Interruptions et formalismes de scripts de tâches

Francis JAMBON
CLIPS-IMAG
BP 53, 38041 GRENOBLE CEDEX 9, France
Francis.Jambon@imag.fr

RÉSUMÉ :
La notion d'interruption ne figure pas toujours explicitement dans les formalismes de description de tâches. Notre objectif est de définir les composantes d'un formalisme capable d'exprimer lisiblement et de manière exhaustive les cas d'interruption de tâche pour les systèmes mono-utilisateur. Nous présentons un aperçu de l'état de l'art sur la modélisation du concept d'interruption au moyen de trois formalismes représentatifs : MAD, les Blocs de Connaissance et UAN qui recouvrent des approches et des capacités d'expression complémentaires. En réponse aux insuffisances de l'existant, nous proposons une définition structurée de la notion d'interruption et un cahier des charges pour un nouveau formalisme, ISAU, qui apporte quelques premiers éléments de réponse. Enfin, pour illustrer ce cahier des charges, nous proposons des relations temporelles permettant d'étendre le pouvoir d'expression de UAN.

MOTS CLÉS : Formalisation, interruption, système critique, UAN, ISAU.

INTRODUCTION

La modélisation de tâche fait depuis longtemps l'objet d'attentions particulières mais les notations servant de support à l'expression de l'analyse de tâche manquent le plus souvent de rigueur sémantique, ne sont pas toujours très lisibles, ou leur pouvoir d'expression ne couvre que partiellement les besoins [2]. En particulier, la notion d'interruption, qui joue un rôle central dans les erreurs humaines [12], ne figure pas toujours explicitement dans les formalismes de description de tâches. Or l'interruption, marquée par une suspension ou même une suppression de l'activité en cours, est une propriété prégnante de l'interaction non seulement dans le cas des systèmes critiques (postes de pilotage, salles de contrôle, etc.), mais également dans les activités quotidiennes comme l'usage du téléphone public, ou en bureautique. Notre objectif est de définir les composantes d'un formalisme capable d'exprimer lisiblement et de manière exhaustive les cas d'interruption de tâche pour les systèmes mono-utilisateur. Les interfaces destinées au travail coopératif posent des problèmes spécifiques de formalisation [8] que nous ne saurions traiter ici.

En réaction à une sous-estimation chronique de l'importance des interruptions (et de leurs conséquences sur la sécurité) et des erreurs [5] dans les formalismes actuels, nous donnons à l'interruption un statut de concept central. L'interruption ne doit pas être vue comme une extension, une greffe a posteriori, d'un langage existant. Elle doit être pensée dès le départ. Modéliser les interruptions, leur rendre la place qu'elles ont toujours eue dans les systèmes, doit permettre aux concepteurs de prendre en compte les problèmes qu'elles posent dès le début du cycle de conception d'une interface homme-machine, et d'y apporter les solutions adéquates. Une banale interruption dans la plupart des cas peut avoir à un moment critique de l'interaction des conséquences dramatiques : Que dirait un contrôleur aérien si, au moment où il doit résoudre un risque de collision entre deux avions, le système lui ouvre une fenêtre sur son écran de contrôle lui demandant de rajouter du papier dans l'imprimante (de strips) ? C'est pourtant ce qui se passe sur la plupart des Macintoshs !

Pour justifier notre point de vue, nous présentons un aperçu de l'état de l'art sur la modélisation de la notion d'interruption dans trois formalismes représentatifs : MAD [13], les Blocs de Connaissance [4], et UAN[1] [7] qui recouvrent des approches et des capacités d'expression complémentaires. Nous illustrons notre analyse comparative au moyen d'un exemple simple mais réel : Le publiphone, i.e. le téléphone public à carte de France Télécom. En réponse aux insuffisances de l'existant, nous proposons une définition structurée de la notion d'interruption ainsi qu'un cahier des charges pour un nouveau formalisme : ISAU (Interruptions Système et Actions Utilisateur). Enfin, pour illustrer ce cahier des charges, nous proposons des relations temporelles permettant d'étendre le pouvoir d'expression de UAN.

LE SCÉNARIO : LE PUBLIPHONE

Dans notre scénario du publiphone, un utilisateur désire téléphoner mais commet une erreur en composant le numéro de son correspondant. Puis, en cours de communication, sa télécarte arrive en fin de crédit. Plus précisément : Après avoir décroché le combiné, l'utilisateur enfiche sa télécarte puis compose le numéro de son correspondant. Au cours de cette tâche, il commet une erreur et s'en aperçoit. Il raccroche puis re-décroche le combiné et, sans erreur cette fois-ci, re-compose le numéro. Au cours de la communication, le publiphone signale à l'utilisateur que sa télécarte arrive en fin de crédit et qu'il doit en changer. L'utilisateur retire la télécarte vide, puis insère une nouvelle télécarte tout en continuant la conversation. A la fin de la communication il raccroche le combiné puis retire sa télécarte, ou bien effectue l'inverse.

[1] Nous avons utilisié UAN plutôt que sa version étendue XUAN [6] car celle-ci n'apportait pas d'éléments nouveaux pour notre problématique.

