DESS GI Examen mardi 19 décembre 2000

Interaction Homme-Machine

Durée : 3 heures — Tout document permis sauf livre

Si le sujet présente des ambiguïtés, précisez vos choix. Il sera tenu compte de vos hypothèses. Toutes les questions sont indépendantes. Le barème est indicatif et vous donne une indication du temps à passer par question (1 point ~ 9 minutes).

A- Modélisation de l'expertise d'un utilisateur (3 points)

Comme le montre la Figure 1, le modèle de Rasmussen décrit trois niveaux de contrôle des actions humaines :
(1) Comportement basé sur les habiletés
(2) Comportement basé sur les règles
(3) Comportement basé sur les connaissances
Ce modèle est à l'origine de la classification des utilisateurs en trois types : expert (1), occasionnel (2) et novice (3).

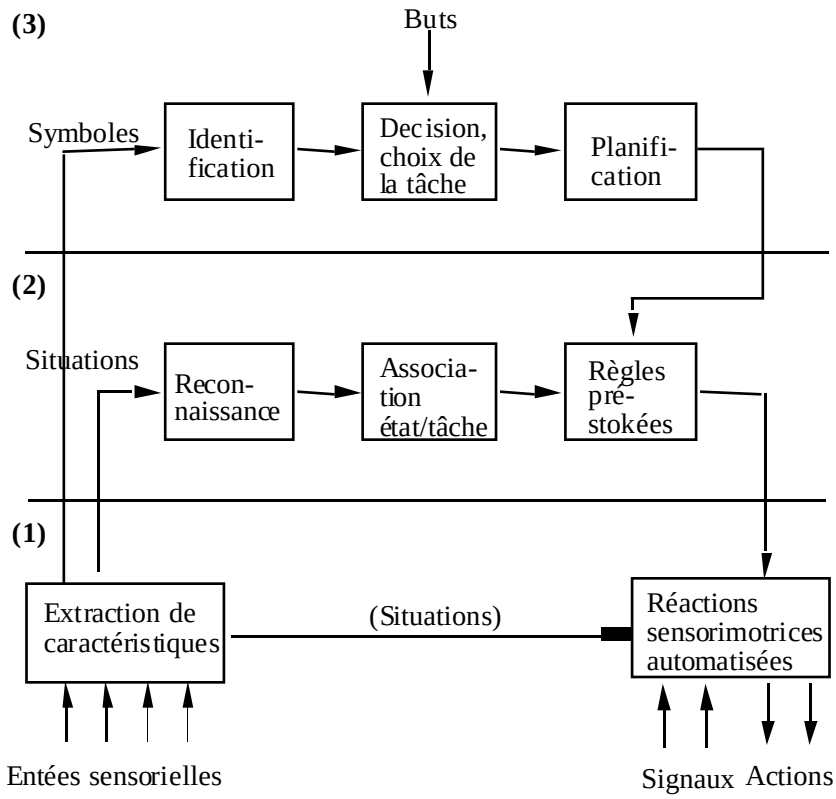

Figure 1 : 3 niveaux d'expertise selon Rasmussen.

Question A-1 (1,5 point)
Modéliser les trois niveaux d'expertise du modèle de Rasmussen en appliquant la Théorie de l'Action de D. Norman (Etapes dans l'accomplissement d'une tâche). Expliquez et justifiez votre réponse.

Question A-2 (1 point)
Modéliser les trois niveaux d'expertise du modèle de Rasmussen en appliquant la Théorie ICS. Expliquez et justifiez votre réponse.

Question A-3 (0,5 point)
Définir les connaissances procédurales et factuelles de l'utilisateur.
Donner un exemple pour les deux types de connaissance.

B- Utilisabilité (6,5 points)

Question B-1 (1 point)

Définir les distances articulatoires et sémantiques.
"La réduction des distances articulatoires et sémantiques améliore l'utilisabilité de l' IHM."
En vous appuyant sur la Théorie de l' Action (D. Norman), justifiez cette recommandation.

Question B-2 (1 point)

Les retours d'information sonores présentent des intérêts et des inconvénients.
Donner un point positif et un point négatif de l'usage de retour d'information sonore dans une interface.

Question B-3 (1 point)

Donner un exemple d'interface qui ne vérifie pas le critère d'observabilité. Justifier votre réponse et expliquer les problèmes engendrés par le non-respect de ce critère. Proposer une solution pour résoudre les problèmes identifiés.

Question B-4 (1,5 point)

La Figure 2 présente une copie d'écran d'Internet Explorer 4.0.
Donner trois critères d'ergonomie vérifiés ou transgressés par cette interface. Pour chaque critère cité, justifier votre réponse.

