ADEPT [Johnson 93], qui s'appuie sur une

formalisation TKS de l'espace des tâches et sur un modèle de l'utilisateur, est capable de produire automatiquement une interface virtuelle projetable ensuite dans les termes d'une boîte à outils réelle telle que Motif. MAD vise un objectif similaire. Siroco n'inclut pas de modèle de tâche explicite mais suggère au concepteur de refléter sous forme d'espaces de travail la décomposition hiérarchique qui résulte de l'analyse de tâche [Normand 92b].

Quant à l'évaluation prédictive, GOMS et ses dérivés tels CCT [Kieras 85] et ETIT [Moran 83] offrent des métriques, certes réductrices mais utiles si l'on considère qu'elles permettent de détecter les erreurs de conception les plus grossières ou d'effectuer des analyses comparatives et donc de guider des choix de conception avant même la mise en œuvre, réputée coûteuse, du prototype.

Modèle de tâche et spécifications externes. Les spécifications externes d'un logiciel interactif définissent la vue que l'utilisateur aura du système final. Cette vue doit donner accès aux services définis dans l'analyse des besoins. Les formalismes orientés tâche-simple et action tels que TAG, ETAG et UAN sont particulièrement bien adaptés à cette représentation.

En dépit des limites que nous évoquerons plus loin, les formalismes orientés action ont une double utilité : comme outils de spécification et comme éléments d'analyse prédictive. En tant qu'outil de spécification, ils fournissent un complément précieux aux spécifications externes informelles. Cet argument tient pour UAN dont la motivation originelle est de servir d'outil de communication entre le concepteur-ergonome et le concepteur de logiciel. A titre prospectif, une telle description pourrait servir de donnée d'entrée à un outil de conception logicielle, qui comme PAC-Expert [Nigay 92], produirait les spécifications internes du logiciel répondant à la description comportementale externe du système.

L'analyse prédictive à partir de modèles de tâche simple concerne nécessairement l'épiderme de l'interface. P. Reisner avec son Action Grammar fut l'une des premières à évaluer l'utilisabilité et la facilité d'apprentissage de la surface de l'interface à partir de métriques sur la grammaire comme le nombre de règles [Reisner 81]. TAG répond à une motivation similaire. En particulier, Payne dans [Payne 86] démontre comment TAG peut être utilisé pour détecter la violation du principe de congruence.

Nous venons de considérer les modèles de tâche d'aide à la conception. Ceux-ci voient leur prolongement dans la conception logicielle mais ne sont pas conservés tels quels. Par exemple, la modélisation de tâche dirige la conception logicielle du contrôleur de dialogue, composant système dont le rôle est de structurer l'espace des fonctions de manière conforme au modèle de tâche. Mais, à l'inverse des modèles embarqués que nous abordons au paragraphe suivant, le modèle lui-même n'est pas transcrit explicitement au sein du logiciel.

3.2. Modèle de tâche comme composant du système final

Les modèles de tâche comme composants logiciels du système final participent, pour l'essentiel, à la robustesse de l'interaction. Nous avons retenu deux exemples illustratifs : le cas des interfaces langagières et l'aide à l'accomplissement de tâche.

Dans les interfaces langagières fondées sur la langue naturelle, le modèle de tâche sert de support à la définition sémantico-pragmatique, complément indispensable du lexique [Pierrel 87]. Cette représentation explicite permet de résoudre en partie le problème des expressions déictiques, des références anaphoriques et des ellipses. Les systèmes graphiques usuels qui canalisent le comportement de l'utilisateur en appliquant le principe des retours d'information pro-actifs [Sellen 92]), ne nécessitent pas de modèle de tâche pour gérer le comportement inattendu de l'utilisateur, ou pour combler les imperfections des systèmes de reconnaissance de "langage naturel". Pour les interfaces avancées et notamment les interfaces multimodales, un modèle de tâche explicite devient incontournable. Cubricon [Neal 88] et MMI2 [Wilson 91] sont deux exemples de tels systèmes incluant un modèle de tâche embarqué.