Septièmes Journées sur l'Ingénierie des Interfaces Homme-Machine (IHM'95), Toulouse, France, 11-13 Octobre 1995, p. 161-168

Ce scénario, bien que fondé sur l'expérience, limite volontairement les possibilités d'interaction avec un publiphone afin de centrer la formalisation de l'interface sur les pierres d'achoppement que sont les interruptions. Le scénario retenu couvre deux sources d'interruptions : Les interruptions provoquées par le système (la télécarte est vide) et celles décidées par l'utilisateur (qui s'aperçoit d'une erreur de saisie de numéro). Les cas où l'utilisateur interrompt le système et où le système s'interrompt lui-même peuvent être traités de manière symétrique. Néanmoins, l'origine de l'interruption est un élément distinctif qu'il convient expliciter pour une analyse comparative de formalismes.

Ayant présenté le scénario, nous allons successivement étudier les réponses de MAD, des Blocs de Connaissance et de UAN.

Méthode Analytique de Description (MAD)

MAD [13], tout comme DIANE [3], répond à une méthodologie rigoureuse fondée sur une approche psycho-ergonomique [15]. Comme le montre la figure 1, 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. Dans une note récente, et pour répondre à des problèmes d'interprétation et d'utilisation de MAD, les auteurs clarifient leur formalisme et introduisent la notion d'interruption [14].

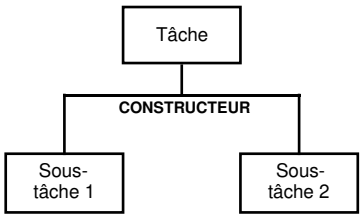

Figure 1 : Structure hiérarchique des tâches MAD

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.

La figure 2 montre notre modélisation du scénario au moyen de MAD. On constate d'emblée que la représentation graphique d'un scénario simple comme le publiphone nécessite une demi-page de description. Nous verrons que les autres formalismes utilisés dans les sections qui suivent sont plus compacts. Mais en corollaire MAD est lisible et une formalisation MAD ne demande que peu d'effort d'interprétation. Notre représentation appelle des remarques complémentaires en rapport avec la couverture d'expression ou la rigueur sémantique du formalisme.

Comme le montre la figure 2, une interruption se traduit directement au moyen du constructeur PAR : La tâche "Composer N°", décorée de l'attribut INTER, sera interrompue par la tâche "Reprise erreur N°" dès que la précondition "Erreur_N°" est satisfaite. En l'absence de l'attribut INTER, la tâche interrompante s'exécutera jusqu'au bout. Alors "Composer N°" peut reprendre et, d'après la spécification, la reprise se fera en début de tâche. Si cette expression traduit correctement le scénario, il convient néanmoins de s'interroger sur le cas où plusieurs tâches devaient interrompre simultanément "Composer N°" (leurs préconditions d'activation deviendraient Vrai en même temps). La notion de priorité répond au problème du choix d'exécution parmi les tâches activables mais conduit à des situations de famine de tâche (une tâche activable moins prioritaire perdrait en permanence son tour d'exécution). La sémantique du formalisme MAD n'offre pas de réponse sur ce point. De même, si plusieurs tâches activables sont de même priorité, laquelle est choisie?

La seconde interruption du scénario (télécarte en fin de crédit) n'est pas représentable correctement dans le formalisme MAD : L'opérateur PAR suppose une exécution entrelacée entre les tâches qu'il relie. Dans notre cas, il s'agit d'un parallélisme vrai : La conversation peut continuer pendant le changement de carte. Le constructeur SIM conviendrait à la situation mais suppose que les tâches soient exécutées par des agents distincts. MAD rejette explicitement le parallélisme vrai pour un agent donné. Or un utilisateur habile (au sens de Rasmussen [11]), et notamment l'habitué du publiphone, est capable d'exécuter plusieurs tâches simultanément, et notamment continuer la conversation pendant le changement de carte.

Une seconde limitation de la couverture d'expression de MAD est l'absence de contraintes temporelles. Dans l'exemple du scénario, les tâches séquentielles "Retirer carte" et "Insérer carte" qui composent la tâche "Changer carte vide" doivent être exécutées dans les limites d'un délai de quelques secondes avant coupure de la ligne. Dans le cas d'un publiphone, le non respect de cette contrainte par l'utilisateur ne devrait pas mettre en jeu la sécurité. Il n'en va pas de même dans les systèmes critiques où l'expression des contraintes temporelles doit être explicite.

MAD ne couvre pas la description des retours d'information en provenance du système. Cette propriété est cohérente avec les intentions initiales des auteurs qui cherchaient en MAD un outil de conception [15]. Il n'empêche que par affinements successifs, la conception doit se préoccuper du comportement perceptible du système, par exemple pour prévenir l'utilisateur du publiphone que sa carte est vide. Dans notre cas, le mode de notification des conditions anormales a un pouvoir déterminant sur le caractère disruptif du système.

