Neuvièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'97), Futuroscope, Cépaduès Éditions, p. 23-29.

Utilisation des spécifications formelles dans le processus de conception des Interfaces Homme-Machine

Philippe BRUN

LRI - Bâtiment 490
Université Paris-Sud
91405 Orsay cedex, France
Philippe.Brun@lri.fr

Francis JAMBON

CLIPS - IMAG
B.P. 53
38041 Grenoble cedex 9, France
Francis.Jambon@imag.fr

RESUME

Il n'existe pas encore à l'heure actuelle de méthodes formelles ou de notations satisfaisantes aussi bien pour les psycho-ergonomes que pour les concepteurs de systèmes interactifs. Les premiers, chargés d'effectuer l'analyse de tâche des futurs systèmes, trouvent les méthodes formelles peu adaptées à la prise en compte des facteurs humains et difficiles à maîtriser. Les seconds reprochent aux notations utilisées par les psycho-ergonomes de ne pas avoir de sémantique formelle et donc d'être limitées à la seule analyse de tâche. En effet, elles n'ont pas la possibilité d'utiliser la puissance d'une sémantique formelle pour effectuer de la vérification, de la génération de code, ou de jeux de tests. Dans cet article, nous essayons de définir une démarche de conception collaborative entre psycho-ergonomes et concepteurs d'interfaces homme-machine. Nous utilisons pour cela la notation de la méthode MAD et le formalisme de spécification des systèmes interactifs XTL qui est une logique temporelle. Nous montrons comment il est possible de concilier les deux approches et tirer partie de la puissance de l'un et de l'autre, tout en permettant une communication plus aisée entre les différents intervenants.

MOTS CLÉS : Formalismes de description de tâches, cycle de développement, vérification de conformité, XTL, MAD.

INTRODUCTION

La prise en compte systématique de l'analyse de tâche dans le processus de conception des interfaces homme-machine, même si elle est jugée indispensable [5], est dans la pratique industrielle peu utilisée. L'une des raisons communément invoquées pour expliquer cette lacune est la difficulté d'appliquer les besoins exprimés par l'analyse de tâche à la construction du système. Palanque et Bastide parlent d'un gouffre entre la vision "anthropocentrique" (orientée expression des besoins) et la vision "ordinocentrique" (orientée sur la conception) des spécifications issues de l'analyse de tâche [12].

En outre, Palanque et Bastide proposent dans le même article un cycle de développement itératif et incrémental visant à la mise en adéquation d'un modèle de tâche et d'un modèle du système tout deux exprimés dans le formalisme des réseaux de Petri orienté objet [11]. Notre démarche vise à un but similaire, c'est-à-dire à rendre explicite l'utilisation de l'analyse de tâche dans le processus de conception d'une interface homme-machine. Cependant, nous nous différencions de l'approche de Palanque et Bastide dans la mesure où nous nous plaçons plus en amont du cycle de conception d'une interface homme-machine, au niveau de la modélisation des tâches et jusqu'à celle de l'interface homme-machine.

De notre point de vue, le processus de conception d'une interface homme-machine peut être divisé en trois grandes étapes : l'analyse de tâche, la description de l'interface homme-machine, et les spécifications logicielles (Cf. figure 1). A chacune de ces étapes intervient un spécialiste du domaine utilisant ses propres notations et/ou formalismes. La disparité des notations et formalismes utilisés entre spécialistes de la psycho-ergonomie et de l'ingénierie des IHM rend difficile la communication des résultats entre ces deux domaines. Communication pourtant vitale car le processus de conception requiert en pratique de nombreux retours en arrière, suite à la découverte de fautes de conception ou de contraintes techniques.

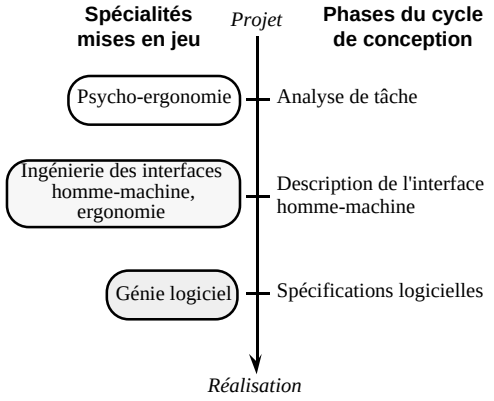

Figure 1 : Spécialités mises en jeu vis-à-vis des phases du cycle de conception d'une IHM.