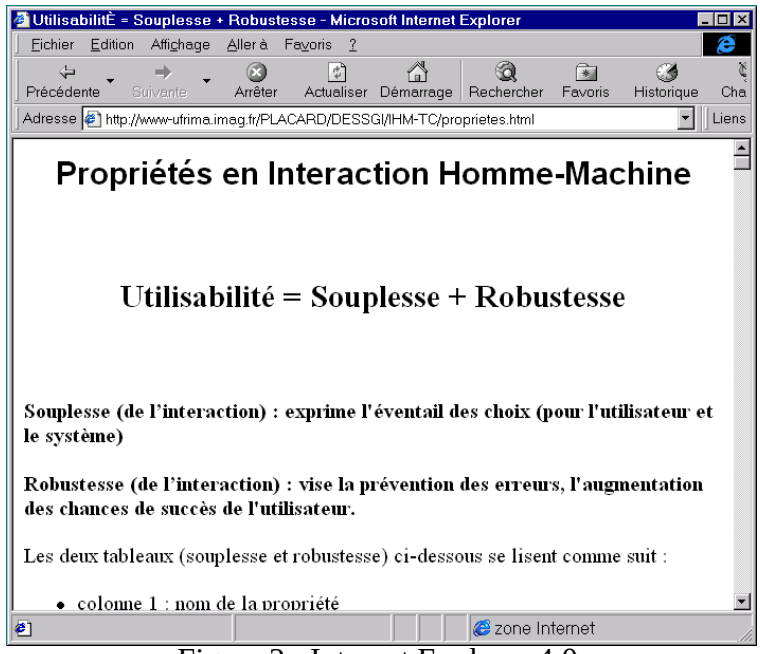

Figure 2 : Internet Explorer 4.0.

Question B-5 (2 points)

La Figure 3 présente une fenêtre contenant 9 icônes.

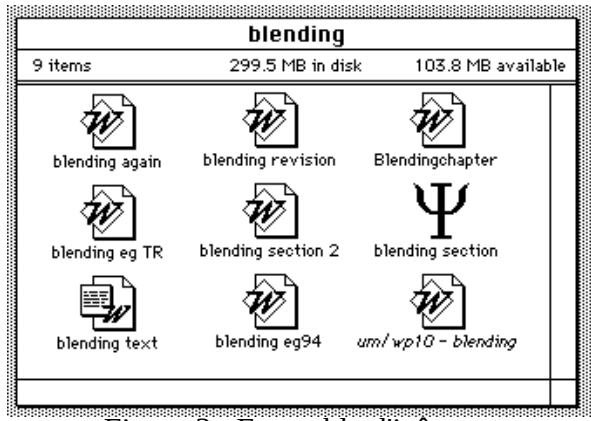

Figure 3 : Ensemble d'icônes.

- Construire le diagramme structurel (méthode « structuring the display ») de la solution proposée à la Figure 3.
- Annoncer quelle est la variable pragmatique. Justifier votre réponse.
- Construire le diagramme de transition (méthode « structuring the display ») dans le cas où la tâche de l'utilisateur consisterait à sélectionner l'icône intitulée "blending section 2".

C - Conception (4 points)

Nous considérons l'éditeur de dessin réalisé en TP avec le langage JAVA. La Figure 4 montre la version finale de l'éditeur de dessin.