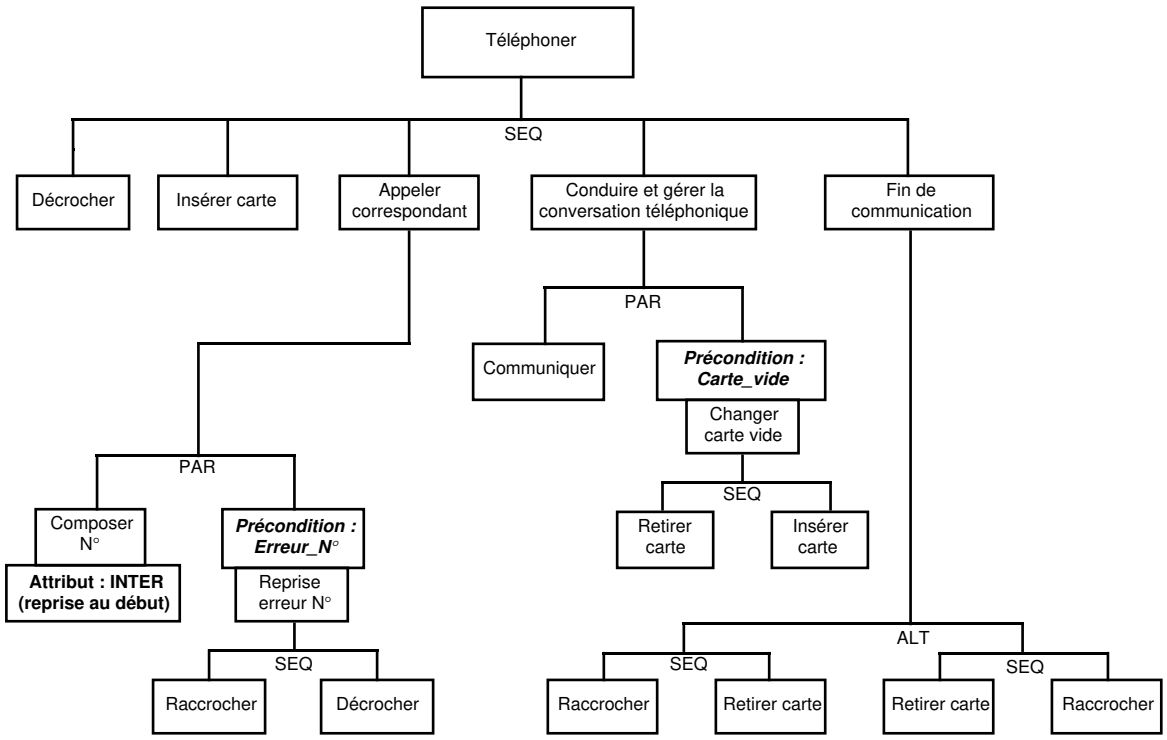

Figure 2 : Formalisation à l'aide de MAD

En résumé, dans sa forme graphique, MAD est peu compacte, mais lisible, et reconnaît le concept d'interruption. Nous relevons toutefois un vide sémantique concernant l'exécution des tâches d'interruption. Certes, l'ambiguïté vaut pour des cas extrêmes mais néanmoins possibles dans les systèmes critiques. MAD n'autorise pas l'expression du parallélisme vrai pour un agent donné ni ne permet l'expression des retours d'information du système.

Les Blocs de Connaissance (blocs)

Les Blocs de Connaissance sont issus de recherches en représentation des connaissances [4]. Ils ont été formalisés et appliqués à un système de télémanipulation spatiale [9], puis enrichis d'une capacité de parallélisme et utilisés pour la mise en œuvre d'un système d'aide à la conduite automobile [10].

Les blocs représentent la connaissance opérationnelle de l'utilisateur d'un système sous forme de tâches. Comme le montre la figure 3, un bloc comprend cinq parties : des préconditions, une procédure, un contexte ou conditions contextuelles, un but et des conditions anormales. Ces dernières rendent explicites les cas d'interruptions et d'erreurs. Les flèches renvoient ou proviennent d'autres blocs. Contrairement aux modèles de tâche usuels, la structure des tâches est un graphe orienté.

Le mécanisme d'exécution des blocs fait appel à un moteur d'inférence qui évalue les conditions des blocs "focalisés", puis si elles sont vérifiées, exécute les procédures de ces blocs. Une procédure contient des actions élémentaires ou d'autres blocs. Ces derniers sont focalisés lorsque la procédure du bloc englobant est exécutée. Les préconditions d'un bloc, nous l'avons vu, déterminent l'exécution de la procédure de ce bloc. Les conditions anormales et le but d'un bloc déterminent respectivement la focalisation des blocs référencés dans ces conditions et dans ce but.

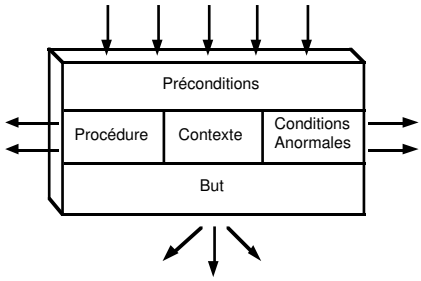

Figure 3 : Structure d'un Bloc de Connaissance.

Pour traiter l'exemple du scénario, nous nous sommes appuyés sur la dernière version des Blocs de Connaissance qui inclut l'expression du parallélisme [10]. Nous proposons deux formalisations du problème. Comme le montrent les figures 4 et 5, nous traitons de manière identique l'erreur de composition du numéro mais l'interruption système est envisagée selon deux points de vue.