Dans cet article, nous nous intéressons à la communication du savoir entre l'analyse de tâche et la description de l'IHM par le biais des spécifications formelles. En effet, l'usage de formalismes dans le processus de conception des IHM est l'un des moyens de rendre celui-ci plus efficace. Il est par ailleurs étudié par la communauté dans le cadre du groupe de travail GP3-

FLASHI du GDR-PRC "Communication Homme-Machine" auquel appartiennent les auteurs [14]. Notre propos peut être vu comme complémentaire aux approches Diane+/Merise [18], Muse/JSD* [8], ou plus récemment MAD*/ISS [4] au sens où nous mettons l'accent sur le point de contact entre le modèle de tâche et les spécifications de l'interface utilisateur. Il se situe par contre en amont d'une démarche comme celle suivie par Mastermind [17].

Notre propos vise à montrer que l'analyse de tâche recouvre des caractéristiques très différentes mais complémentaires entre le point de vue de la psycho-ergonomie et celui de l'ingénierie des IHM. Nous parlerons respectivement de vision "anthropocentrique" et "interactocentrique" (orienté vers la description de l'interface homme-machine). Nous pensons que l'une des difficultés de communication n'est pas tant la différence des formalismes utilisés, mais le point de vue de chaque spécialité. En effet, parmi les nombreux formalismes destinés à la spécification des IHM [2], des traducteurs entre formalismes peuvent être définis avec plus ou moins de difficultés selon les cas [6, 13]. Par contre, le point de vue de celui qui utilise le formalisme est lui très différent.

Cet article a pour but de mettre en lumière cette différence par le truchement d'un cas d'étude : la conception de Post-it® Notes collaboratifs électroniques. Notre démarche s'adresse à la fois aux psycho-ergonomes et aux concepteurs d'interfaces, c'est pourquoi nous utilisons successivement deux notations, MAD et XTL, que nous décrivons brièvement ci-après, à l'issue de la présentation de notre étude de cas. Le but de cet article n'est pas de présenter les formalismes, mais de montrer comment ils peuvent soutenir le processus de conception des IHM. Nous présentons ensuite notre cycle de conception en mettant en lumière l'intérêt des deux point de vue évoqués : "anthropocentrique" et "interactocentrique". Nous montrons également comment nous avons pu vérifier la conformité entre ces deux spécifications.

ÉTUDE DE CAS : LES POST-IT® NOTES

Notre exemple de travail est un utilitaire informatique ressemblant aux Post-it® Notes de 3M™. Pour cet exemple, nous nous sommes inspirés de l'étude de cas commune aux groupes FLASHI, GT-M3, et GT-SCOOP du GDR-PRC "Communication Homme-Machine" [10]. Notre version des Post-it® cumule les difficultés de spécification au sens où l'utilisateur peut envoyer un Post-it® à un autre utilisateur distant. L'utilitaire que nous désirons réaliser peut être vu comme une unification des logiciels "Stikies" et "Boardcast" existant actuellement sur Macintosh™. Pour plus de simplicité, nous nommerons dans la suite de cet article ce Post-it® collaboratif "Posticol", fusion de "Post-it" et "collaboratif". Nous avons élaboré un scénario typique d'utilisation de nos Posticols :

Un utilisateur, un ingénieur, rédige un Posticol, le détache du bloc, et le met dans un coin de son écran pour se rappeler d'un rendez-vous avec son chef de projet. Un autre utilisateur, un chef de projet, rédige de la même manière un Posticol, mais lui l'envoie à tous les membres de son équipe pour annuler la réunion prévue le soir même. Il rédige ensuite un autre Posticol pour changer la réunion qu'il avait prévue avec un de ces ingénieurs. Celui-ci, notre premier utilisateur, voyant le Posticol de son chef de projet arriver, en prend connaissance, puis décide de détruire le Posticol qu'il a sur son écran car il n'a plus lieu d'être : la réunion a été reportée. Il modifie néanmoins le Posticol de son chef de projet pour se rappeler du plan qu'il souhaite suivre au cours de la réunion. Puis il place ce Posticol dans le coin de son écran, maintenant libre.

