Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

La négociation dans le commerce électronique

Bahali, Bugeni

Award date:
2000

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. May. 2026

La négociation dans le commerce électronique

Travail réalisé pour l'obtention du diplôme de maîtrise en Informatique

par
Liliane Bahali Bugeni

Année Académique 1999-2000

Remerciements

Je dédie ce travail à mon feu papa dont les encouragements m'ont permis d'arriver au bout de mes études aujourd'hui. Je le dédie aussi à ma mère pour sa patience et ses encouragements.

Je tiens à remercier Monsieur François Bodart pour avoir bien voulu diriger ce travail malgré ses multiples occupations. Mes remerciements à Monsieur Tung Bui, professeur à l'Université de Hawaii, qui a accepté de m'accueillir comme stagiaire pour la préparation de mon mémoire. Merci également à Madame et Monsieur Courtoy et Daipra pour leur soutient tant moral que matériel. Je remercie toutes les personnes qui m'ont aidé à réaliser ce travail, ils sont très nombreux et je ne pourrais pas les citer tous ici.

Abstract (version française)

Le commerce électronique se développe de plus en plus et divers outils sont crées pour assister les utilisateurs dans les transactions commerciales. La négociation joue un rôle important dans le processus d'une transaction commerciale. Nous nous sommes intéressés, dans ce mémoire, aux outils électroniques de support à la négociation. La discipline de l'intelligence artificielle est d'un apport important dans la réalisation d'agents intelligents susceptibles d'apprendre les stratégies d'une négociation. L'utilisation des tels agents rend possible la réalisation de certaines formes de négociation dans les transactions commerciales sur l'Internet. Nous proposons également dans ce mémoire, un prototype pour le support de la conversation dans une négociation réalisée sur l'Internet.

Abstract (english version)

Rapid advances of the Internet have opened new opportunities for business. Negotiation plays an important role in business processes. Many tools have been created for supporting commercial transactions over the Internet. This thesis focuses on electronic tools that could be used to support negotiation over the Internet. Artificial intelligence domaine is intending to create intelligent agents that could learn negotiation strategies. This thesis proposes a framework of a computer application for supporting negotiation conversation over the Internet.

TABLE DES MATIERES

TABLE DES MATIERES 4

INTRODUCTION 7

PREMIERE PARTIE 10

CHAPITRE 1 : NOTIONS SUR LA NÉGOCIATION 11

1.1. INTRODUCTION 11
1.2. IMPORTANCE DE LA NÉGOCIATION DANS UNE TRANSACTION COMMERCIALE 12
1.3. DIFFÉRENTES CATÉGORIES DE NÉGOCIATION DANS UNE TRANSACTION COMMERCIALE 13
*1.3.1. **Différents types de négociation en fonction du nombre de parties impliquées*** *13*
a) Négociation 1:1 14
b) Négociation 1:N, N:1 14
c) Négociation N:M 14
*1.3.2. **Différents types de négociation en fonction du nombre d'attributs*** *15*
a) Négociation mono-attribut 15
b) Négociation multi-attributs 15
*1.3.3. **Négociation itérative/non itérative*** *15*
*1.3.4. **Négociation faisant intervenir des intermédiaires*** *16*

CHAPITRE 2 : SUPPORT ÉLECTRONIQUE À LA NÉGOCIATION DANS LE CADRE DU COMMERCE ÉLECTRONIQUE 17

2.1. INTRODUCTION 17
2.2. QUELQUES AVANTAGES D'UN SUPPORT ÉLECTRONIQUE À LA NÉGOCIATION 17
2.3. TYPES DE TRANSACTIONS COMMERCIALES SUSCEPTIBLES D'ÊTRE SUPPORTÉES PAR UN OUTIL ÉLECTRONIQUE 18
*2.3.1. **Transactions faisant intervenir des intermédiaires*** *18*
*2.3.2. **Enchères non structurées*** *19*
*2.3.3. **Enchères structurées*** *20*
*2.3.4. **Le marchandage*** *20*
2.4. DIFFICULTÉS POUR AUTOMATISER LA NÉGOCIATION 21
2.5. DIFFÉRENTES APPROCHES POUR FOURNIR UN SUPPORT À LA NÉGOCIATION 22
*2.5.1. **Approche des systèmes d'aide à la décision et/ou à la négociation*** *22*
*2.5.2. **Approche considérant la négociation comme un processus dynamique : Evolutionary System Design*** *24*
2.6. APPORT DE L'INTELLIGENCE ARTIFICIELLE 27
a) Modèle de la théorie des jeux 28
b) Intelligence artificielle distribuée 28

CHAPITRE 3 : LES VENTES AUX ENCHÈRES 31

3.1. INTRODUCTION 31
*3.1.1. **Les ventes aux enchères dans la réalité*** *31*
*3.1.2. **Stratégies des ventes aux enchères*** *32*
3.2. DIFFÉRENTS TYPES D'ENCHÈRES 33
*3.2.1. **Les enchères anglaises*** *33*
a) Description 33
b) Règle de fonctionnement 33
c) Stratégie de l'acheteur 34
d) Avantages/ Inconvénients 34
*3.2.2. **Les enchères hollandaises*** *35*
a) Description 35
b) Règle de fonctionnement 35
c) Stratégie de l'acheteur 35
d) Avantages/ Inconvénients 35
*3.2.3. **Les enchères aveugles*** *36*

a) Description 36
b) Règle de fonctionnement 36
c) Stratégie de l'acheteur 36
d) Avantages/ Inconvénients 36
3.2.4. Les enchères Vickrey *37*
a) Description 37
b) Règle de fonctionnement 37
c) Stratégie de l'acheteur 37
d) Avantages/ inconvénients 37
3.2.5. Reverse Auction *38*
a) Description 38
b) Règle de fonctionnement 38
c) Stratégie du vendeur 38
d) Avantages/ Inconvénients 39
3.3. LES ENCHÈRES EN LIGNE OU VIRTUELLES 39
3.3.1. Mécanisme des enchères en ligne *39*
3.3.2. Caractéristiques importantes dans une vente aux enchères en ligne *41*
3.3.3. Fonctionnement sur le Web des enchères anglaises et hollandaises *42*
a) Les enchères anglaises 42
b) Les enchères hollandaises 44
3.3.4. Evaluation comparative des différents types des ventes aux enchères *46*