Le formalisme des Blocs de Connaissance décrit de manière très efficace l'interruption due à l'erreur de saisie du numéro : cette erreur est une condition anormale et le formalisme offre un statut explicite aux situations anormales. La condition anormale "Erreur_N°" devenant

vraie, la procédure du bloc "Composer N°" est interrompue et le bloc "Raccrocher-Décrocher" est exécuté suivi d'un retour au début du bloc "Composer N°".

L'interruption système due à la carte vide est modélisée dans la figure 4 sous forme d'une condition anormale. En conséquence, comme dans MAD, nous perdons l'expression du parallélisme entre la tâche de communication et le changement de carte. La solution de la figure 5 répond à ce problème mais l'événement "Carte_vide", modélisé maintenant comme une précondition, perd son statut d'exception.

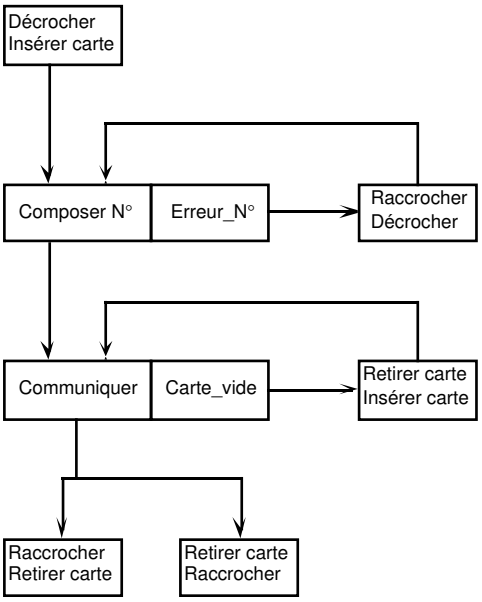

Figure 4 : Formalisation à l'aide des Blocs de Connaissance en utilisant uniquement les CA.

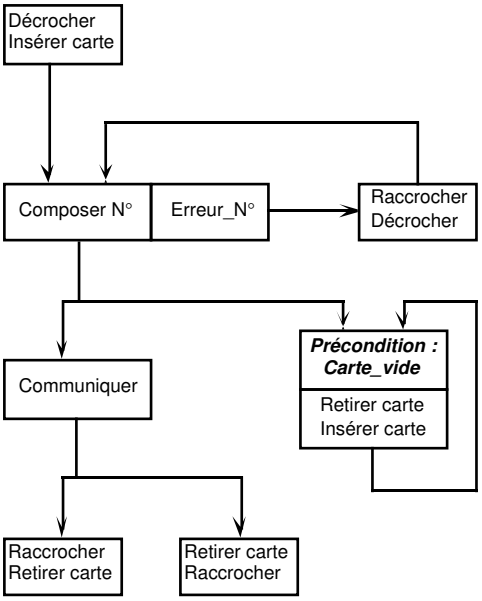

Figure 5 : Formalisation à l'aide des Blocs de connaissance en utilisant les CA et les préconditions.

Comme MAD, les Blocs de Connaissance ne permettent pas de décrire les retours d'information issus du système. On ignore donc comment l'utilisateur s'aperçoit que sa carte est vide et qu'il doit la changer. Si, contrairement à MAD, les blocs admettent le parallélisme entre plusieurs tâches exécutées par un même agent, ils ne couvrent pas l'expression des contraintes temporelles. Enfin, une interruption qui doit être traitée en parallèle avec la tâche en cours perd son statut de condition anormale. UAN répond partiellement à ces limitations.

User Action Notation (UAN)

A l'origine, UAN devait servir de support à l'expression des spécifications d'IHM de type manipulation directe. Le concepteur spécifie les actions de l'utilisateur sur les dispositifs physiques (d'où le nom "User Action Notation"), avec en regard les réactions du système et ses changements d'état interne pertinents. La Figure 6 illustre le format tabulaire d'une description de tâche élémentaire en UAN. Puis le langage s'est enrichi d'opérateurs de composition et de relations temporelles pour modéliser les tâches complexes qu'un utilisateur peut accomplir avec un système. Parmi ces opérateurs, nous relevons :

- Séquence (A B) signifie que les tâches A et B seront exécutées dans l'ordre A puis B.
- Choix (A|B) indique que l'utilisateur peut accomplir la tâche A ou la tâche B.
- Répétition (A*) dénote l'accomplissement multiple de la tâche A.
- Attente (t>x secondes ou t<y secondes) signifie que l'utilisateur doit attendre x secondes, ou au contraire ne doit pas dépasser y secondes avant d'exécuter la tâche suivante.
- Entrelacement bidirectionnel (A<->B) exprime que les tâches A et B s'effectuent durant le même intervalle de temps, mais une seule des tâches peut être active à la fois.
- Entrelacement monodirectionnel (A ->B) signifie que la tâche A peut interrompre la tâche B puis s'exécuter jusqu'à sa fin, et non l'inverse.
- Parallélisme (A ‖ B) dénote l'exécution simultanée des tâches A et B.
- Ordre indifférent (A&B) : l'utilisateur peut accomplir A puis B ou B puis A.

Tâche : ABCD

Action utilisateur	Retour d'information de l'interface	État interne
Action 1	Retour 1	État 1
Action 2	Retour 2	État 2
Action 3	Retour 3	État 3

Figure 6 : Structure d'une tâche UAN.