Ces Posticols sont destinés à soutenir la collaboration entre les membres d'une équipe ou d'un laboratoire. Ils viennent en complément du courrier électronique, et se différencient de celui-ci par leur instantanéité et leur diffusion à un domaine restreint du point de vue réseau. Nous nous sommes donnés comme objectif la réalisation de cette application. Celle-ci utilise le langage Java™ de manière à être multi-plateforme (Cf. figure 2). La réalisation est en cours et l'application sera disponible à l'adresse : <http://iihm.imag.fr/jambon/posticol/>.

Figure 2 : Exemple de la présentation possible d'un Posticol.

NOTATIONS UTILISEES : MAD ET XTL

La Méthode Analytique de Description (MAD)

MAD répond à une méthodologie rigoureuse fondée sur une approche psycho-ergonomique [15]. Comme le montre la figure 3, le formalisme s'appuie sur une décomposition hiérarchique des tâches reliées par des constructeurs : la séquence (SEQ) exprime l'enchaînement séquentiel de plusieurs tâches, l'alternative (ALT) traduit la possibilité de choix entre plusieurs tâches, le parallélisme (PAR) désigne l'exécution entrelacée de plusieurs tâches par un même agent, et la simultanéité (SIM) dénote l'exécution simultanée de plusieurs tâches par des agents distincts [16].

Dans MAD, la description d'une tâche comprend :

- Une identification de tâche (numéro, nom de la tâche).
- Des éléments structurants (but, état initial, préconditions, corps de la tâche, postconditions, état final).
- Des attributs : tâche facultative (FAC), boucle (@), tâche prioritaire (PRIOR) et/ou interruptible (INTER), un niveau de priorité.

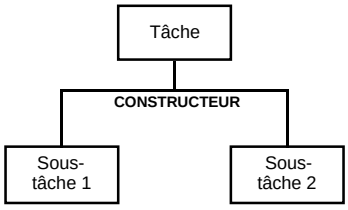

Figure 3 : Structure hiérarchique des tâches MAD

Le formalisme XTL (eXtended Temporal Logic)

XTL est un formalisme basé sur la logique temporelle et qui possède une sémantique précise. Par rapport aux logiques temporelles plus classiques [3, 7, 9] et aux formalismes dédiés à la spécification de systèmes interactifs, XTL a été construit pour prendre en compte des aspects peu abordés, et ce de manière formelle.

En particulier XTL est adapté à la spécification des aspects temporels du point de vue qualitatif et quantitatif ainsi qu'à la spécification de la concurrence et des singularités [6]. De plus, XTL possède une syntaxe claire et proche d'un langage de programmation ce qui rend les spécifications faciles à comprendre pour un informaticien. Par manque de place il ne nous est pas possible ici de décrire précisément la sémantique de XTL que l'on peut trouver dans [1].

En revanche, nous allons expliciter de manière semi-formelle chaque spécification. Le but ici étant de réfléchir sur les liens qui unissent modèles de tâches et modèles du système et non pas de présenter XTL exhaustivement.

ANALYSE DE TÂCHE / CYCLE DE VIE

Notre analyse de l'utilisation de nos Posticols, l'aspect anthropocentrique, se base sur une compilation des scénarii d'utilisation décrits au début de cet article. En cela nous avons suivis une démarche simplifié de celle proposée pour la méthode MAD par Sebillotte [15]. Le résultat de cette analyse a été décrit selon le formalisme MAD, associé à cette méthode. Nous avons ensuite spécifié ce cycle de vie en XTL de manière à pouvoir utiliser les capacités formelles, en termes de preuves, de cette logique temporelle.