CHAPITRE 4 : L'E-CONTRACTING 48

4.1. INTRODUCTION 48
4.2. BESOINS DES UTILISATEURS 49
4.3. ASPECT JURIDIQUE DE L'E-CONTRACTING 50
4.3.1. Problèmes liés à la nature de l'Internet *51*
4.3.2. Solutions proposées par le législateur *52*

DEUXIEME PARTIE 54

CHAPITRE 5 : PROTOTYPE POUR UN SUPPORT À LA NÉGOCIATION 1:N SUR L'INTERNET 55

5.1. DESCRIPTION DU PROTOTYPE 55
5.1.1. Définition du contexte *56*
5.1.2. Structure des objets du dialogue *57*
a) Structure d'une offre 57
b) Structure d'une contre-offre 57
c) Structure d'une demande d'information 57
d) Structure de l'accord sur une offre 58
e) Structure d'une confirmation d'un accord 58
f) Structure de l'historique de la négociation 58
g) Structure de l'annulation d'une négociation 59
5.2. SPÉCIFICATIONS DES DIALOGUES 59
5.2.1. Le dialogue de l'offre *59*
5.2.2. Le dialogue de la contre-offre *61*
5.2.3. Le dialogue de demande d'information *62*
5.2.4. Le dialogue de l'acceptation de l'offre *63*
5.2.5. Le dialogue de la confirmation de l'offre *63*
5.2.6. Le dialogue de l'abandon de la négociation *64*
5.2.7. Le dialogue de la consultation de l'historique *65*
5.3. ARCHITECTURE LOGICIELLE DU PROTOTYPE 65
5.3.1. Analyse de la tâche *66*
a) Diagramme des buts et sous-buts 66
b) Décomposition en actions 68
c) Identification des objets de la tâche 71
d) Fonctions sémantiques 71
e) Graphe d'enchaînement des fonctions 72
f) Choix des attributs de dialogue 74
g) Identification des unités de présentation (UP) 75
h) Identification des fenêtres (FL) 75
i) Structuration des objets du dialogue 78
5.4. INTÉRÊT DU PROTOTYPE 84
5.5. RÉALISATION DU PROTOTYPE 84

5.6. EXTENSION POSSIBLE DU PROTOTYPE 84

ANNEXES 88

1. SPÉCIFICATIONS FONCTIONNELLES 89
2. SCHÉMA ENTITÉ-ASSOCIATION 95
3. CODE SQL DE LA BASE DE DONNÉES 96
4. CODES SOURCES 104
5. PRÉSENTATION DE L'INTERFACE DU PROTOTYPE 159

RÉFÉRENCES 166

Introduction

La croissance, l'intégration et la puissance des technologies de l'information et de la communication entraînent des nombreux changements dans la société actuelle. Aujourd'hui, l'Internet est considéré comme étant le réseau mondial. De nombreux réseaux s'y sont attachés permettant ainsi l'échange de l'information entre des utilisateurs quelle que soit la distance les séparant. Le but principal d'un réseau est de permettre aux applications de s'échanger de l'information. Sur l'Internet, de nombreuses applications ont été développées : le courrier électronique (e-mail), les nouvelles (news), la connexion à distance (telnet), transfert de fichier (ftp), la toile (World Wide Web), etc. Le World Wide Web est actuellement l'une des applications phares de l'Internet à tel point que beaucoup d'utilisateurs le considèrent comme étant l'Internet. Le Web sert de support pour la réalisation des activités telles que la distribution de documents, les programmes radio, les catalogues de bibliothèques et le commerce électronique.

Plusieurs raisons poussent les gens à faire des échanges commerciaux par voie électronique. Premièrement, la croissance rapide des utilisateurs de l'Internet constitue un marché potentiel de grande taille et attractif pour n'importe quels échanges commerciaux. Deuxièmement, l'Internet offre une nouvelle façon de faire des échanges commerciaux qui peut vaincre les barrières géographiques et temporelles. Troisièmement, les fournisseurs de service peuvent avoir plus d'information sur leurs clients et les clients peuvent recevoir un meilleur service. Par exemple, les vendeurs peuvent recueillir des informations sur les acheteurs à travers le réseau, et les acheteurs peuvent obtenir facilement des meilleurs produits[1] en accédant à plus de commerces ou en ayant plus d'alternatives disponibles. L'Internet offre une structure de marché dans laquelle les vendeurs tant professionnels qu'occasionnels peuvent réaliser des bonnes affaires. Le commerce prend donc une nouvelle dimension sur l'Internet.

Il n'y a pas une définition universelle du commerce électronique. Le terme commerce électronique peut désigner toute forme de transaction réalisée à distance à l'aide d'outils tels que le téléphone, le Minitel ou l'ordinateur. Certaines définitions vont jusqu'à inclure les échanges du secteur public vers le public même lorsqu'ils sont gratuits. Dans ce mémoire, nous nous limiterons aux transactions commerciales utilisant les ordinateurs à travers des réseaux informatiques.

