Un modèle de dialogue

à base de scripts

1. Introduction

Dans ce chapitre le lecteur trouvera un exemple détaillé de la réalisation d'un dialogue homme machine multimodal pour un logiciel de dessin (ICPdraw = Interaction et Communication Parlée pour le dessin). Le modèle de dialogue adopté est fondé sur les scripts de dialogue et le modèle de tâche sur les scripts de tâche. Il s'agit d'un cas d'école, volontairement simplifié pour en faciliter la compréhension.

ICPdraw est une application de dessin (type MacDraw™) multimodale [Caelen, 92a]. L'utilisateur dispose d'une palette d'outils graphiques et de menus de fonctions et il dispose d'un langage d'interaction proprement multimodal (parole, clavier, geste à la souris). Le logiciel prévoit l'ensemble des fonctions habituelles nécessaires au dessin : sélection de plusieurs objets par pointage ou entourage ou désignation verbale, déplacement des objets, changement des coloris, dessin d'objets géométriques, etc. Ces objets géométriques peuvent être groupés, cachés ou saisis par des "poignées". Le contexte interactionnel est synergique (c'est-à-dire qu'on peut parler en même temps que l'on agit avec la souris, par exemple énoncer la commande "dessine un cercle ici" accompagnée d'un clic souris de désignation du lieu).

Les explications, les demandes de confirmation, les demande de précision et les aides sont fournies à l'utilisateur par synthèse vocale ou par affichage textuel dans la zone de dialogue ou sur des fenêtres volatiles lorsque la réponse est longue. Un aperçu général de l'interface est donné fig. 1.

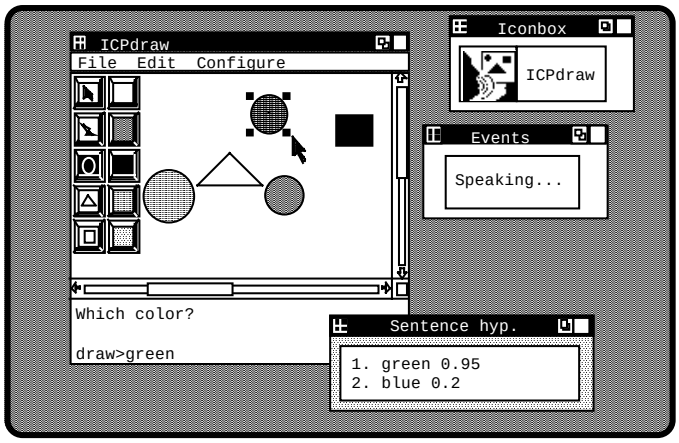


Fig. 1 : L'interafce graphique d'ICPdraw (version Xwindows). Elle est composée de quatre fenêtres, la première (ICPdraw) est découpée en zones et menus définissant l'espace de travail graphique et l'espace de dialogue, la deuxième (Iconbox) contient un logo, la troisième (Events) visualise l'état des canaux de communication et indique, par exemple, à l'utilisateur quand il peut parler, la

quatrième (Sentence hyp.) visualise les résultats du système de reconnaissance vocale (les 4 meilleures hypothèses sont retenues et rangées selon un score décroissant).

2. Les langages d'interaction d'ICPdraw

On distingue dans ICPdraw quatre types de langages d'interaction :

- le langage d'interaction verbale pour la manipulation des objets de l'application,
- le langage d'interaction gestuelle, qui est une extension à la désignation par pointage et à la sélection de menus et d'outils,
- le langage de demande d'aide, permettant à l'utilisateur de demander des aides en cours de travail (comment faire, comment dire, etc.),
- le langage de réplique qui lui est autorisé en réponse aux questions de la machine.

Le langage de manipulation

Le langage interne de manipulation des objets d'ICPdraw est défini comme une liste de commandes dont la forme logique est :

Com(<arg_1><arg_2>...<arg_n>)

Com représente une commande (action élémentaire),

arg_i sont les arguments de la commande.