La seconde utilisation intéressante des modèles de tâches embarqués est l'aide dynamique à la résolution de tâche. Le principe est le suivant : le comportement de l'utilisateur est capté par le système, puis comparé au modèle de tâche prévu (c.-à-d. le modèle embarqué). En cas de détournement, le système en avertit l'utilisateur et peut proposer une approche (comme dans les didacticiels) ou suggérer de prendre en charge l'accomplissement de la tâche (on parle alors de migration dynamique de tâche) comme dans les systèmes de conduite de processus. Cette technique donne de bons résultats pour les domaines opératifs où la tâche est parfaitement structurée. OPAL est une bonne illustration de ce principe [Hoshstrasser 89]. Inversement, pour les tâches qui induisent un plan de résolution opportuniste, et notamment en conception, cette technique reste inopérante.

Le modèle de tâche embarqué voit un nouveau domaine d'application avec l'évaluation automatique des interfaces. C'est ce que nous tentons actuellement dans notre équipe [Balbo 92]. A l'inverse des modèles précédents qui s'appuient tous sur une décomposition hiérarchique, nous avons adopté une organisation plate : un graphe de tâches de base qui représente les cheminements possibles dans l'espace des commandes du système. Cette technique nous a permis de détecter automatiquement des difficultés d'utilisation et notamment des changements de direction reflétant des culs-de-sac "cognitifs", une surcharge de l'écran, etc.

4. Modèle de tâche et formalismes

Tout formalisme prend part à l'utilisabilité et à la portée du modèle dont il est le support. Concernant les formalismes pour les modèles de tâche, nous avons identifié quatre axes d'analyse : éléments directeurs, rigueur sémantique, puissance d'expression, et lisibilité.

4.1. Eléments directeurs

Le formalisme utilisé pour l'expression d'un modèle induit une approche dans l'analyse du problème. Par exemple, dans TKS, MAD, UAN, GOMS et CLG, l'objet central de la description est la tâche : le formalisme support suppose une analyse fonctionnelle du problème. Dans d'autres, comme Siroco et ETAG, les objets directeurs sont les concepts du domaine : le formalisme support induit alors une approche par objets sur lesquels sont appliquées des opérations perçues comme des tâches.

4.2. Rigueur sémantique

La rigueur sémantique est une propriété appréciée : absence d'ambiguïtés, propriétés démontrables et développement d'outils automatiques. Les formalismes de modélisation de tâche souffrent généralement d'un manque de rigueur. CLG et GOMS sont des exemples notoires. Siroco et TKS, qui conduisent à la génération automatique d'interfaces, offrent au contraire une sémantique rigoureuse mais leur puissance d'expression est-elle suffisante?

4.3. Puissance d'expression

La puissance d'expression est une seconde propriété intéressante puisqu'elle détermine la portée descriptive du modèle. Pour les formalismes de modèles de tâche, nous relevons les niveaux de description, l'expression des relations temporelles, la description des relations entre les expressions d'entrée formulées par l'utilisateur et les expressions de sortie émanant du système.

Niveaux de description. Selon les formalismes, les niveaux de description ont un usage différent. Dans CLG, les niveaux tâche, sémantique, syntaxique, et interaction servent de pont entre les domaines de conception. En particulier, le niveau tâche représente l'utilité du système dans le domaine considéré tandis que le niveau sémantique enrichit le modèle du niveau tâche avec des concepts dirigés par le système. Le niveau tâche définit les besoins et le niveau sémantique sert de relais vers la spécification logicielle détaillée. Dans

GOMS, un niveau correspond davantage à un niveau d'abstraction : les opérations d'un niveau donné présentent la même granularité c'est-à-dire une durée d'exécution de même ordre de grandeur. Dans UAN ou MAD, le raffinement décrit essentiellement les plans d'actions possibles.