---

[1] Dans ce travail, le terme produit désigne un bien ou un service

Le commerce électronique est très attractif, mais le processus des transactions commerciales sur l'Internet est souvent complexe. Les différentes parties impliquées ont des intérêts différents : d'une part, recueillir et analyser l'information et négocier les contrats, d'autre part, exécuter les transactions en toute sécurité et avoir des services de suivi sur l'Internet. Comme pour les transactions commerciales matérielles, on distingue quatre phases pour les transactions commerciales sur l'Internet : la phase d'information, la phase de négociation, la phase d'exécution et la phase de l'après-vente [2]. Il est, par conséquent, important de développer un environnement qui peut soutenir la croissance du marché électronique et contrôler le l'évolution de sa complexité.

Il existe une grande surcharge d'informations disponibles sur le Web. Par conséquent, les utilisateurs peuvent avoir du mal à retrouver l'information qu'ils recherchent. Pour accélérer la vitesse et améliorer la performance des transactions commerciales, certaines tâches telles que la recherche et la comparaison des informations sont confiées à des agents logiciels. Un agent logiciel est un programme qui peut agir de façon autonome et accomplir une tâche sans une intervention humaine directe.

L'étude présentée dans ce document porte essentiellement sur la négociation dans les transactions commerciales réalisées sur l'Internet. Nous avons subdivisé ce mémoire en deux parties. Dans la première partie nous présentons un état des lieux des recherches actuelles dans le domaine de la négociation et plus particulièrement les recherches portant sur l'utilisation des agents intelligents dans les outils d'aide à la négociation. Dans la deuxième partie nous proposons le prototype d'un outil susceptible de guider les parties qui veulent négocier un achat sur l'Internet.

Le chapitre 1 définit la notion de négociation, montre son importance dans le processus des échanges commerciaux et donne ses caractéristiques principales. Dans le chapitre 2, nous introduisons la notion de négociation automatisée. Nous justifions ensuite pourquoi il est important d'offrir un support électronique à la négociation sur le Web et nous donnons des raisons qui rendent le processus de négociation difficile à automatiser. Nous parlons également de l'apport de l'intelligence artificielle dans les recherches qui essaient d'utiliser des agents intelligents pour créer des systèmes de négociation entièrement automatisés, susceptibles de négocier à la place des humains. Nous examinons aussi l'approche des systèmes d'aide à la décision/négociation qui tentent de créer des outils de support électronique à la négociation. Dans le chapitre 3, nous analysons le cas des ventes aux enchères, une forme de négociation basée uniquement sur le prix et supportée par des agents logiciels. Nous parlons aussi des différents types de vente aux

enchères et leur fonctionnement sur l'Internet. Certaines formes de ventes aux enchères sont entièrement automatisées sur l'Internet.

Lorsqu'une une négociation se termine par un consensus, elle doit être suivie de la signature du contrat dont les termes auront été discutés lors de la négociation. Il nous semble donc intéressant d'analyser les problèmes qui peuvent se poser lors de la conclusion des contrats sur l'Internet. Le chapitre 4 traite de la réalisation des contrats sur l'Internet (e-contracting) et ses conséquences juridiques. Dans le chapitre 5, nous proposons un système capable de guider un vendeur et un acheteur lors d'une négociation sur l'Internet. Ce système propose une forme structurée d'offre (ou de contre-offre) susceptible de permettre aux parties qui négocient de formaliser leurs propositions de manière claire. Il offre également aux parties la possibilité de s'exprimer de manière non structurée lorsqu'elles le désirent, grâce à des messages sous forme de texte libre.

PREMIERE PARTIE

La première partie se subdivise en quatre chapitres. Le premier chapitre introduit la notion de négociation. Le second chapitre 2, quant à lui, traite de la notion de support électronique à la négociation et présente l'état des recherches. Dans le troisième chapitre, nous analysons un cas spécifique de négociation, celui de la vente aux enchères dont le support électronique est, dans certains cas, entièrement automatisé. Nous abordons, enfin dans le chapitre 4, la dimension légale à laquelle on ne peut échapper dès lors qu'il y a négociation commerciale et conclusion d'un contrat de vente.

Chapitre 1 : Notions sur la négociation

1.1. Introduction

"La négociation est l'action de négocier, de discuter les affaires communes entre des parties en vue d'un accord" (dictionnaire Le Petit Larousse). Lax et Sebenius, 1986, définissent la négociation comme étant un processus d'interaction potentiellement opportuniste par lequel deux ou plusieurs parties, ayant un certain conflit apparent, cherchent à améliorer la situation à travers des actions décidées conjointement [21].

La négociation implique à la fois le conflit et la coopération. Au cours d'une négociation, chacune des parties cherche à pousser l'autre partie à accepter ses propositions. La négociation est un processus dynamique dans lequel chaque partie possède un objectif à atteindre. Très souvent, leurs objectifs sont opposés et par conséquent, chaque partie doit faire des concessions pour essayer de parvenir à un consensus. Le processus évolutif des discussions n'est pas linéaire. A certains moments de la négociation, la discussion peut avancer vers un consensus et à d'autres s'en éloigner. Il est difficile de prévoir à l'avance quelle direction vont prendre les discussions. Cela dépend de non seulement de l'habilité, de la culture et des informations dont disposent les négociateurs, mais aussi de la volonté des parties à atteindre le but qu'elles se sont fixées.

Le processus de négociation nécessite un espace d'états et des fonctions d'utilité. Par état, il faut entendre la combinaison d'un certain nombre d'attributs. Les attributs d'une négociation sont les éléments sur lesquels portent les discussions. L'adjonction d'un attribut ou sa suppression va modifier un état. Un acheteur et un vendeur qui n'arrivent pas à se mettre d'accord sur une combinaison {prix, date de livraison} (état 1) peuvent changer d'état en ajoutant les termes du crédit comme attribut supplémentaire à la négociation (état 2 = combinaison {prix, date de livraison, termes du crédit}). Les fonctions d'utilités déterminent le niveau de satisfaction d'une partie pour un attribut donné. Un acheteur pourrait avoir une utilité élevée pour un prix bas, tandis que le vendeur pourrait avoir une fonction d'utilité totalement opposée à celle de l'acheteur, lui même étant disposé à vendre à un prix élevé.