A ce langage interne correspond un langage externe multimodal (à destination de l'utilisateur), à deux composantes, langagière et gestuelle.

La composante langagière

Les énoncés admissibles dans le langage oral sont des énoncés :

- bien formés comme : "dessiner un cercle vert ici"
- avec ellipse des mots grammaticaux comme : "dessiner cercle vert ici"
- de réitération comme : "encore" ou "...cercle vert"
- de rectification comme : "non... vert"
- de confirmation comme : "oui"

Ils peuvent être décrits par la grammaire Go (avec une signification évidente des notations, | opérateur "ou", . opérateur "et", () élément facultatif, {} liste) :

Go

Com	->	Action | Réitération | Rectification
Action	->	Vo.(GN).(GPL) | Vo.Pr.(GPL)
Action	->	Vi
Réitération	->	GN | GPL | AdjC
Réitération	->	{encore}
Rectification	->	non.GN | non.GPL | non.AdjC

Rectification	->	plus.AdjT
Tâche	->	Vo.(GT)
GT	->	(Dét.).NT
GN	->	(Dét).(AdjT).N.(AdjC) | Dét.AdjT | Dét.AdjC
GPL	->	LocP | LocA
Pr	->	PP | PD
PP	->	{le, les}
PD	->	{çà}
Vo	->	{dessiner, déplacer, effacer, dupliquer}
Vi	->	{annuler, quitter}
Dét	->	{le, les, un, deux, trois, quatre, ce, ces}
AdjT	->	{grand, petit}
N	->	{carré, cercle, triangle, trait}
NT	->	{noms des tâches répertoriées}
AdjC	->	{blanc, noir, bleu, jaune, rouge, vert}
LocP	->	{à droite, à gauche, en haut, en bas, au centre}
LocA	->	{ici, là, vers là, par ici, par là}

Le module qui traite des entrées en langage naturel (parlées ou écrites) dispose d'analyseurs linguistiques capables de fournir la structure des constituants (c-structure) et la structure fonctionnelle (f-structure) de la commande. Ces analyseurs utilisent le formalisme des grammaires lexicales fonctionnelles [Reynier, 90] et fonctionnent avec le système de reconnaissance automatique de parole continue ECHO (voir annexes). ECHO utilise des modèles de Markov cachés.

La composante gestuelle

Le langage gestuel comprend des actions de désignation/sélection et des actions de dessin à main levée. Il se formalise par la grammaire Gg :

Gg

Définitions		
A1	:	enfoncement du bouton n°1 de la souris
A2	:	enfoncement du bouton n°2 de la souris
R	:	relâchement d'un bouton
Traj	:	trajectoire du curseur de la souris
A1(Lieu).R	:	désigne un lieu de la zone de travail
A1(Lieu).Traj.R	:	désigne une aire de la zone de travail

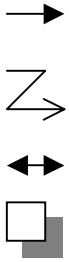
Grammaire Gg :

Action -> A1(Obj).R | A1(Lieu).Traj.R (sélection d' objet(s))

Action -> A1(palette_carré).R.A1(Lieu).R (dessiner un carré)
Action -> A2(Lieu).Traj(carré).R (dessine un carré)

Action -> A1(palette_triangle).R.A1(Lieu).R (dessiner un triangle)
Action -> Traj(triangle).R (dessiner un triangle)

Action -> A1(palette_cercle).R.A1(Lieu).R (dessiner un cercle)
Action -> Traj(cercle).R (dessiner un cercle)



Action -> (Désigne(obj)).A1(palette_kill).R (effacer cet objet)	
Action -> Traj(Z) | Traj(α)).R (effacer cet objet)	
Action -> A1(Obj).Traj.R (déplacer cet objet)	
Action -> Traj.R (dupliquer cet objet)	

La racine commune aux deux grammaires Go et Gg est l'entrée "Com". L'union des deux grammaires en une seule définit la grammaire multimodale Gm = Go U Gg.