Dans notre exercice, nous n'avons pas toujours utilisé les trois colonnes du format UAN. Nous les avons explicitées seulement lorsqu'elles permettaient d'alimenter la discussion. Les quatre tableaux des figures 7 à 10 répondent à l'expression du scénario. Nous remarquons la

concision de la notation, à peine plus gourmande en espace que les Blocs de Connaissance, mais qui en contrepartie rend explicites les retours d'information du système.

Le tableau de la figure 7 montre la structure de la tâche de plus haut niveau : "Téléphoner". En séquence, l'utilisateur exécute les tâches "Décrocher" et "Insérer Carte" puis saisit le numéro du correspondant avant d'entrer en communication. Tout en maintenant la conversation avec son interlocuteur (opérateur ||), l'utilisateur peut changer de télécarte autant de fois qu'il le faut (symbole *). Puis il retire sa carte et raccroche le combiné dans l'ordre qui lui convient (opérateur &). L'expression du scénario est directe et simple mais le statut de condition anormale du changement de carte vide ou d'erreur de saisie reste implicite.

Tâche : Téléphoner
Décrocher
Insérer carte
Composer N°
(Communiquer || (Changer carte vide)*)
(Raccrocher & Retirer carte)

Figure 7 : Tâche UAN Téléphoner.

La figure 8 démontre l'inadéquation des opérateurs d'entrelacement (tel <->, cité plus haut) pour exprimer la terminaison brutale d'une tâche par une autre. Les opérateurs d'entrelacement de UAN expriment l'interruptibilité mais supposent une reprise de la tâche interrompue (sans pour autant préciser le lieu de reprise !). Dans notre cas, il s'agit de tuer la tâche de saisie afin de la re-exécuter en son début. Dans UAN, il n'existe pas d'opérateur "tuer tâche". Pour décrire l'erreur de saisie, nous avons dû modéliser la tâche au niveau des actions physiques et spécifier que la tâche de saisie ne se termine que si au moins un numéro à deux chiffres a été composé. Cette description est lourde et peu lisible et l'opérateur | ne rend pas compte du caractère disruptif de l'erreur de saisie.

Tâche : Composer N°
(Taper touche clavier | Reprise erreur N°)*
Taper touche clavier
Taper touche clavier

Figure 8 : Tâche UAN Composer numéro.

Les figures 9 et 10 démontrent les capacités de UAN à exprimer les contraintes temporelles d'exécution entre deux tâches séquentielles. Par exemple, dans la tâche de reprise sur erreur de numérotation, il convient pour l'utilisateur de raccrocher le combiné et de ne pas attendre plus de 2 secondes avant de le décrocher à nouveau. Le facteur temps, qui joue un rôle central dans les systèmes critiques, est donc explicité dans UAN. En revanche, UAN n'offre pas de matériau formel dédié à l'expression des anomalies.

Des trois formalismes étudiés, UAN possède le plus grand ensemble d'opérateurs de relations entre tâches. Il est le seul à considérer le facteur temps et à décrire explicitement les retours d'information. UAN, comme MAD, couvre le concept d'interruption sans toutefois, à la différence des Blocs des Connaissance, avoir la possibilité de lui donner le statut particulier de condition anormale. Sur ce point, UAN et MAD ne font pas exception. Comme pour la plupart des formalismes, l'expression des cas simples est directe mais la modélisation des erreurs est complexe. En conséquence, la description n'est lisible que par ceux qui l'ont produite. Ou alors, ces formalismes souffrent de l'absence d'un guide d'utilisation, ou encore d'une méthode comme celle associée à MAD.

Tâche : Reprise erreur N°	
Action Utilisateur	Retour d'information
Raccrocher	Affiche (DÉCROCHER)
(t < 2 secondes)	
Décrocher	

Figure 9 : Tâche UAN Reprise erreur N°.

Tâche : Changer carte vide	
Action Utilisateur	Retour d'information
	Affiche (RETIRER CARTE VIDE)
Retirer carte	Affiche (NOUVELLE CARTE)
(t < 10 secondes)	
Insérer carte	

Figure 10 : Tâche UAN Changer carte vide.

PROPOSITIONS

Ayant fait l'état des lieux, nous constatons qu'aucun des formalismes précédemment utilisés n'apporte de solution réellement satisfaisante car leur pouvoir d'expression est trop limité. Dans un premier temps, nous proposons une modélisation des interruptions afin de compléter les méthodologies d'analyse de tâche existantes, comme par exemple celle issue de MAD [15]. Puis, nous définissons le cahier des charges d'un formalisme, ISAU, capable d'intégrer la notion d'interruption de manière la plus exhaustive et la plus lisible possible. Enfin, pour illustrer ce cahier des charges, nous proposons des relations temporelles permettant d'étendre le pouvoir d'expression de UAN.

Modélisation de la notion d'interruption

Comme préalable à la définition de la notion d'interruption et de déroutement, nous devons définir la notion d'entités interactives ou plus simplement d'agents. Nous considérons un agent comme une entité pouvant interagir avec un autre agent à l'aide de canaux de communication. Un agent peut être un utilisateur ou bien un composant d'un système informatique. Il est implicitement défini qu'un agent est capable d'un minimum de traitement de l'information qu'il reçoit, et d'initiative d'action.