Si l'on se réfère aux négociations matérielles ou traditionnelles entre deux parties, il arrive souvent que la négociation commence alors que les parties n'ont pas une idée claire sur l'espace

d'états et/ou ne connaissent pas leurs fonctions d'utilité. L'espace d'états s'élargit lorsque des nouvelles considérations entrent en jeu et les fonctions d'utilité sont redéfinies lorsqu'une personne a une réaction face à une affaire proposée.

Dans une négociation, l'acheteur et le vendeur essaient de déterminer une solution optimale à travers un espace de solutions possibles [3, 6], la notion d'optimalité étant relative à la subjectivité du décideur. La négociation peut donc être considérée comme la recherche d'une solution optimale. Mais la négociation diffère d'une recherche d'une solution optimale normale (recherche opérationnelle) par le fait que, dans une négociation, chaque partie possède des informations secrètes que l'autre partie ne connaît pas. En plus, les parties ne connaissent pas les fonctions d'utilité des autres. Toutes les deux parties ont des motivations qui les poussent à cacher leurs préférences. Trouver une solution optimale dans un tel environnement est un défi difficile à réaliser. Toutes les parties sont en compétition, mais doivent chercher conjointement l'espace de solutions possibles.

La négociation humaine se base souvent sur le secret et d'autres stratégies qui ne sont pas fondées sur la vérité (stratégies mensongères), dans le seul but de tirer le plus de profit de la situation.

1.2. Importance de la négociation dans une transaction commerciale

La négociation joue un rôle non négligeable dans une transaction commerciale. Nous définissons cette dernière avant d'examiner le rôle de la négociation.

Une transaction commerciale peut être décomposée en quatre phases :

- La phase d'information : l'acheteur est renseigné sur le produit qui lui est proposé;
- La phase de négociation : l'acheteur et le vendeur discutent les conditions de vente du produit;
- La phase d'exécution : consiste en la signature des accords convenus à l'issue de la négociation et en la livraison du produit pour lequel l'accord a été conclu;
- La phase de l'après-vente : l'acheteur prend possession ou jouit du produit convenu.

C'est lors de la phase de négociation que les parties discutent réellement sur la valeur du produit. La négociation permet au vendeur de montrer à l'acheteur la valeur du produit qu'il offre, et à l'acheteur de mieux évaluer la valeur du produit avant de prendre une décision. Chacune des parties utilise des stratégies pour pousser l'autre à accepter ses conditions. A la différence des transactions dans lesquelles le produit est à prendre ou à laisser, une transaction dans laquelle les parties peuvent négocier offre plus de possibilités aux parties de conclure une affaire. La valeur du produit change au fur et à mesure que la négociation avance, les différentes parties faisant intervenir des éléments qui augmentent ou diminuent la valeur du produit. Il est donc indispensable de pouvoir bien réaliser ce processus au cours d'une transaction commerciale.

1.3. Différentes catégories de négociation dans une transaction commerciale

On peut distinguer les types négociations en fonction du nombre de parties impliquées, du nombre d'attributs (par exemple le prix, les conditions de retour) qui interviennent dans la négociation, de la nature itérative de la négociation et selon que la négociation fait intervenir des intermédiaires ou pas (C. Beam [4]).

1.3.1. Différents types de négociation en fonction du nombre de parties impliquées

Dans une négociation, nous pouvons avoir deux ou plusieurs parties. Nous avons trois types possibles de configurations vendeur/acheteur : la négociation 1:1, la négociation 1:N, N:1 et la négociation N:M. La notation i:j se lit i vendeurs et j acheteurs.

a) Négociation 1:1

On parle de négociation 1:1 lorsqu'un vendeur négocie avec un seul acheteur.

b) Négociation 1:N, N:1

On parle de négociation 1:N lorsqu'un vendeur est en présence de plusieurs acheteurs et négocie en parallèle avec chacun d'eux. Si le produit qu'il offre est unique et indivisible, il cesse la négociation avec les autres acheteurs potentiels dès qu'il atteint un consensus avec l'un d'entre-eux. S'il dispose, par contre, de plusieurs occurrences du produit proposé, il peut continuer à négocier avec les autres acheteurs après avoir conclu la négociation avec l'un d'entre-eux.

Généralement, une négociation entre le vendeur et un client est secrète. Le vendeur optera pour ce type de négociation si, par exemple, il souhaite vendre ses produits à des prix différents et adapter son offre en fonction du client ou de l'intérêt manifeste de ce dernier. Par contre, il choisira une négociation "publique", accessible à tous les autres clients, dans le but de susciter la concurrence entre les clients et ainsi maximiser son profit.

On parlera de négociation N:1 lorsqu'un acheteur met en compétition plusieurs vendeurs. La situation de l'acheteur est la même que celle du vendeur dans le cas 1:N : la négociation peut être également publique ou secrète.

c) Négociation N:M

Dans la configuration N:M, plusieurs vendeurs négocient avec plusieurs acheteurs. Les vendeurs comme les acheteurs soumettent leurs offres, requêtes et les termes de la négociation à un intermédiaire. Ce dernier compare les différentes propositions et essaie de faire coïncider les offres des vendeurs aux requêtes des acheteurs. Puis, informe les parties concernées s'il arrive à établir une correspondance entre les propositions d'un vendeur et d'un acheteur.