Le langage de demande d'aide

Le langage de demande d'aide définit l'ensemble des énoncés de demande d'aide que peut formuler l'utilisateur. L'aide est activée par le contrôleur de dialogue ou sur demande de l'utilisateur. Cette aide est présentée oralement (synthèse de la parole) ou de manière écrite (dans une fenêtre volatile). Il ne s'agit pas ici pour la machine de générer des énoncés complexes et extrêmement variés : on peut simplement utiliser des phrases "à trous" dans lesquelles sont insérés les éléments variables.

La demande d'aide porte sur :

- les objets de l'application (type, couleur, position),
- les opérations possibles sur les objets,
- les langages d'interaction,
- l'aide elle-même.

La grammaire du langage de demande d'aide Ga est la suivante :

Ga

Dem_aide	->	Dem_aide_obj | Dem_aide_op | Dem_aide_l | Dem_aide_aide
Dem_aide_obj	->	quel.N | quelles.sont.les.couleurs
Dem_aide_obj	->	quelle.est.la.couleur.du.N
Dem_aide_obj	->	quelle.est.la.taille.du.N
Dem_aide_obj	->	où.est.le.N
Dem_aide_op	->	comment.Vi | comment.Vo | comment.V
Dem_aide_op	->	qu'est.ce.que.LocP
Dem_aide_op	->	qu'est.ce.que.LocA
Dem_aide_l	->	quel.est.le.vocabulaire
Dem_aide_l	->	quels.sont.les.gestes
Dem_aide_aide	->	aide
V	->	{réitérer, rectifier, corriger, confirmer}
Vo	->	{dessiner, déplacer, effacer, dupliquer}
Vi	->	{annuler, quitter}
N	->	{carré, cercle, triangle, trait}
LocP	->	{à droite, à gauche, en haut, en bas, au centre}
LocA	->	{ici, là, vers là, par ici, par là}

Ce langage est assez sommaire car il n'a d'intérêt que pour le début de l'apprentissage de l'utilisateur qui a surtout besoin de savoir ce qu'il peut faire, comment le faire et comment l'exprimer. Il y a des demandes de précison comme "qu'est.ce.que.LocP" qui lui permettent de avoir ce que signifie exactement "à droite", "à gauche", etc.

Le langage de réplique

Dans les cas d'ambiguïté de commande, d'imprécision, d'erreur, d'incompréhension, etc. la machine est amenée à poser des questions à l'utilisateur. De la part de la machine le langage est figé mais les réponses de l'utilisateur peuvent être relativement variées (mais on peut fermer le nombre de réponses en choisissant judicieusement la forme des questions). Elles sont régies par une grammaire dite des répliques, sensible au contexte de la question. Ces contextes sont :

- Clarification, confirmation
- Réparation des erreurs,
- Demande de précision,
- Incompréhension.

Carification, confirmation *Grammaire des répliques Gr*

Qc1 : "Quel objet parmi ceux-ci ?" + clignotement R -> GN | {celui-là, celui-ci}
Qc2 : "L'objet précédent ?" R -> {oui, non}
Qc3 : "Est-ce d'accord ?" R -> {oui, non}
etc.

Réparation des erreurs

Qe1 : "Je ne peux pas exécuter votre commande"
Qe2 : "Désolé, je ne peux rien faire"
Qe3 : "Désolé, je ne sais pas répondre à la question"
Qe4 : "Voulez-vous annuler ?" R -> {oui, non}
etc.

Demande de précision

Qo : "Quel objet ?" R -> GN | {celui-là, celui-ci}
Qc : "De quelle couleur ?" R -> AdjC
Qt : "De quelle taille ?" R -> AdjT
Ql : "Où ?" R -> LocP | LocA
etc.

Incompréhension

Qi1 : "Pouvez-vous répéter ?" R -> Enoncé_com| Dem_aide
Qi2 : "Pouvez-vous reformuler votre demande ?" R -> Dem_aide
Qi3 : "Pouvez-vous reformuler votre commande ?" R -> Dem_com
etc.