Description selon MAD

Selon MAD, la description de l'utilisation d'un Posticol peut s'effectuer de la manière suivante :

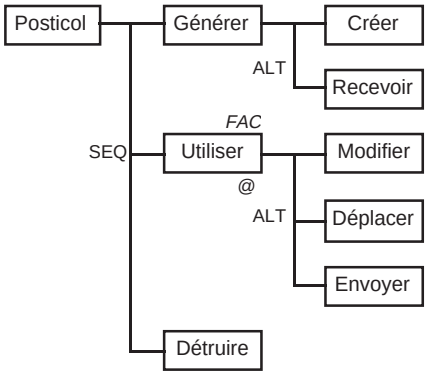

Figure 4 : Spécification de l'utilisation d'un Posticol selon MAD

Cette description de l'utilisation d'un Posticol ne peut pas être considérée comme une véritable analyse de tâche. En effet, ce sont plutôt les trajectoires d'interaction possibles lors de l'utilisation d'un Posticol qui sont décrites figure 4. Si l'on désire spécifier l'utilisation de plusieurs Posticols, il faut recourir à un artifice de spécification, c'est-à-dire spécifier que plusieurs Posticols peuvent être utilisés simultanément par l'utilisateur. Nous sommes ici obligés de spécifier la simultanéité (SIM) de l'utilisation de plusieurs Posticol et non l'entrelacement (PAR) car un Posticol peut arriver pendant que l'utilisateur interagit avec un autre Posticol (Cf. figure 5).

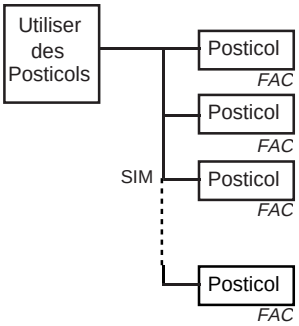

Figure 5 : Spécification de l'utilisation de plusieurs Posticols selon MAD

Ces descriptions ne permettent pas de concevoir directement l'interface homme-machine de notre application Posticol. Nous allons voir que l'utilisation de la logique temporelle XTL peut nous y aider.

Spécification équivalente en XTL

En XTL le cycle de vie d'un Posticol se spécifie de la façon suivante :

```
① module Posticol
  while (true)
  [
        u_générer;
        {u_utiliser};
        u_détruire;
  ]

module u_générer
   u_créer OR recevoir;

module u_utiliser
  while (true)
  [
        u_modifier OR
        u_déplacer OR
        u_envoyer
  ]
```

En XTL ";" a pour sémantique "AND Next" c'est à dire "Et dans le prochain état du système". Ce n'est donc pas un simple séparateur. De même, "{p}" où p est une formule valide a pour sémantique "empty OR p" avec empty désignant l'intervalle de temps vide. En d'autres termes "{p}" signifie "p vraie 0 ou 1 fois". Nous avons en outre préfixé par "u_" ce qui concerne l'utilisateur, le reste concerne le système. Dans notre cycle de vie on crée ou on reçoit un Posticol, on l'utilise éventuellement puis on le détruit. C'est donc un comportement linéaire.

Intégration du cycle de vie d'un Posticol dans le système de gestion de plusieurs Posticols

On veut maintenant, dans notre futur système interactif, pouvoir gérer à l'aide de l'interface plusieurs Posticols en même temps ce qui implique un comportement non linéaire qui n'est pas pris en compte par le module Posticol. Pour cela une modification non triviale de la spécification est nécessaire. En effet, on ne peut pas simplement modifier la spécification ① comme on pourrait en avoir envie en écrivant :

```
module Posticols
   Posticol^n
```

Ce qui signifierait : Posticol; Posticol; Posticol; ... n fois. C'est à dire toujours un déroulement linéaire. Il faut donc modifier notre spécification et mettre en parallèle les actions du module Posticol.

DESCRIPTION DE L'INTERFACE

La spécification du cycle de vie d'un Posticol étant terminée, nous devons résoudre le problème de l'utilisation simultanée de plusieurs Posticols. Nous utilisons pour cela les possibilités de preuves que permet XTL.

Interface d'utilisation des Posticols

XTL nous permet de manipuler des opérateurs de la concurrence avec au choix une sémantique du parallélisme ou de l'entrelacement. Dans notre cas on considère que l'on peut recevoir un Posticol pendant que l'on fait autre chose. Nous choisissons donc dans un premier temps d'utiliser le parallélisme non synchrone.

```
② module Posticol
  while (true)
  [
          ( {u_créer};
            {u_utiliser};
            {u_détruire}; )
          |||
          {recevoir};
  ]

module u_utiliser
  while (true)
  [
        u_modifier OR
        u_déplacer OR
        u_envoyer
  ]
```

Dans le premier cas ① seule l'action u_utiliser était facultative, dans le second ② les trois actions u_créer, u_utiliser, u_détruire deviennent facultatives (elles peuvent être exécutées ou non à chaque tour de boucle) et en parallèle (opérateur ||| du parallélisme non synchrone) avec l'action de 'recevoir' un Posticol extraite du module u_générer. Nous allons maintenant vérifier que notre spécification est bien conforme à ce que l'on souhaitait.