1.3.2. Différents types de négociation en fonction du nombre d'attributs

La négociation peut porter sur le prix, mais également sur le délai de livraison, les conditions de retour, etc. Nous pouvons, par conséquent, distinguer les négociations en fonction du nombre d'attributs qui sont pris en compte dans les discussions.

a) Négociation mono-attribut

On parle de négociation mono-attribut lorsqu'elle ne porte, par exemple, que sur le prix : les parties ne marchandent que sur le prix du produit sans faire intervenir d'autres éléments pouvant modifier la valeur du produit.

b) Négociation multi-attributs

On parle de négociation multi-attributs lorsqu'elle fait intervenir plusieurs attributs tels que le prix, le délai de livraison, le mode de payement, le service après-vente et les conditions de retour des marchandises. Lors de l'achat d'un véhicule, par exemple, la négociation entre le vendeur et l'acheteur est une négociation multi-attributs car elle peut faire intervenir le nombre de places assises dans le véhicule, le prix du véhicule, la puissance du moteur, le délai de livraison, le mode de payement, la marque du véhicule, les options, etc.

1.3.3. Négociation itérative/non itérative

Dans la négociation itérative, les parties peuvent améliorer l'information concernant la partie adverse pour permettre de mieux conduire la négociation et augmenter les chances d'atteindre un objectif satisfaisant. Chacune des parties donne une valeur aux différents attributs intervenant dans la négociation. Pour faire évoluer la négociation, les parties modifient le poids affecté aux différents attributs. L'opération est répétée plusieurs fois jusqu'à ce qu'un consensus soit atteint, ou que la négociation se conclue sur un échec.

1.3.4. Négociation faisant intervenir des intermédiaires

La négociation peut faire ou non intervenir un intermédiaire. Les négociations qui ne font pas intervenir des intermédiaires nécessitent que les deux parties déterminent leurs besoins, se rencontrent, négocient et arrivent éventuellement à un accord directement. Par contre, certaines négociations ont recours à la médiation d'une tierce partie pour assister au moins une des parties. Cette tierce partie peut être une personne mais peut être également un support électronique. Le chapitre suivant traite des supports électroniques à la négociation.

Chapitre 2 : Support électronique à la négociation dans le cadre du commerce électronique

2.1. Introduction

Comme nous l'avons vu dans le chapitre précédent, la négociation peut être considérée comme une recherche de solution optimale. La négociation électronique va tenter d'automatiser le processus de négociation.

La négociation automatisée dans le commerce électronique est définie comme étant le processus par lequel deux ou plusieurs parties ayant recours aux outils et techniques du commerce électronique marchandent des produits de façon multilatérale dans l'intention d'obtenir un gain mutuel [2]. Le processus de négociation automatisée doit avoir la capacité de faire des propositions et d'éventuelles contre-propositions si nécessaire. Précisons, par ailleurs, qu'une négociation n'est dite automatisée que si elle est réalisée entièrement par des ordinateurs : ainsi, une négociation supportée par le courrier électronique, et où il existe donc une interaction humaine, ne peut être considérée comme automatisée.

La négociation électronique implique la création d'agents [7, 15, 16, 17, 18, 23, 3] programmés sur base des stratégies de la négociation. Lorsqu'une négociation doit se dérouler, les différentes informations relatives à cette négociation leur sont communiquées. Des tels agents peuvent négocier soit avec des êtres humains, soit avec d'autres agents logiciels, et proposer une solution qui pourrait nécessiter l'accord d'un humain.

2.2. Quelques avantages d'un support électronique à la négociation

La négociation étant considérée comme une recherche de solution optimale, nous avons vu précédemment qu'il est difficile de trouver une solution optimale dans un environnement où toutes les parties sont en compétition, mais doivent chercher conjointement l'espace de solutions possibles. Etant donné cette difficulté de la recherche à définir l'espace des solutions possibles et

l'échec relativement fréquent que rencontrent les êtres humains pour réaliser cette tâche, la mise au point d'agents automatisés qui pourraient rechercher cet espace de solutions possibles serait un apport énorme pour les négociateurs.

La fourniture d'un support électronique à la négociation permettrait d'accélérer le processus des transactions commerciales et surtout permettre aux parties concernées de se consacrer à d'autres activités sur l'Internet. Les parties concernées pourront faire autre chose pendant que "le négociateur automatique" négocie à leur place. Comme les parties qui négocient ont souvent des objectifs opposés, un négociateur automatique devrait aider à trouver une solution satisfaisante pour toutes les parties.

Le processus de négociation automatisée, bien que très intéressant, est encore loin d'être au point. C'est la raison pour laquelle dans le cadre de ce mémoire, nous ne nous intéressons qu'à des supports actifs d'aide à la négociation qui pourraient éventuellement conduire à une négociation entièrement automatisée.

2.3. *Types de transactions commerciales susceptibles d'être supportées par un outil électronique*

Caroline Beam [4] a identifié quatre types de transactions qui pourraient théoriquement être prises en charge par un outil électronique : les transactions faisant intervenir des intermédiaires, les enchères non structurées, les enchères structurées et le marchandage.

2.3.1. Transactions faisant intervenir des intermédiaires

Comme nous l'avons vu au chapitre 1, une transaction commerciale peut être divisée en quatre phases : la phase d'information, la phase de négociation, la phase d'exécution et la phase de l'après-vente. Chacune des phases peut être réalisée par un ou plusieurs fonctions d'intermédiation (Tableau 2.1). Les fonctions d'intermédiation aident ou permettent à un acheteur et/ou à un vendeur de réaliser une transaction. Par exemple, le vendeur peut faire intervenir un intermédiaire de transport (par exemple FedEx, DHL) ou encore un intermédiaire pour le paiement (par ex. Visa, Mastercard). L'acheteur et/ou le vendeur peut faire intervenir un tiers de confiance (trusted third party) pour assurer que les parties respectent leurs engagements.