Le modèle de tâches

On ne peut pas dire que dans les *tâches de conception* comme le dessin, il y ait un véritable modèle de tâche. En effet le dessin ne peut être une activité planifiée (sauf s'il s'agit de reproduire un dessin et non de le créer), il favorise au contraire une activité de type opportuniste. Le modèle de tâches peut donc se ramener à un ensemble de scripts dans lequel il faut laisser à l'utilisateur un contrôle le plus souple possible. On a donc défini dans ICPdraw un modèle hiérarchisé seulement à deux niveaux : les scripts et les scénarios.

La syntaxe des scripts et des scénarios

Convenons d'une syntaxe simple pour décrire un langage de scripts (cela nous évitera d'utiliser une écriture plus lourde liée à un langage de programmation).

Un script (et un scénario) a la structure suivante :

Type : Nom(ARGUMENTS)
précondition
corps du script
effet

La précondition est une formule qui doit être vérifiée pour pouvoir activer le corps du script. Le corps du script contient des instructions ou des arguments qui recevront une valeur au moment de l'activation du script. L'effet est un appel à un autre script qui sera lui-même exécuté si sa précondition le permet. Les conventions de notations sont les suivantes :

Les actions (ou fonctions) sont directement associées aux langages d'interactions, par exemple Vo signifie un verbe de la catégorie Vo.

Les opérateurs :
% indique un prototype ou un nom de classe
$ est une valeur d'une variable
= est l'affectation d'une valeur au membre gauche
| est le connecteur "ou" non exclusif
. est le connecteur "et"
>> est un lien entre informations multimodales

Les scripts élémentaires

Les actions de base du système sont en nombre réduit (pour simplifier l'exposé). Elles portent sur les objets élémentaires (carré, cercle, trait, triangle). Ces actions et ces objets sont décrits à l'aide de scripts avec les conventions décrites ci-dessus :

La base des scripts élémentaires est :

```
Action : Sélectionner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
  Activation = A1(obj).R | A1(zt).traj.R
  OBJ = OBJ | $obj | $traj
  TAILLE = TAILLE
  COUL = COUL
  VISIBLE = VISIBLE
  SELECT = 1

Action : Dé-Sélectionner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
  Activation = A1(zt).R
  OBJ = Tout(obj)
```

```
   TAILLE = TAILLE
   COUL = COUL
   VISIBLE = VISIBLE
   SELECT = 0

Action : Dessiner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
   Activation = Vo(%dessiner) | A2(zt).traj(X).R | A1(palette).R.A1(zt).R
   OBJ = OBJ | $N | $PP | $PD | $X | $palette
   LIEU = LIEU | $LocP | $LocA=A1($zt) | $défaut
   TAILLE = TAILLE | $AdjT | $traj | $défaut
   COUL = COUL | $AdjC | $palette | $défaut
   VISIBLE =1
   SELECT = 1

Action : Effacer (OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
   Activation = Vo(%efface) | A2(obj).traj(Z).R | A2(obj).traj(α).R |
A1(palette_kill).R.Sélectionner(OBJ)
   OBJ = OBJ | $N | $PP | $PD | $obj
   LIEU = LIEU
   TAILLE = TAILLE
   COUL = COUL
   VISIBLE = 0
   SELECT = 1

Action : Agrandir(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
   Activation = A1(poignée(OBJ)).traj.R
   OBJ = OBJ
   LIEU = LIEU
   TAILLE = TAILLE | $AdjT | $traj
   COUL = COUL
   VISIBLE = 1
   SELECT = 1

Action : Déplacer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
   Activation = Vo(%déplace) | A1(obj).traj.R
   OBJ = $N | $PP | $PD | $obj
   LIEU = $LocP | $fin(traj) | $LocA=A1($Lieu)
   TAILLE = TAILLE
   COUL = COUL
   VISIBLE = 1
   SELECT =1

Action : Dupliquer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
   Activation = Vo(%duplique) | A2(obj).traj.R
   OBJ = $N | $PP | $PD | $obj
   LIEU = $LocP | $fin(traj) | $LocA=A1($Lieu)
   TAILLE = TAILLE
   COUL = COUL
   VISIBLE = 1
   SELECT =1

Action : Quitter
   Activation = Vi(%quitte) | A1(menu_quit)
```