Du point de vue de l'architecture des processeurs « les interruptions sont définies comme les événements (exception faite des instructions de branchement) qui changent le déroulement normal de l'exécution d'un programme » [16]. Parmi ces interruptions les auteurs distinguent les interruptions internes provoquées lors de l'exécution d'une instruction par le processeur et les interruptions externes dues à l'environnement (entrées/sorties). En s'inspirant de ce point de vue système nous avons défini une interruption comme une discontinuation dans l'exécution d'une tâche par un agent provoquée par un autre agent ; et un déroutement comme une discontinuation dans l'exécution d'une tâche provoquée par l'agent lui-même. Après avoir décrit le modèle de la notion de déroutement, nous généraliserons celui-ci aux interruptions.

Un déroutement se décompose en trois phases séquentielles *(figure 11)* : le prologue, le corps, et enfin l'épilogue. La réunion de ces trois phases est appelée la tâche de traitement d'interruption. Le corps de l'interruption contient les tâches véritablement utiles de l'interruption, ce pour quoi elle a été déclenchée. Ces tâches peuvent être indépendantes du contexte, c'est à dire de la tâche interrompue. Au contraire, le prologue et l'épilogue sont des tâches articulatoires fortement dépendantes à la fois du contexte de l'interruption, mais également du corps de l'interruption lui-même.

Une interruption comprend une tâche de déclenchement de l'interruption *(figure 11)*, puis une tâche de traitement d'interruption proprement dit exécutées respectivement par l'agent interrompant et l'agent interrompu. la tâche d'interruption est similaire à celle du déroutement, quand à la tâche de déclenchement de l'interruption, on peut la décomposer en trois phases. La première phase est une prise de décision de déclencher l'interruption. La deuxième est la signalisation en direction de l'autre agent de l'arrivée d'une interruption, par l'intermédiaire d'un ou plusieurs canaux de communication. Cette signalisation peut avoir une durée significative qui peut être celle de la durée de la tâche d'interruption, et même au-delà. Enfin la tâche optionnelle de surveillance permet à l'agent interrompant de vérifier l'acquittement et éventuellement la bonne exécution de l'interruption. Cette tâche peut être effectuée en parallèle de la tâche de signalisation.

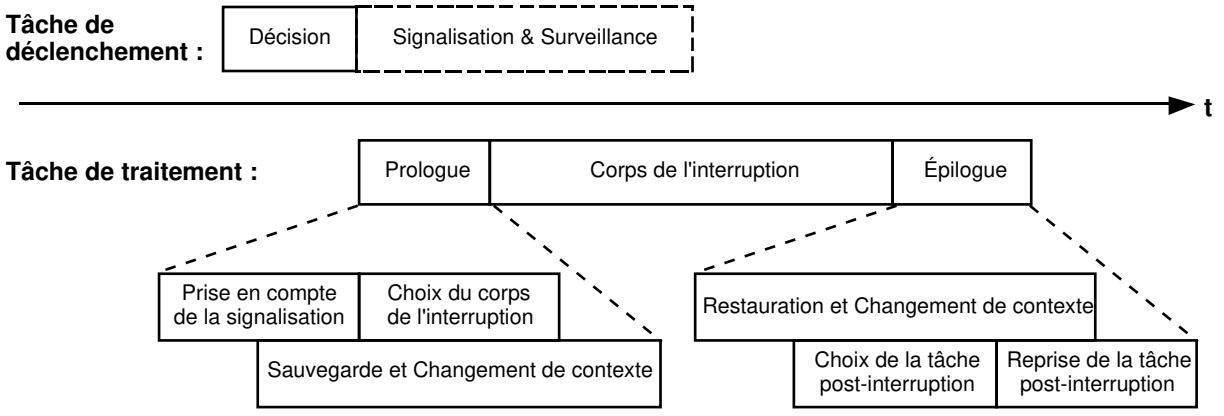

Figure 11 : Tâches de déclenchement et de traitement d'une interruption

Afin d'illustrer notre propos, considérons l'exemple d'un étudiant en train de fixer une affiche sur le mur de sa chambre alors que son téléphone sonne. Modélisons cet exemple avec nos deux agents : l'agent interrompu est l'étudiant, tandis que l'agent interrompant réunit le trio interlocuteur, téléphone, et affiche[2]. Selon notre modèle de déclenchement d'une interruption, la *décision* de déclencher la sonnerie du téléphone est prise par l'interlocuteur lorsqu'il décide de téléphoner, la *signalisation* est bien évidement la sonnerie du téléphone, et la *surveillance* est le fait qu'au bout d'un certain nombre de sonneries l'interlocuteur décidera qu'il n'y a personne à l'autre bout, et raccrochera. Le cas de l'agent interrompu est plus intéressant. Le *prologue* comprend la prise en compte de la sonnerie, suivie du choix effectué par l'agent pour savoir s'il va répondre ou non. Si l'étudiant décide de répondre, une tâche de sauvegarde de contexte, ici finir rapidement de fixer une punaise pour éviter que l'affiche ne se déchire, doit être exécutée. Enfin l'étudiant peut entreprendre de répondre au téléphone, ce qui constitue le *corps de l'interruption*. L'*épilogue* débute dès que l'étudiant a raccroché le combiné du téléphone. Il restaure alors partiellement son contexte (où en étais-je ?), puis choisit ou non de reprendre la tâche de fixer l'affiche. Si oui, il doit finir de restaurer le contexte en retirant la punaise mise rapidement lors du prologue. Puis

[2] Le choix de l'affectation des deux agents est réalisé selon les centres d'intérêt de la modélisation. Ici le centre d'intérêt est l'étudiant, le monde extérieur à cet agent est donc modélisé par l'autre agent.

il peut enfin finir de fixer son affiche au mur de sa chambre.