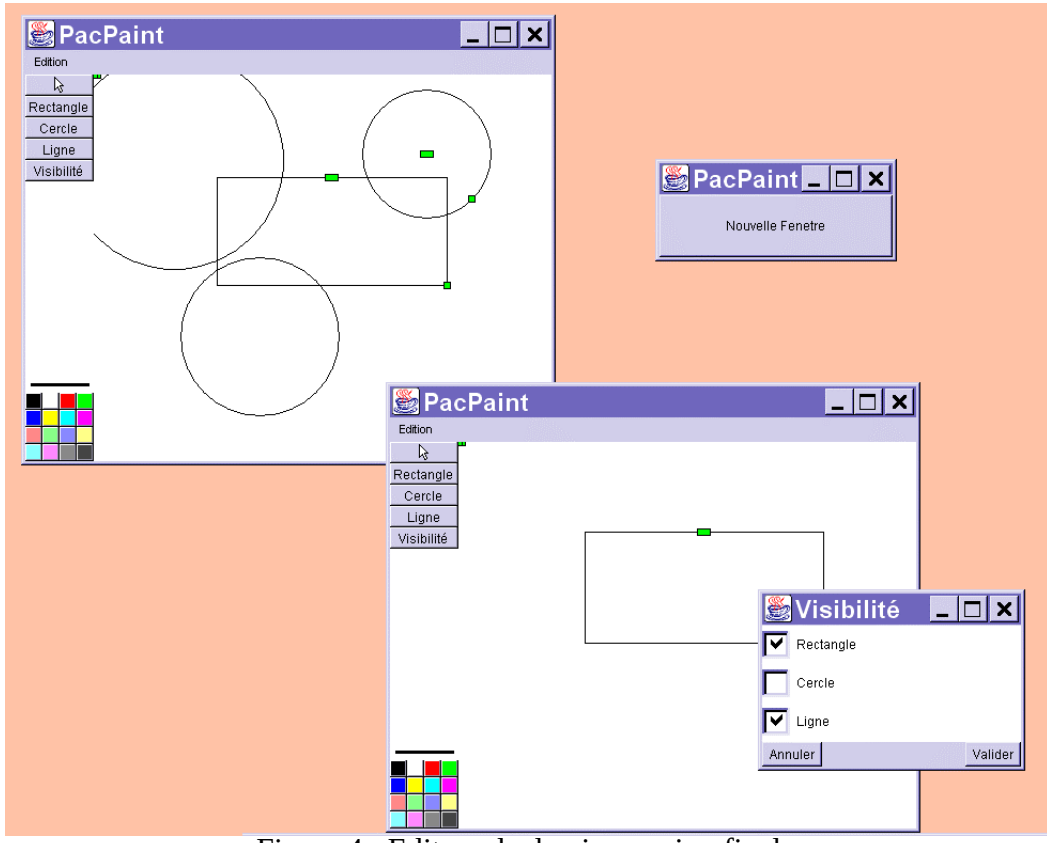

Figure 4 : Editeur de dessin, version finale.

Question C-1 (2 points)
Construire un arbre de tâches en HTA de l'éditeur de dessin version finale avec toutes les extensions, comme présentée à la Figure 4. Justifier votre solution.

Question C-2 (2 points)
Décrire en UAN le "retaillage" (changement de taille) d'un rectangle dans l'éditeur de dessin (Figure 4).
Rappel : Pour retailler un rectangle, l'utilisateur sélectionne dans la palette de dessin le mode "sélection" (icône Flèche) puis elle/il sélectionne le carré vert en bas à droite du rectangle à modifier. Tout en maintenant le bouton de la souris appuyé, elle/il déplace la souris. Le rectangle est alors déformé selon le mouvement de la souris, jusqu'à ce que l'utilisateur relâche le bouton de la souris.

D- Réalisation logicielle en JAVA d'un agent PAC (3.5 points)

Nous nous intéressons à étendre la réalisation logicielle d'un agent PAC du projet "Editeur de dessin" fait en TP avec le langage JAVA. L'Annexe 1 rappelle la description des trois classes réalisant un agent PAC.

Question D-1 (0,5 point)
Rappeler le rôle des méthodes newAction, newUserAction et newSystemAction dans la classe Control.

Question D-2 (1 point)

- Modifier les classes Abstraction et Presentation pour que la communication entre la facette Contrôle et les facettes Abstraction et Présentation soit localisée dans une méthode unique, respectivement newControlToAbstAction et newControlToPresAction.
- Modifier la classe Control en conséquence.

Question D-3 (2 points)

La communication entre deux agents PAC est actuellement synchrone. L'agent qui envoie un message à un de ses fils par exemple ne reprend le contrôle que lorsque le fils a fini de traiter l'événement envoyé. Dans le cas d'opération longue (comme l'affichage d'animations) il est intéressant d'avoir une communication asynchrone. Les agents travaillent alors en parallèle. Les deux types de communication (synchrone et asynchrone) sont utiles au sein d'une hiérarchie d'agents PAC.

- Modifier la classe Control pour rajouter la communication asynchrone : l'agent envoie un événement à un de ses fils et continue à fonctionner tandis que l'agent fils en parallèle traite l'événement reçu.
- Annoncer ensuite ce qu'il faut modifier dans les agents si l'on souhaite une communication asynchrone.

Information JAVA : Création d'un Thread

```
/* Methode synchrone */
public void testS (int val)
{
     System.out.println(val);
}
/* Creation d'une nouvelle méthode asynchrone */
public void testA (int val)
{
/* Creation d'un thread */
     Test thread1;
     thread1 = new Test (val);
     thread1.start();
}
class Test extends Thread
{
     private int v;

     public Test (int val)
     {
        v = val;
     }
     public void run()
     {
        testS (v);
     }
}
```

E- Architecture logicielle (3 points)

Nous nous intéressons à l'architecture logicielle de l'éditeur de dessin réalisé en TP avec le langage JAVA.