La base des objets et des atrributs est :

```
Triangle : Sorte_de OBJ
   Nom = triangle
   Position = LIEU
   Taille : TAILLE
   Couleur : COUL
   Visible = VISIBLE
   Sélection =  SELECT

Carré : Sorte_de OBJ
   Nom = carré
   Position = LIEU
   Taille : TAILLE
```

```
   Couleur : COUL
   Visible = VISIBLE
   Sélection =  SELECT

Cercle : Sorte_de OBJ
   Nom = cercle
   Position = LIEU
   Taille = TAILLE
   Couleur = COUL
   Visible = VISIBLE
   Sélection =  SELECT

Trait : Sorte_de OBJ
   Nom = trait
   Position = LIEU
   Taille = TAILLE
   Couleur = COUL
   Visible = VISIBLE
   Sélection =  SELECT

Position : Sorte_de ATTRIBUT
   LIEU = (x, y)

Taille : Sorte_de ATTRIBUT
   TAILLE = {grand, petit}

Couleur : Sorte_de ATTRIBUT
   COUL = {rouge, jaune, vert, bleu, blanc, noir}

Visible : Sorte_de ATTRIBUT
   VISIBLE = {0, 1}

Sélection : Sorte_de ATTRIBUT
   SELECT = {0, 1}
```

Les éléments déictiques et anaphoriques sont définis par :

```
le :            Anaphore >> OBJ(SELECT=1)
                Cataphore >> A1(Obj).R | A1(Lieu).Traj.R

les :           Anaphore >> {OBJ(SELECT=1)}
                Cataphore >> A1(Lieu).Traj.R

çà :            Déictique >> A1(Obj).R | A1(Lieu).Traj.R

ici, là :       Déictique >> A1(Lieuj).R | A1(Lieu).Traj.R

vers là, par ici, par là :          Déictique >> A1(Lieu).Traj.R

ce :            Déictique >> (GN($N) | GN($AdjT) | GN($AdjC)).(A1(Obj).R | A1(Lieu).Traj.R)

ces :           Déictique >> (GN($N) | GN($AdjT) | GN($AdjC)).A1(Lieu).Traj.R

le, un :        Dét >> GN($N) | OBJ($AdjT) | OBJ($AdjC) | A1(Obj).R | A1(Lieu).Traj.R

les,{nbre} :  Dét >> GN($N) | OBJ($AdjT) | OBJ($AdjC) | A1(Lieu).Traj.R

à droite, à gauche, en haut, en bas, au centre :  Dét >> Zone_travail
```

Les scénarios

Les scénarios sont des scripts de niveau supérieur qui permettent d'encapsuler les actions de base. Ils obéissent aux mêmes règles de syntaxe (la variable HIST définit un pointeur sur l'historique de la tâche).

Base de scénarios

```
Action : Annuler(HIST)
   Activation = Vi(%annuler) | A1(menu_undo)
   HIST = Dessiner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) | Agrandir(OBJ, LIEU,
TAILLE, COUL, VISIBLE, SELECT) | Déplacer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
| Dupliquer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) | Sélectionner(OBJ, LIEU, TAILLE,
COUL, VISIBLE, SELECT) | Dé-Sélectionner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) |
Effacer (OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)

Action : Commande-Action(HIST)
   Activation = Com
   HIST = Dessiner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) | Agrandir(OBJ, LIEU,
TAILLE, COUL, VISIBLE, SELECT) | Déplacer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
| Dupliquer(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) | Sélectionner(OBJ, LIEU, TAILLE,
COUL, VISIBLE, SELECT) | Dé-Sélectionner(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) |
Effacer (OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT) | Annuler | Quitter

Action : Demande-aide(HIST)
   Activation = Dem_aide
   HIST = Aide_obj(N, LocP, LocA) | Aide_op(V, Vi, Vo) | Aide_l | Aide_aide

Action : Introduction
   "Bienvenue sur ICPdraw"

Action : Aide_obj              (non décrit)
```

Il est très facile d'écrire les scripts d'aide (Aide_obj, Aide_op, Aide_l, Aide_aide), nous ne les détaillons pas ici.