Relations temporelles. Les formalismes qui expriment la décomposition hiérarchique de tâches incluent tous la possibilité d'exprimer le séquencement (tâche A puis tâche B), l'alternative (tâche A ou bien tâche B), la composition (tâche A et tâche B dans un ordre quelconque), l'itération, ou encore l'exécution concurrente de plusieurs tâches (tâche A en même temps que tâche B). Toutefois, contrairement à UAN, la plupart des formalismes ne distinguent pas l'exécution véritablement parallèle de l'exécution entrelacée. Ils ignorent le concept d'interruptibilité ou l'évoquent de manière implicite. De même, ils ne permettent pas d'exprimer des délais de garde entre l'exécution de plusieurs tâches.

Relations entre les entrées et les sorties. La plupart des formalismes de modèle de tâche mettent l'accent sur la spécification des entrées, c'est-à-dire sur la description du comportement que l'utilisateur doit observer pour se servir du système avec succès. Si la finalité du modèle est de prédire l'utilisabilité du système, c'est ignorer le rôle des retours d'information. Et pourtant il a été démontré que les "skilled users become good at getting the right information from the display." [Howes 91, pp. 113]. A ce titre TAG a été étendu par D-TAG (Display-TAG) pour tenir compte des informations visuelles [Howes 90]. La grammaire multipartie de Shneiderman est un autre exemple [Shneiderman 82]. Les formalismes qui considèrent explicitement la description des sorties, tels CLG et UAN, manquent cependant de rigueur ou bien sont réducteurs. Par exemple, l'opération UAN, "display[Object]" ne fournit aucune information sur le rendu visuel de cet objet (texture, couleur, structure, etc.).

4.4. Lisibilité

La lisibilité d'un formalisme vient souvent en contradiction avec la rigueur et le pouvoir d'expression. Le succès grandissant de UAN ne tient pas seulement à son pouvoir d'expression mais pour une large part, semble-t-il, au caractère lisible de sa notation.

5. Synthèse

En synthèse, cet article propose une revue simplifiée des modèles de tâche. Notre analyse comprend deux axes de réflexion : la finalité d'un modèle de tâche et le formalisme support.

Concernant la finalité, nous observons deux rôles complémentaires de la modélisation de tâche : 1) outil de conception intervenant dans le processus de développement d'un logiciel : spécifications des besoins et spécifications externes ; 2) composant intégré au logiciel final en vue d'augmenter la robustesse de l'interaction.

Concernant les formalismes, nous avons évoqué quatre critères pertinents : l'analyse induite (fonctionnelle ou par objets), la rigueur sémantique (avec l'ouverture possible vers la réalisation d'outils automatiques), la description des relations temporelles (et notamment la distinction entre exécution entrelacée, parallélisme vrai et interruptibilité entre tâches), la richesse souhaitable de la description du comportement du système en sortie, et la lisibilité. Avec l'expression du parallélisme et la description des expressions de sortie, nous avons esquissé un aspect seulement de la couverture des formalismes. Une étude intéressante consisterait à étudier comment étendre les formalismes actuels au cas des interfaces multimodales.

Cet article pose les bases d'une classification des modèles de tâche et exprime un ensemble de propriétés souhaitables dans ce domaine. Il ne prétend pas être exhaustif. Un atelier sur ce thème lors du colloque IHM'93 pourrait l'enrichir de manière utile.

6. Références bibliographiques

[Balbo 92]
S. Balbo, J. Coutaz, “Un pas vers l’évaluation automatique des interfaces homme-machine”, ERGO-IA’92, Biarritz, 1992, pp. 367-382

[Barthet 88]
M.-F. Barthet, “Logiciels interactifs et ergonomie, modèles et méthodes de conception”, Dunod Informatique, 1988

[Card 83]
S.K. Card, T.P. Moran & A. Newell, “The psychology of Human Computer Interaction”, Lawrence Erlbaum Associates, 1983

[Hartson 92]
H.R. Hartson & P.D. Gray, "Temporal aspects of tasks in the User Action Notation", Human-computer interaction, Vol. 7, pp. 1-45, 1992

[Hoshstrasser 89]
B. H. Hoshstrasser, N. D. Geddes, “OPAL, Operator Intent Inferencing for Intelligent Operator Support Systems”, Technical Report, Search Technology, Inc., 4725 Peachtree Corners Circle, Norcross, GA 30092