Nous partons de l'architecture logicielle de l'éditeur de dessin de base qui permet de créer des rectangles, des cercles et des lignes (sans aucune extension).

Question E-1 (1 point)

Nous considérons l'extension qui consiste à sauvegarder les dessins réalisés dans des fichiers.

- Expliquer où sont gérés les fichiers de dessin.
- Expliquer au sein de l'architecture logicielle l'ensemble des messages échangés lorsque l'utilisateur sauvegarde un dessin.

Question E-2 (1 point)
Nous considérons l'extension qui consiste à gérer les couleurs de contour des dessins, comme le montre la Figure 4.

- Dessiner l'architecture logicielle avec la gestion de la couleur. Expliquer où est mémorisé la couleur courante.
- Expliquer comment est géré la couleur en montrant l'ensemble des événements échangés au sein de l'architecture lorsque l'utilisateur change la couleur.

Question E-3 (1 point)
Nous considérons l'extension qui consiste à visualiser plusieurs fois le même dessin (vue multiple), comme le montre la Figure 4.

- Dessiner l'architecture logicielle pour gérer la vue multiple.
- Expliquer comment est géré la vue multiple en montrant l'ensemble des événements échangés au sein de l'architecture pour garantir la cohérence visuelle lorsque l'utilisateur vient de créer un nouveau rectangle dans une fenêtre alors que le dessin est aussi visualisé dans deux autres fenêtres.

Annexe 1 : Rappel, trois classes pour un agent PAC

```
package PacModel;
public class Control implements Cloneable
{
        protected Abstraction  Abstr = null;
        public Abstraction getA() {return Abstr;}
        protected Presentation Pres  = null;
        public Presentation getP() {return Pres;}
        protected Control PacFather  = null;
        protected String Name  = "";
        public String getName() {return Name;}
        private static int Counter = 0;
        public Control (Control Ctrl)
        {
              PacFather  = Ctrl;
              String ClassName = getClass().toString();
              ClassName = ClassName.substring(ClassName.lastIndexOf("."));
              if (ClassName.startsWith(".C_"))
                  Name = ClassName.substring(3)+Counter;
              else
                  System.out.println("Bad agent Name (must start with C_ and be in a package)
                  ->"+getClass().toString());
              Counter++;
        }
        public void changeFacets(Presentation Pres, Abstraction Abstr)
        {
              this.Pres     = Pres;
              if (Pres != null)
                  Pres.changeControl(this);
              this.Abstr    = Abstr;
              if (Abstr != null)
                  Abstr.changeControl(this);
        }
        public void newUserAction (String Message, Object Attachment)
        {
              System.out.println("ERROR : General User   action in "+getClass().toString() +" non-catched: "
              +Message);
        }
        public void newSystemAction (String Message, Object Attachment)
        {
              System.out.println("ERROR : General System action in "+getClass().toString() +" non-catched :"
              +Message);
        }
```

```
        public void newAction (String Message, Object Attachment)
        {
            System.out.println("ERROR : General Internal pac action in "+getClass().toString()
            +" non-catched : "+Message);
        }
        public Control getClone()
        {
            try {
                    Presentation P = null;
                    Abstraction A = null;
                    Control C = (Control)clone();
                    if (Pres != null)
                       P = (Presentation)(Pres.getClone());
                    if (Abstr != null)
                       A = (Abstraction)(Abstr.getClone());
                    C.changeFacets(P, A);
                    return C;
            } catch(Exception e) {System.out.println("ERROR: can't duplicate a control.");}
            return null;
         }
}
```

```
package PacModel;
public class Abstraction implements Cloneable
{
        protected Control Ctrl = null;
        public Abstraction (Control Ctrl)
        {
            this.Ctrl = Ctrl;
        }
        public void changeControl(Control Ctrl)
        {
        this.Ctrl = Ctrl;
        }
        public Abstraction getClone()
        {
            try { return (Abstraction)clone();
            } catch(Exception e) {System.out.println("ERROR: can't duplicate an abstraction.");}
            return null;
        }
}
```

```
package PacModel;
public class Presentation implements Cloneable
{
        protected Control Ctrl = null;
        public Presentation (Control Ctrl)
        {
            this.Ctrl = Ctrl;
        }
        public void changeControl(Control Ctrl)
        {
            this.Ctrl = Ctrl;
        }
        public void update()
        {
        }
        public Presentation getClone()
        {
            try { return (Presentation)clone();
            } catch(Exception e) {System.out.println("ERROR: can't duplicate a presentation.");}
            return null;
        }
}
```