Le contrôleur de dialogue d'ICPdraw

Le contrôleur de dialogue est normalement chargé (voir le chapitre 2) :

- de l'organisation de la communication (gestion des tours de parole),
- du contrôle dynamique de l'interaction,
- du choix des stratégies de dialogue (réactivité, négociation, coopérativité, etc.)
- de la réparation des erreurs de communication,
- de l'aide à l'apprentissage ou de l'apprentissage lui-même,
- des aides dans la tâche et de la conduite des activités.

Nous présentons ci-après deux variantes de contrôle du dialogue, l'une sans gestion de stratégies, sans apprentissage et sans demande d'aide, l'autre avec ces trois aptitudes.

Un premier modèle

Ce premier modèle est à stratégie entièrement réactive. Il est implémenté à l'aide d'un automate d'états finis (fig. 1). Un tel automate est équivalent à une grammaire à contexte libre, ce qui classe le modèle dans la catégorie des modèles structurels (voir chap. 2). Remarquons qu'il est d'ailleurs plus proche d'un modèle d'interaction (de type manipulation directe multimodale) que d'un véritable modèle de dialogue.

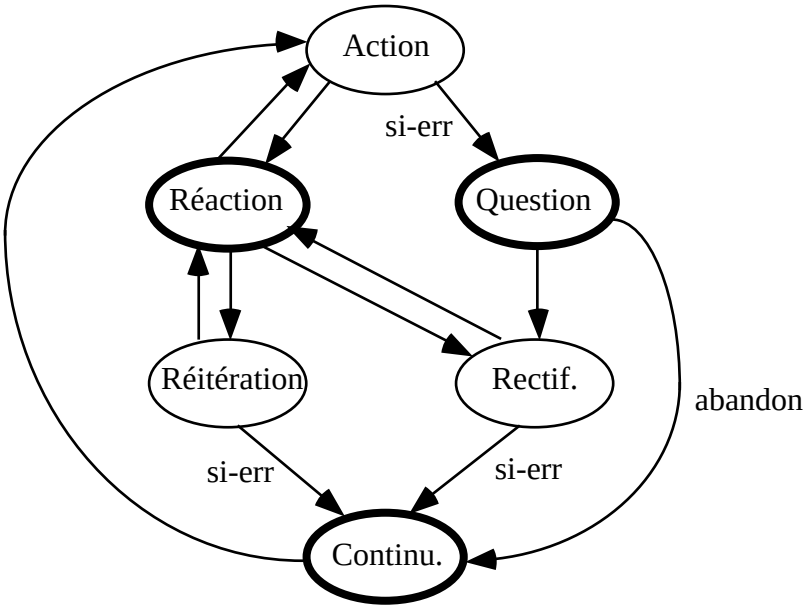


Fig. 1 : Le modèle d'interaction

Il fonctionne simplement de la manière suivante :

- si l'action est correctement interprétée et exécutable alors le système passe dans l'état "réaction" (il exécute la commande de l'utilisateur) et attend une nouvelle commande d'action, de réitération ou de rectification,
- si l'action conduit à une situation incohérente, le système pose une question et se met en attente d'une réponse de rectification. Deux erreurs consécutives ne sont pas tolérées, le système passe dans l'état "continuation" c'est-à-dire en attente d'une nouvelle action. Les incidences du dialogue sont volontairement évitées, elles sont traitées comme des abandons implicites (avec retour à une nouvelle action).