Phases d'une transaction commerciale	Fonctions d'intermédiation
Information	- fonction de recherche, - fonction d'agrégation (pour agréger les demandes des vendeurs et acheteurs et les fournir à un même endroit) - fonction de recommandation de produits - fonction d'information (pour vendre des informations sur un produit, autres que le prix)
Négociation	- fonction de négociation
Exécution	- fonction de transport - fonction de payement - fonction de sécurité - fonction d'escrow (société fiduciaire) - fonction d'authentification
Après-vente	

Tableau 2.1.

Les transactions qui nous intéressent ici sont celles pour lesquelles au moins une des fonctions est réalisée par un intermédiaire électronique. Un intermédiaire électronique est une entité qui effectue au moins une fonction d'intermédiation de façon automatique (un agent logiciel).

Actuellement, presque toutes ces fonctions d'intermédiation sont réalisables sur l'Internet par des agents logiciels. Mais, la fonction de négociation n'est réalisable sur l'Internet actuellement que pour les transactions commerciales faisant intervenir un intermédiaire qui négocie uniquement sur le prix. Ce genre de transaction est appelé "brokered transaction" et les intermédiaires, quant à eux, sont appelés intermédiaires en ligne ou "online brokers".

Il n'y a pas de stratégie générale pour l'intermédiation en ligne, chaque intermédiaire utilisant la stratégie optimale pour la niche dans laquelle se situe le produit pour lequel il négocie.

2.3.2. Enchères non structurées

Les enchères non structurées sont des transactions qui s'opèrent entre un acheteur et plusieurs vendeurs, dans lesquelles ces derniers mettent leurs offres en commun et les proposent

directement à l'acheteur [4]. Les enchères non structurées sont une négociation N:1, multi-attributs et peuvent se dérouler de manière itérative.

Ces enchères sont dites non structurées car il n'y a pas de règles imposées aux vendeurs pour soumettre leurs offres. Chaque groupe de vendeurs a ses propres critères pour regrouper les offres. Les groupes se forment en fonction des objectifs des vendeurs.

Ce type de transaction ne convient pas à la négociation électronique parce que les offres n'étant pas structurées, il est difficile de déterminer les règles et stratégies nécessaires pour l'implémentation d'agents susceptibles de prendre en charge la négociation.

2.3.3. Enchères structurées

Les enchères structurées sont celles qui sont utilisées pour les ventes aux enchères [2, 3, 4, 9, 10, 11]. C'est une négociation 1:N qui porte sur le prix uniquement. Elles peuvent donc être prises en charge par un intermédiaire électronique. Les enchères structurées sont des transactions qui conviennent le mieux à la négociation électronique car les règles peuvent être parfaitement définies. Il est, par conséquent, facile de créer des agents logiciels capables de réaliser une vente aux enchères en ligne. Nous analysons dans le chapitre suivant les types de vente aux enchères et leur fonctionnement sur l'Internet.

2.3.4. Le marchandage

Le marchandage est une négociation de type 1:1 qui peut être multi-attributs et est souvent réalisée de manière itérative [3, 4].

Il n'existe pas encore d'agents logiciels pour réaliser une négociation multi-attributs dans le commerce électronique. Le manque de règles et de stratégies de marchandage fait défaut pour envisager de créer des agents pour supporter le marchandage dans le commerce électronique. Actuellement, les recherches destinées à déterminer les stratégies à appliquer dans la négociation électronique multi-attributs sont encore au stade expérimental. Ces recherches peuvent être regroupées en deux approches :

- ***Les recherches pour la mise au point d'agents basés sur des stratégies pré-programmées [23, 25, 26] :*** il s'agit de créer des agents qui doivent contenir toutes les stratégies nécessaires pour conduire une négociation. Ces agents sont entièrement autonomes et libres d'acquérir de l'information supplémentaire venant de l'extérieur au fur et à mesure que la négociation avance. Toutefois, les stratégies à utiliser dans chaque cas sont prédéterminées.

- ***Les recherches pour la mise au point d'agents susceptibles apprendre des stratégies [24, 27, 28] :*** dans ce cas-ci, on voudrait avoir des agents qui ont la capacité d'apprendre pendant le marchandage (exemple, la programmation génétique).

2.4. Difficultés pour automatiser la négociation

L'automatisation de la négociation nécessite la définition de l'ontologie, c'est-à-dire d'un vocabulaire capable de décrire les objets d'une manière claire et non ambiguë à un agent logiciel. Trouver un vocabulaire non ambigu pour spécifier une automobile, ou un produit alimentaire, ou un horaire pour la livraison peut s'avérer très compliqué. Le problème de l'ontologie est crucial pour le processus de négociation car on doit s'assurer que les agents se réfèrent au même bien.

Le deuxième problème qui rend difficile l'automatisation de la négociation est lié à la stratégie de la négociation. Si la stratégie de négociation d'un agent est connue par l'autre agent, le premier agent pourrait être dans une situation désavantageuse. Supposons, par exemple, que l'acheteur sait que la stratégie du vendeur est d'accepter toute offre au-dessus d'un certain seuil de valeur. L'acheteur peut commencer par offrir 0 franc, puis offrir au vendeur 1 franc supplémentaire à chaque fois, jusqu'à ce que le seuil de valeur du vendeur soit atteint. A ce point, l'affaire est conclue mais c'est une solution extrême pour le vendeur qui cède le bien au plus bas prix possible.

Il est difficile de trouver une bonne stratégie dans une négociation entre individus. Concevoir un programme implémentant une bonne stratégie est encore plus difficile.