Ce petit exemple montre que le prologue et l'épilogue d'une tâche d'interruption peuvent se décomposer en sous-tâches aux rôles bien définis. Le prologue est initié par la prise en compte de la signalisation en provenance de l'agent interrompant. Débute aussitôt une tâche de sauvegarde et de changement de contexte qui permet de mettre en suspend la tâche interrompue sans conséquence grave, et de préparer la reprise. Cette tâche peut durer jusqu'à la fin du prologue. Juste après la prise en compte de la signalisation, l'agent interrompu doit choisir s'il doit exécuter une interruption, et si oui, quel corps d'interruption exécuter.

L'épilogue est construit de manière symétrique. Une restauration partielle du contexte de la tâche interrompue est nécessaire afin que l'agent interrompu puisse choisir quelle tâche effectuer après le corps de l'interruption. Il peut choisir de reprendre la tâche interrompue ou au contraire tout abandonner pour faire autre chose. Ensuite, les actions de sauvegarde du contexte ayant mis en suspend la tâche interrompue doivent être défaites, et enfin l'agent peut commencer la tâche post-interruption choisie. Cette tâche de reprise et le symétrique de la prise en compte de la signalisation, elle peut, de même, être constituée d'une signalisation en direction de l'autre agent. Cette tâche était absente de notre exemple précédent, dans le cas d'un système de multifenêtrage ce peut être le clic dans une nouvelle fenêtre.

Besoins en formalisation

La description de l'interface du publiphone à l'aide de trois formalismes aux caractéristiques complémentaires, puis la modélisation de la notion d'interruption permettent de définir une partie des besoins en pouvoir d'expression du formalisme ISAU. Ce formalisme, actuellement à l'état d'ébauche doit être considéré comme un canevas d'étude.

ISAU doit permettre de décrire exhaustivement la signalisation d'une interruption par l'intermédiaire du ou des canaux de communication disponibles entre les agents. En conséquence, ISAU, comme UAN, doit permettre d'exprimer les retours d'information en provenance de l'interface du système, et plus généralement toute sortie en direction de l'utilisateur. Nous ne proposerons pas de solutions à ce besoin, et nous contenterons pour l'instant des capacités de UAN.

ISAU doit permettre de décrire exhaustivement l'interruption d'une tâche par une autre par l'intermédiaire de relation temporelles, comme UAN, et/ou par l'intermédiaire d'attributs, comme MAD. Néanmoins cette description des interruptions ne doit pas occulter la description des tâches habituelles par un flot de notations parasites. Ce critère de lisibilité est très difficile à évaluer objectivement. Afin de se faire une idée globale de la lisibilité d'une description nous prendrons en compte la place occupée par cette description, et la facilité avec laquelle un utilisateur novice détecte la tâche de base de l'interaction.

Dans ISAU nous avons préféré privilégier les relations temporelles entre les tâches au détriment de l'usage des attributs. Ceci permet de supprimer l'ambiguïté présente dans MAD sur l'identité de la tâche interrompue, et d'améliorer la lisibilité. Une interruption d'une tâche peut revêtir quatre formes principales :

- Interruption sans reprise.
- Interruption avec entrelacement.
- Interruption avec parallélisme.
- Interruption avec reprise.

Les quatre cas généraux d'interruptions ne décrivent en fait que les cas simples. Le cas général est bien plus complexe à modéliser. Examinons l'exemple d'une personne qui reçoit un coup de téléphone d'une personne très bavarde alors quelle était en train de lire un catalogue. On peut imaginer que le prologue et l'épilogue de la tâche d'interruption "Coup de téléphone" suspendront la tâche "Lire catalogue", mais il y a fort à parier que pendant le corps de l'interruption, c'est à dire la tâche "Écouter personne", la lecture du catalogue reprendra en parallèle. On est donc obligé de représenter l'interruption avec parallélisme, puis de définir des attributs de priorité d'interruption pour décrire précisément le déroulement réel de l'interaction. La démarche consiste alors à décrire à l'aide de relations temporelles l'interaction la plus libre possible, et de la brider ensuite à l'aide des attributs de priorité entre les tâches. La modélisation des interruptions peut nous aider dans cette tâche.

L'interruption avec reprise peut encore être détaillée :

- Reprise au début de la tâche interrompue.
- Reprise à la fin de la tâche interrompue.
- Reprise au point d'interruption (en cours).
- Reprise à un point particulier à préciser.

Les formalismes MAD et UAN définissent déjà en partie ces relations temporelles nécessaires à la description des cas d'interruptions, respectivement sous la forme d'attributs, ou de relations temporelles.