Ce qui conduit à l'algorithme suivant :

```
Dialogue
        Introduction
        HIST=Ø
        Continuation : Attente(Acte)
        Interprétation(Acte, Com)
        TantQue Com ≠ Quitter Faire
                si erreur alors
                        Analyse (erreur, Question(Q))
                        Attente(Acte)
                        Interprétation(Acte, Com)
                                si erreur alors aller_à Continuation
                                finsi
                finsi
                Commande-action(HIST)
                Continuation : Attente(Acte)
                Interprétation(Acte, Com)
        finTantQua
FinDialogue
Interprétation(X, Y) : procédure qui sélectionne le script Y pour un acte de dialogue Acte
Analyse(erreur, X) : en fonction du type d'erreur la machine fait l'action X
Question(Q) : la machine pose la question Q
Réaction : exécution de l'acte et attente d'un nouvel acte
Continuation : attente d'un nouvel acte
```

Dans ce modèle, la commande est exécutée chaque fois que c'est possible, même si la commande est incomplètement spécifiée (et donc le résultat est immédiatement perceptible), sinon le système reste sans réaction.

Un deuxième modèle

Dans ce deuxième modèle, les stratégies sont plus variées et les aides mieux adaptées. Ce modèle intègre aussi l'apprentissage d'une séquence d'actions comme une tâche. Pour tenir compte de l'apprentissage il faut ajouter l'action suivante dans la base des actions élémentaires :

```
Action : Apprentissage(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
        Activation = erreur
```

Cette action permet de déclencher une procédure d'apprentissage (certes complexe ! mais non traitée dans ce chapitre) pour construire un nouvel objet OBJET (cela suppose que ce nouvel OBJET est rajouté dans la base). L'algorithme de ce deuxième modèle de dialogue est maintenant :

Notations :

U = usager {aidé=1, non-aidé=0}
π = profondeur du dialogue = nombre de tours de parole {nombre entier}
σ = type de stratégie utilisée {réactive, coopérative}
seuil = seuil arbitraire à partir duquel on change de stratégie

```
Début
        Introduction
        Usage(σ, U)
        seuil = 20
        HIST = Ø
        π= 0
        Acq(A)
        si erreur alors Apprentissage(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
        finsi
        Tantque A ≠ Quitter
                cas  A = Commande_action(HIST)
                cas  A = Demande_aide(HIST) ; U = 1
                cas A = Ø ; Parole("Pouvez-vous reformuler, je n'ai pas compris")
                si π > seuil alors σ = réact
                finSi
                Acq(A)
                si erreur alors Apprentissage(OBJ, LIEU, TAILLE, COUL, VISIBLE, SELECT)
        FinTantque
        Clôture
Fin
```

```
Acq(A)
        Attente(Acte)
        Interprétation(Acte, A)
        si U alors Explication(A)   si σ = coop      alors Qc(A)
                                                    Attente(Acte)
                                                    π= π + 1
                                                    Interprétation(Acte,B)
                                                    si B = %non alors erreur
                                                    finsi
                                    finsi
        finsi
fin
```

Les scénarios à ajouter sont :

```
Introduction
        Window_texte(“ Bienvenue sur ICPdraw ”)
        Parole(“ Bienvenue sur ICPdraw ”)
Fin
```

```
Usage(∑, U)
        Parole(“ Connaissez-vous le système ? ”)
        Attente(Acte)
        Interprétation(Acte, A)
        si A = %oui’
                alors σ = réact.
                sinon σ = coop ; Icone(visage) ; Parole(“ Je vais vous décrire brièvement le système, mais vous pouvez m’interrompre en cliquant sur mon icône. ICPdraw est un logiciel de dessin multimodal qui interprète des commandes verbales et gestuelles. Il fonctionne aussi comme un éditeur graphique, à l’aide des menus et de la palette. Les actions verbales sont énoncées à l’impératif. Par exemple, dessine un cercle vert en haut. Les actions gestuelles sont des gestes esquissés à la souris, elles se font en appuyant sur le bouton du milieu ”)
        finsi
        Parole(“ Voulez-vous une aide permanente ? ”)
        Attente(Acte)
        Interprétation(Acte,A)
        si A = %oui’
                alors U = 1
                sinon U =0
        Parole(“ Maintenant vous pouvez travailler ”)
Fin
```

```
Clôture
        Parole(“Voulez-vous sauvegarder le dessin ?)
        Attente(Acte)
        Interprétation(Acte, A)
        si A = %oui’ alors Sauvegarde
        finsi
        Parole (“au revoir”)
fin
```