Démonstration par l'absurde (p et t sont en fait des variables implicites):

Théorème :

```
Quelque soit t1<t3, t2<t3 et P Posticol,

 u_utiliser(p,t3) => u_créer(p,t1)
                     OR
                     recevoir(p,t2)
```

En d'autres termes le fait d'utiliser le Posticol p au temps t3 implique t-il que celui-ci a déjà été créé ou reçu ?

Preuve :

Le module Posticol implique l'assertion vraie au moins une fois :

```
 ( {u_créer(p,t1)};
   {u_utiliser(p,t3};
   {u_détruire(p,t4)}; )
 |||
 {recevoir(p,t2)};
```

On déduit cette assertion en l'extrayant de la boucle while qui nous indique une répétition à l'infinie des formules temporelles qu'elle contient. Notons que p et t sont en fait des variables implicites : on manipule des

Posticols p de différentes façons à des moments différents t.

D'après la sémantique présentée au ① cette assertion est équivalente à :

```
( (empty OR u_créer(p,t1))
  AND Next
  ((empty OR u_utiliser(p,t3))
  AND Next
  (empty OR u_détruire(p,t4))) )
|||
(empty OR recevoir(p,t2))
```

en supposant (empty OR u_créer(p,t1)) ≡ empty, il existe un cas où l'assertion suivante et vraie :

```
( u_utiliser(p,t3)
  AND Next
  u_détruire(p,t4) )
|||
recevoir(p,t2)
```

Comme l'opérateur ||| ne permet pas de situer qualitativement dans le temps l'action recevoir, le théorème énoncé plus haut est faux : on a pas forcément t2<t3 c'est à dire que l'on ne peut pas situer dans le temps l'opération u_utiliser par rapport à l'opération "recevoir". En conséquence on peut utiliser et détruire un Posticol non encore crée et non encore reçu ❑

On doit donc avoir recours à des préconditions pour assurer que l'interface ne permettra pas à l'utilisateur d'enclencher une action détruire sans existence de Posticol. Ces préconditions sont explicitées dans la description MAD figure 6. Pour éviter ce problème dans la spécification XTL on doit modifier notre spécification ② qui devient :

```
module Posticol
   u_créer OR recevoir;
  while (true)
  [
     ( {u_créer};
       {u_utiliser};
       {u_détruire}; )
     |||
     {recevoir};
  ]
```

On peut encore raffiner cette spécification en objectant que l'action de recevoir est en fait une interruption suivi d'un parallélisme, recevoir passant en parallèle avec les trois autres actions. En XTL il suffit de changer l'opérateur de parallélisme ||| par l'opérateur <-|| (interruption puis parallélisme) pour prendre en compte cette singularité. La spécification devient alors :

```
③ module Posticol
  while (true)
  [
     (   {u_créer};
         {u_utiliser};
         {u_détruire};   )
       <-||
     {recevoir};
  ]
```

Description en MAD

A partir de la spécification de l'interface en XTL ③, nous pouvons obtenir une spécification équivalente en MAD destinée au psycho-ergonome (figure 6). Cette description selon le formalisme de MAD utilise les extensions associées aux interruptions proposées dans [6].

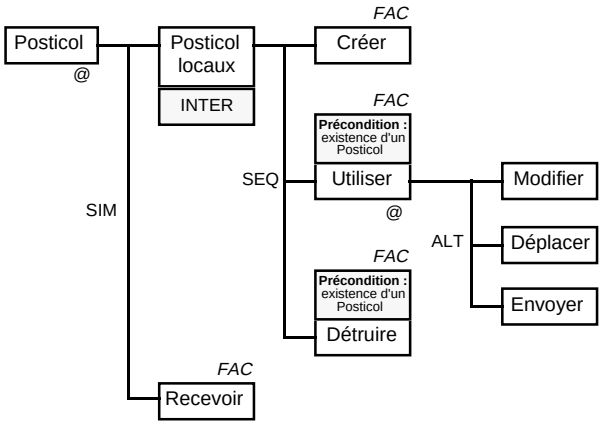

Figure 6 : Spécification de l'interface des Posticol selon MAD

Notons que cette spécification, représentant l'interface du système utilise des préconditions. En effet, celles-ci sont devenues obligatoires car le cycle de vie des Posticols n'est plus explicite. Celles-ci seront utiles au concepteur de l'IHM pour représenter l'activation des boutons ou items de menu de la future interface.