Extension de UAN

Afin d'illustrer notre cahier des charges pour ISAU, nous proposons des relations temporelles permettant d'étendre le pouvoir d'expression de UAN, mais aussi de XUAN [6] qui est déjà une version étendue de UAN. À l'aide des relations "interruption avec reprise au début de la tâche interrompue" représentée par "**<-¡**" et "interruption avec parallélisme" représentée par "**<-||**" la formalisation du publiphone (figure 12) ne pose pas de problème particulier. Les tâches "Reprise erreur N°" et "Changer carte vide" sont identiques aux tâches UAN décrites précédemment (figures 9 et 10). Les cinq autres types d'interruptions recensés précédemment peuvent également être associés à des relations temporelles de façon à permettre à UAN et XUAN de représenter complètement les interruptions.

Tâche : Téléphoner

Décrocher

Insérer carte

(Composer N° **<-i** Reprise erreur N°)

(Communiquer <-|| (Changer carte vide)*)

(Raccrocher & Retirer carte)

Figure 12 : Tâche Téléphoner formalisée à l'aide de UAN étendu aux interruptions

CONCLUSION

ISAU a apporté de premiers éléments de réponse à la formalisation des interruptions. Il définit une modèlisation des interruptions et augmente la puissance d'expression des formalismes de scripts de tâche existants, en particulier UAN, en leur ajoutant de nouvelles relations temporelles. Ces ajouts nous ont permis de décrire avec simplicité notre scénario d'étude, le publiphone.

Néanmoins, nous avons laissé de côté l'étude des niveaux d'interruption et en corollaire le problème de l'imbrication des tâches d'interruptions. De même, nous n'avons pas étudié les retours d'information en provenance de l'interface qui sont pourtant l'une des clés du déclenchement des interruptions.

De plus, d'autres problèmes plus fondamentaux apparaissent lorsque l'on cherche à décrire une interface où les agents en présence ont des comportements assez indépendants, c'est le cas en particulier du collecticiel. Le principe « action d'un agent ENTRAÎNE réaction de l'autre agent » n'est alors plus respecté systématiquement. XUAN tente de remédier à ce problème [6] en prenant pour point de départ les relations temporelles définies par J. Allen [1]. Mais ce n'est pas suffisant et d'autres extensions doivent être effectuées.

REMERCIEMENTS

Cette étude à été financée par l'étude DGA/DRET 93-419 intitulé "Besoins en formalisation pour le dialogue multimodal" réalisé en collaboration avec ICP-INPG et Thomson-CSF.

BIBLIOGRAPHIE

1. Allen, J.F. Maintaning knowledge about temporal interval. *Communications of the ACM*. 26, 11 (November 1983), 832-843.

2. Balbo, S. *Un pas vers l'évaluation automatique des interfaces homme-machine*. Thèse en Informatique : Univerité Joseph Fourier (Grenoble I), 1994.

3. Barthet, M.-F. *Logiciels interactifs et ergonomie : Modèles et méthodes de conception*. Dunod, Paris, France, 1988.

4. Boy, G. The block representation in knowledge acquisition for computer integrated documentation. In *Proceedings of 4th AIAA-sponsored knowledge acquisition integrated documentation for knowledge-based systems workshop,* 1989.

5. Brun, P. and Beaudoin-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),* 1995, à paraître.

6. Gray, P., England, D. and McGowan, S. *XUAN: Enhancing the UAN to capture temporal relation among actions*. Departement of computing science, University of Glasgow, February 1994, research report IS-94-02.

7. Hartson, R.H. and Gray, P.D. Temporal aspects of tasks in the User Action Notation. *Human Computer Interaction*, Lawrence Erlbaum Associates, 1992.

8. Jambon, F. and Karsenty, L. Formalisation des interfaces et travail coopératif : Quelles conséquences ? In *Proceedings of Sixièmes Journées sur l'Ingénierie des Interfaces Homme-Machine (IHM'94),* 1994, pp. 163-168.

9. Mathé, N. *Assistance intelligente au contrôle de processus : Application à la télémanipulation Spatiale*. Thèse en Intelligence Artificielle : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1990.

10. Pleczon, P. *Éléments de méthodologie et outils pour l'assistance à l'opérateur : Application à la conduite automobile*. Thèse en Informatique : École Nationale Supérieure de l'Aéronautique et de l'Espace, 1992.

11. Rasmussen, J. *Information processing and Human-Machine Interaction : An approach to cognitive engineering*. North-Holland, 1986.

12. Reason, J. *L'erreur humaine*. Presses Universitaires de France, Paris, France, 1993.

13. Scapin, D.L. and Pierret-Golbreich, C. MAD : Une méthode analytique de description des tâches. In *Proceedings of Colloque sur l'Ingénierie des Interfaces Homme-Machine (IHM'89),* 1989, pp. 131-148.

14. Sebillotte, S., Alonso, B., Fallah, D., Hamouche, H. and Scapin, D.L. *Note de recherche concernant le formalisme MAD*. INRIA Rocquencourt, Novembre 1994, Document interne.

15. Sebilotte, S. Décrire des tâches selon les objectifs des opérateurs : De l'interview à la formalisation. *Le Travail humain*, Presses Universitaires de France, Paris, France, 1991.

16. Walker, W. and Cragon, H.G. Interrupt processing in concurent processors. *Computer*. 28, 6 (June 1995), 36-45.