[Howes 90]
A. Howes & S.J. Payne, “Display based competence: Towards user models for menu driven interfaces”, International Journal of Man-Machine Studies, 33 (6), 1990, pp. 637-655

[Howes 91]
A. Howes & R.M. Young, “Predicting the learnability of task-Action Mappings”, CHI’91 proceedings, 1991, pp. 113-118

[Johnson 91]
H. Johnson & P. Johnson, “Task Knowledge Structures : Psychological basis and integration into system design”, Acta Psychologica, pp. 3-26, 1991

[Johnson 93]
P. Johnson, S. Wilson, P. Markopoulos & J. Pycock, “ADEPT-Advanced Design Environment for Prototyping with Task Models”, Interchi’93 proceedings, pp. 56, 1993

[Kieras 85]
D. Kieras & P.G. Polson, “An Approach to the Formal Analysis of User Complexity”, International Journal of Man Machine Studies, 22, 1985, pp. 365-394

[Moran 81]
T.P. Moran, "The command Language Grammar: a representation for the user interface of interactive computer systems", in Internal Journal of Man-Machine studies, no 15, pp. 3-50, 1981

[Moran 83]
T.P. Moran, "Getting into a System: External-Internal Task Mapping Analysis", in CHI’83 proceedings, 1983, pp. 45-49

[Neal 88]
J. Neal, C. Thielman, K. Bettinger, J. Byoun, "Multi-modal References in Human-Computer Dialogue", Proceedings of AAAI-88, 1988, pp. 819-823

[Nigay 92]
L. Nigay, J. Coutaz, “PAC-Expert: Towards an automatic generation of Dialogue Controllers”, Rapport de recherche, Laboratoire de Génie Informatique, IMAG RT 83, mai, 1992.

[Normand 92a]
V. Normand, “Task Modelling in HCI: Purposes and Means, State of the Art and Research Issues”, Rapport de Recherche, Thomson-CSF, Division SDC, 7 rue des Mathurins, BP 10, 92223 Bagneux Cedex

[Normand 92b]
V. Normand, “Le modèle Siroco : de la spécification conceptuelle des interfaces utilisateur à leur réalisation”, thèse de l’Université Joseph Fourier, Grenoble, 1992

[Payne 86]
S.J. Payne & T.R.G. Green, “Task-Action Grammars: a model of the mental representation of task languages”, in Human-Computer Interaction, Vol. 2, No. 2, Lawrence Erlbaum Associates Publishers, 1986

[Pierrel 87]
J-M. Pierrel, “Dialogue Oral Homme-Machine”, Hermes, 1987

[Pierret-Golbreich 89]
C. Pierret-Goldreich, I. Delouis & D.L. Scapin, "Un Outil d'Acquisition et de Représentation des Taches Orienté-Objet", rapport de recherche INRIA no 1063, Programme 8 : Communication Homme-Machine, août 1989

[Reisner 81]
P. Reisner, “Formal Grammar and Design of an Interactive System”, IEEE Transaction on Software Engineering, SE-5, 1981, pp. 229-240

[Shneiderman 82]
B. Shneiderman, “Multi-party grammars and related features for designing interactive systems”, IEEE Transactions on Systems, Man, and Cybernetics, 12, 1982, pp. 148-154

[Sellen 92]
A.J. Sellen, G.P. Kurtenbach, W.A.S. Buxton, “The Prevention of Mode Errors through Sensory Feedback”, Human Computer Interaction, Vol. 7, Number 2, 1992, pp. 141-164

[Tauber 90]
M.J. Tauber, “ETAG: Extended Task-Action Grammar - A language for the description of the user’s task language”, in Proceedings of INTERACT’90, D. Diaper et al. (Editors), Elsevier Science Publishers, North-Holland, 1990

[Wilson 91]
M. Wilson, “The first MMI2 Demonstrator: A Multimodal Interface for Man Machine Interaction with Knowledge Based System”, Deliverable D7, ESPRIT project 2474 MMI2, Tech. report Rutherford Appleton Laboratory, Chilton Didcot Oxon OX11 0QX, RAL-91-093, 1991