DISCUSSION

Cycle de conception

Notre approche met en lumière les étapes d'un cycle de conception centré sur la collaboration entre la psycho-ergonomie et l'ingénierie des IHM. Ce cycle de vie peut se résumer selon les étapes suivantes :

- Analyse de la tâche / du cycle de vie par un spécialiste de la psycho-ergonomie et aboutissant à une description MAD de celle-ci.
- Traduction en logique temporelle XTL de cette description.
- Modification de la spécification XTL de manière à respecter les contraintes de l'implémentation tout en vérifiant formellement la conformité des spécifications obtenues avec le travail des psycho-ergonomes.

- Traduction en MAD des spécifications obtenues de manière à pouvoir les communiquer aux psycho-ergonomes.
- Retour éventuel à la deuxième étape si les spécialistes de la psycho-ergonomie ont désiré modifier les spécifications.

Spécifications minimale et maximale

La première description de l'utilisation d'un Posticol représente le comportement typique d'un utilisateur face à l'application, nous l'avons nommé aspect "anthropocentrique". Cet aspect peut aussi être considéré comme la spécification "minimale" de l'application au sens où l'interface réalisée doit répondre à tous les besoins exprimés dans cette description.

Au contraire, la spécification en XTL finale obtenue après modification de la description MAD représente l'aspect "interactocentrique" de l'interface. Cet aspect peut être considéré comme l'interface "maximale" à réaliser au sens où de nombreuses fonctionnalités on pu être ajoutées par rapport à la description de départ. Cependant cette interface doit inclure l'interface minimale définie par le psycho-ergonome pour être conforme à l'analyse de la tâche.

A partir de ces notions d'interfaces minimale et maximale nous pouvons définir la notion de classe d'équivalence pour un ensemble de spécification d'interfaces. En effet, l'interface minimale peut être considérée comme l'élément générateur de plusieurs spécifications équivalentes d'interface se conformant à l'analyse de tâche. La tâche du concepteur d'IHM étant de choisir parmi ces possibilités quelle est celle la plus adéquate selon son expertise et les possibilités techniques dont il dispose.

Cette classe d'équivalence définit toutes les spécifications possibles issues d'une première description. Notre problème est la définition de la relation binaire sur l'ensemble de toutes les spécifications d'interfaces possibles correspondant à cette classe d'équivalence. Celle-ci doit être à la fois réflexive, symétrique et transitive pour correspondre à la définition mathématique d'une classe d'équivalence.

Vient ensuite le problème de la génération des possibilités. Il est par exemple possible, si l'utilisateur exécute deux tâches indépendantes en séquence, de générer des interfaces autorisant l'alternative (l'une puis l'autre dans n'importe quel ordre), l'entrelacement, ou bien encore le parallélisme. Puis vient la vérification que les qualités de l'interface sont réunies. La démonstration par l'absurde décrite au paragraphe précédent en est un exemple.

CONCLUSION

Nous avons montré dans cet article qu'il est possible d'aider la communication entre psycho-ergonomes et concepteurs d'IHM par le biais des méthodes formelles. Nous avons également montré l'importance des possibilités de preuves associées à la logique temporelle XTL. En outre, nous avons mis en lumière le fait que l'utilisation de notations et formalismes distincts entre les acteurs de la conceptions d'interfaces homme-machine n'est pas l'élément déterminant de leur manque de coopération.

Cependant, nous n'avons qu'ébauché la notion de classe d'équivalence associée aux spécifications d'interface. Nous sommes néanmoins certain que cette notion ouvre de bonnes perspectives de travail, du point de vue de la vérification de conformité, de la génération de toutes les interfaces possibles, et du filtrage automatique des possibilités sur critères ergonomiques.