La plupart des stratégies dans les négociations entre individus sont basées sur le secret et la confidentialité. Chaque partie préfère garder secrète l'information concernant la stratégie qu'elle va appliquer pour éviter que la partie adverse ne l'utilise en sa faveur. La connaissance de la

stratégie d'une partie par la partie adverse réduit le profit que le propriétaire de la stratégie aurait pu tirer de la négociation si sa stratégie était restée secrète. Ce type de stratégie n'est pas bonne à utiliser pour les agents logiciels supportant la négociation car, pour qu'elle soit efficace, elle doit se baser sur la confidentialité. En effet, chaque avancée dans une session de négociation dévoile une partie de la stratégie. Avec le temps, on peut finir par découvrir la plupart des stratégies utilisées par les agents logiciels. Plus la stratégie est complexe, plus il sera difficile de la comprendre, mais également, plus il sera difficile de la programmer dans un agent logiciel.

Plutôt que d'essayer de concevoir des stratégies de négociation ingénieuses ou très bien protégées pour ne pas être comprise par une partie adverse, il serait mieux d'utiliser des stratégies qui peuvent être connues par la partie adverse sans que la partie dont la stratégie est dévoilée soit dans une situation de désavantage. Cette stratégie est intéressante et peut facilement être programmée dans un agent logiciel.

2.5. *Différentes approches pour fournir un support à la négociation*

L'objectif d'un outil d'aide à la négociation est d'assister les parties qui négocient à prendre une décision sur le prix, la date de livraison, etc. Un tel outil devrait fournir une base de données commune, contenant des informations sur les négociations en cours et précédentes. Il devrait également permettre d'établir les liens de communication entre les parties, et aider à identifier les zones d'accord et de désaccord, vraisemblablement à travers une discussion mutuelle et des votes périodiques sur les questions.

Il est plus facile de fournir un service d'aide à la négociation lorsque les parties sont géographiquement rapprochées, les liens de communication étant plus faciles à établir dans ce cas là. Les discussions et les votes à distance sont plus difficiles dans un environnement distribué que dans un environnement concurrent (les parties sont localisées à un même endroit) car lorsque les parties sont dispersées, la base de données doit contenir des composants centralisés et décentralisés. Comme l'Internet est accessible partout au monde, il pourrait faciliter les négociations commerciales distribuées. Les agents logiciels pourraient être utilisés pour fournir une architecture logicielle distribuée offrant un support à la négociation.

2.5.1. Approche des systèmes d'aide à la décision et/ou à la négociation

Les systèmes d'aide à la décision ou DSS (Decision Support Systems) sont des logiciels conçus pour aider les décideurs à opérer de meilleurs choix [2]. Un système d'aide à la négociation ou NSS (Negotiation Support System) est un logiciel spécialement adapté pour aider les négociateurs à prendre des meilleures décisions et à négocier de façon plus productive [6]. Un NSS est considéré généralement comme une extension d'un GDSS (Group Decision Support System) composé des DSS individuels interconnectés via un système de communication [8].

Souvent les négociateurs connaissent assez bien les problèmes de la négociation, mais ne sont pas capables de formuler leurs préférences avec précision ni évaluer correctement les impacts des différents accords possibles. L'objectif d'un NSS est d'aider l'utilisateur à évaluer les revendications de son partenaire, à se faire une idée du bénéfice additionnel d'une situation donnée, à maximiser les opportunités et à négocier avec doigté les termes du contrat.

R. Blaming et T. Bui [13] ont montré comment la technologie des systèmes d'aide à la décision peut être incorporée dans le commerce électronique. Pour ces auteurs, l'Internet pourrait être considéré comme un outil d'aide à la négociation. La négociation a lieu dans le commerce électronique lorsque le prix, la date de livraison, etc., ne sont pas déterminés dans le marché mais sont donnés séparément pour chaque transaction.

Les NSS peuvent être classés en deux catégories [14] : les "solution-driven" NSS et les "process support" NSS. Un NSS "solution-driven" fournit des solutions alternatives ou des suggestions d'accords possibles aux parties qui négocient, à l'inverse, un NSS "process support" est conçu pour supporter le processus de négociation, de la préparation à la signature du contrat, et non pas pour fournir de solutions. En cela, il se distingue du NSS "solution-driven" par le fait qu'il offre des canaux de communication enrichis et un travail coopératif.

Les NSS sont des puissants outils capables d'améliorer les négociations. Mais ils sont encore loin de pouvoir supporter une négociation totalement automatisée : ils nécessitent encore une intervention humaine constante. Le lancement du problème initial ainsi que la prise de décision finale sont laissés au négociateur humain.

Deux exemples des systèmes de négociation basés sur les NSS :

1) L'Université de Carleton au Canada (www.business.carleton.ca/inspire/) a mis au point un système interactif d'aide à la négociation sur le Web, appelé INSS (Internet

Negotiation Support System) fonctionnant comme un NSS solution-driven. Ce système aide deux utilisateurs à négocier sur un problème prédéfini. Il contient un mécanisme pour l'évaluation des préférences, un système de messagerie interne et des outils graphiques qui permettent de visualiser la progression de la négociation. Il offre au négociateur une méthode pour construire sa fonction d'utilité lui permettant d'évaluer les propositions. Cependant, ce système ne fournit pas un support complet du processus de négociation aux utilisateurs pour organiser et négocier des questions complexes. L'INSS n'est pas un système entièrement automatisé.

2) Un autre système de négociation fonctionnant comme un NSS "process support", appelé CBSS (Collective Bargaining Support System), a été développé par Yuan, Rose et Hasanuddin [14]. Ce système permet une communication et une interaction en temps réel. Il contient un système de messagerie et un menu principal composé de trois parties : la pré-session, la session et l'aide. La pré-session supporte les préparations de la négociation, la session supporte un processus de négociation structurée et l'aide fournit une aide en ligne. Le système peut être utilisé sur le Web et doit pouvoir supporter un processus complet de négociation.