Commentaires

La description de ces deux modèles n’est évidemment pas suffisamment détaillée pour passer directement à la programmation. Il y aurait lieu de décrire plus précisément les procédures d’exécution des scripts et des scénarios. Mais là n’était pas le but de ce chapitre.

On remarquera dans le deuxième modèle que :

a) la stratégie mise en œuvre est dynamique. Elle dépend du choix volontaire de l’utilisateur et de la profondeur du dialogue (vers la fin du dialogue on revient systématiquement à une stratégie réactive),

b) l’offre d’aide est aussi dynamique, comme pour la stratégie, elle est librement choisie par l’usager ou imposée s’il fait trop d’erreurs.

c) la procédure d’apprentissage est activée par les erreurs de la machine : on suppose que ces erreurs sont engendrées par le fait qu’elle ne sait pas faire ce qu’on lui demande.

Evidemment ces trois points ne sont pas des lois pour le dialogue, ce ne sont que des exemples de ce qu’il est possible de faire.

Conclusion

ICPdraw est une application de démonstration qui a réalisée sur une station de travail Silicon Graphics Indy avec un système de reconnaissance vocale en mots enchaînés, ECHO, mis au point au laboratoire CLIPS-IMAG (et précédemment à l'ICP dans le cadre du projet Multiworks ESPRIT II n°2105). Le vocabulaire utilisé est d'une cinquantaine de mots, la syntaxe est limitée comme indiqué dans ce chapitre.

Le dialogue est en temps réel. Cette plate-forme permet de tester et de valider (a) les concepts du dialogue multimodal ainsi que ceux (b) d'une architecture répartie. Elle a permis de faire émerger le problème de la désignation multimodale — notamment dans le cas ou les objets sont animés (problème de la référence perdue lorsque l'on prononce la commande "détruis le cercle du bas" pendant que l'on déplace ce cercle avec la souris) — et d'initialiser les études sur l'usage des modalités. En effet on ne maîtrise pas encore les situations dans lesquelles les usagers utiliseront préférentiellement tel mode ou tel autre et s'ils auront tendance à se contenter d'habitudes appauvrissantes vis-à-vis de l'offre multimodale.

L'usage du multimodal produit des effets de bord car les commandes peuvent devenir ambiguës en se recouvrant dans le temps : une certaine désynchronisation du geste et de la parole, des phénomènes de répétition, etc. augmentent avec la charge cognitive ou la charge de travail. L'utilisateur tend à spécialiser les modes (la parole pour des commandes répétitives ou qui ne nécessitent pas la mobilisation du regard, ou la précision du geste c'est-à-dire exigeant une planification motrice ou perceptive importantes) et le geste pour des actions immédiates et élémentaires à forte réactivité.

Après une étude restreinte, il apparaît que cette interface améliore l'efficacité globale de l'usager en lui permettant d'exécuter plusieurs commandes en même temps : c'est le parallélisme des commandes qui lui permet notamment d'anticiper et de préparer les objets du dessin à l'avance. Ainsi on s'aperçoit que ses stratégies sont optimisées en fonction de ces nouvelles possibilités : par exemple il peut afficher pêle-mêle une série d'objets par commande vocale ("dessine un cercle noir", "un rouge", "un vert", etc.) puis dans un second temps il peut les positionner précisément à la souris (éventuellement en continuant à en afficher de nouveaux par la voix).

L'utilisateur n'atteint pas une telle expertise sans entraînement. Au bout d'un temps d'usage, il simplifie ses commandes en les rendant très brèves et souvent très elliptiques. C'est pourquoi nous avons choisi ce vocabulaire et cette syntaxe qui ont pu paraître très simples.