REMERCIEMENTS

Les auteurs tiennent à remercier les membres du GDR-PRC "Communication Homme-Machine", et en particulier les membres du groupe FLASHI.

BIBLIOGRAPHIE

1. Brun, P. XTL: a temporal logic for the formal development of interactive systems. Paterno, F. et Palanque, P. (Ed.). In *Formal Methods for Human-Computer Interaction*, Springer-Verlag, 1997.

2. Brun, P. et Beaudouin-Lafon, M. A taxonomy and evaluation of formalisms for the specification of interactive systems. In *Proceedings of 10th annual conference of the British Human-Computer Interaction Group (HCI'95)* (August-September, University of Huddersfeild, UK), 1995.

3. Emerson, E.A. et Srinivasan, J. Branching Time Temporal Logic. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Springer-Verlag, 1989.

4. Gamboa Rodrígez, F. et Scapin, D.L. Editing MAD* task description for specifying user interfaces, at both semantic and presentation levels. In *Proceedings of Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'97)* (June 4-6, Granada, Spain), Berlin, Germany : Springer-Verlag, 1997.

5. Hix, D. et Hartson, H.R. *Developping user interfaces: Ensuring usability through product & process*. John Wiley & Sons, inc., Newyork, USA, 1993.

6. Jambon, F. *Erreurs et interruptions du point de vue de l'ingénierie de l'interaction homme-machine*. Thèse en informatique : Université Joseph Fourier (Grenoble I), décembre 1996.

7. Lamport, L. The temporal logic of actions. *ACM transactions on programming languages and systems*. 3, 16 (May 1994), pp. 872-923.

8. Long, J. Integrating human factors with software engineering for human-computer interaction. In *Proceedings of Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine (IHM'95)* (11-13 octobre, Toulouse, France), Cépaduès-Éditions, 1995, pp. 1-19.

9. Manna, Z. et Pnueli, A. The anchored version of the temporal framework. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Springer-Verlag, 1989.

10. Nigay, L., Palanque, P., Salber, D., Riveill, M. et Vigouroux, N. *Étude de cas commune FLASHI, GT-M3, GT-SCOOP : un système de "Post-it Notes"*. GDR-PRC "Communication Homme-Machine", 19-20 septembre 1996, Document de travail URL : http://outlet.imag.fr/scoop/etudedecas.html.

11. Palanque, P. *Modélisation par objets coopératifs d'interfaces homme-machine dirigées par l'utilisateur*. Thèse en informatique : Université Paul Sabatier (Toulouse I), 1992.

12. Palanque, P. et Bastide, R. Un cycle de développement pour la spécification formelle de systèmes interactifs. In *Proceedings of Huitièmes journées sur l'ingéniérie de l'Interaction Homme-Machine (IHM'96)* (16-18 septembre, Grenoble, France), Cépaduès-Éditions, 1996, pp. 51-56.

13. Palanque, P., Bastide, R. et Senges, V. Task model - system model: towards an unifying formalism. In *Proceedings of HCI International conference* (9-14 July, Yokoohama, Japan), Elsevier, 1995, pp. 489-494.

14. Palanque, P. et Girard, P. *Groupe de travail GP3-FLASHI (Formalismes et Langages Appliqués aux Systèmes Hautement Interactifs)*. GDR-PRC Communication Homme-Machine, 1996, Rapport d'activité.

15. Sebillotte, S. *Méthodologie pratique d'analyse de la tâche en vue de l'extraction de caractéristiques pertinentes pour la conception d'interfaces*. INRIA Rocquencourt, mai 1994, Rapport Technique n°163.

16. Sebillotte, S., Alonso, B., Fallah, D., Hamouche, H. et Scapin, D.L. *Note de recherche concernant le formalisme MAD*. INRIA Rocquencourt, Novembre 1994, Document interne du groupe Psycho-Ergo.

17. Szekely, P., Sukaviriya, P., Castells, P., Muthukumarasamy, J. et Salcher, E. Declarative interface models for user interface construction tools: the Mastermind approach. In *Proceedings of Engineering for Human-Computer Interaction* Chapman & Hall, 1996.

18. Tarby, J.-C. *Gestion automatique du dialogue homme-machine à partir de spécifiactions conceptuelles*. Thèse en informatique : Université Paul Sabatier (Toulouse I), septembre 1993.