2.5.2. Approche considérant la négociation comme un processus dynamique : Evolutionary System Design

La négociation est un processus de résolution de conflits qui évolue avec le temps (un processus dynamique). Certains chercheurs (Mwana, 1996 et Shakun, 1988) [15, 21, 29] utilisent les NSS pour essayer concevoir des systèmes qui peuvent prendre en compte l'évolution de la représentation d'un problème dans un groupe. Ce système, appelé "Evolutionary System Design" (ESD), est une méthodologie de résolution de conflits visant à amener les négociateurs à modifier leurs objectifs et jugements individuels de manière à augmenter les chances d'atteindre une solution commune. Cette méthodologie aide à définir et à trouver une solution pour les problèmes qui changent dans le temps.

Le tableau 2.2. donne une description d'un "Evolutionary System Design". Ce processus est subdivisé en quatre étapes au cours desquelles la négociation évolue vers une solution optimale.

- *Première étape : simulation de l'offre*

Etant donné que le vendeur et l'acheteur d'un article sont souvent opposés par le prix, le vendeur voulant vendre au prix le plus élevé possible et l'acheteur voulant acheter le produit au prix le moins élevé possible. La première étape du ESD simule le processus par lequel le vendeur pousse l'acheteur à considérer des caractéristiques de l'article qu'il vend mais que l'acheteur n'a pas considérées pour estimer la valeur de l'article en question. L'acheteur soulève également des éléments qui le pousse à ne pas offrir plus d'argent sur le produit proposé.

- *Deuxième étape : simulation de la concession*

Les parties voulant faire avancer la négociation font des concessions sur certaines valeurs (importance) qu'elles attribuent aux différentes caractéristiques de l'article concerné. Par exemple si un client désire acquérir un véhicule dont les caractéristiques principales sont la puissance du moteur et le faible coût d'achat, le vendeur essayer de l'amener à faire changer d'avis sur le caractère "prix" en lui montrant les avantages d'un véhicule de prix élevé, très puissante et spacieuse (introduction d'un attribut supplémentaire).

- *Troisième étape : simulation du changement d'objectifs*

Les parties changent leurs objectifs si elles se rendent compte de l'impossibilité d'atteindre leurs premiers objectifs. Par exemple, l'acheteur pourrait choisir de louer une voiture plutôt que d'en acheter une.

- *Quatrième étape : simulation du changement de partenaire*

Les parties changent de partenaire si elles se rendent compte qu'ils ne pourront pas atteindre une solution commune avec le partenaire courant. Par exemple, changer de concessionnaire de voiture.

Le problème qui se pose dans ce système est de connaître le moment où il faut passer d'une étape à l'autre et celui du caractère linéaire du système : peut-on passer de la première étape directement à la troisième ou quatrième? Ou encore de la deuxième à la quatrième?

Etapes	Espaces	Description de l'évolution	Résolution de conflit
Première étape	Solutions possibles (prédéfinies ou définies pendant la négociation)	Les parties modifient constamment leurs solutions courantes en progressant dans leurs espaces de solutions possibles respectifs dans le but de rechercher une solution commune.	Trouver la meilleure solution à un problème donné qui satisfait les parties.
Deuxième étape	Changement des valeurs (jugements/ préférences)	Si les parties n'arrivent pas à un consensus, après avoir parcouru leurs espaces de solutions possibles, elles peuvent changer les valeurs attachées à leurs objectifs respectifs. Le changement des valeurs donne un nouvel ensemble de solutions et les parties peuvent alors recommencer le processus pour rechercher un consensus.	Changer la représentation du problème en changeant les questions qui n'ont pas été résolues.
Troisième étape	Le changement des objectifs	Si les parties n'arrivent pas à un consensus après avoir changé plusieurs fois les valeurs des attributs de leurs objectifs, elles peuvent changer leurs objectifs respectifs et, par conséquent, elles pourront avoir des nouvelles solutions qui pourront leur permettre d'atteindre un consensus.	Changer la perception du problème.
Quatrième étape	Changement des partenaires	Si les parties ne peuvent toujours pas atteindre un consensus après avoir changé les objectifs, alors elles doivent changer de partenaire pour une nouvelle négociation.	Trouver un nouveau partenaire pourrait accroître la chance de trouver une solution commune.

Tableau 2.2. : Evolutionary System Design et résolution de conflit [21]

2.6. *Apport de l'intelligence artificielle*

La discipline de l'intelligence artificielle s'intéresse à la conception d'agents susceptibles de négocier entre eux [38-41].

Un agent doit pouvoir respecter les caractéristiques suivantes :

- ***autonomie*** *:* c'est-à-dire la capacité d'agir par lui-même sans une intervention humaine. Dans ce cas, l'agent doit être capable de prendre lui-même des initiatives plutôt que d'agir en réponse à l'environnement;

- ***coopération avec d'autres agents*** *:* c'est-à-dire la capacité d'interagir avec d'autres agents ou avec des humains à travers un langage de communication;

- ***capacité d'apprentissage*** *:* c'est-à-dire la capacité d'apprendre pendant qu'il agit et interagit avec son environnement externe. Un agent est intelligent lorsqu'il est en mesure d'apprendre.

On distingue deux sortes d'agents intelligents en fonction de la manière dont ils doivent acquérir et exécuter leurs instructions. On a d'une part, des agents intelligents qui doivent être créés avec toutes leurs stratégies, c'est-à-dire des agents possédant une large mémoire contenant des instructions détaillées pour chaque situation possible. D'autre part, on a des agents qui sont capables d'apprendre. Plutôt que d'avoir une large mémoire, ils peuvent acquérir l'expérience à partir des négociations précédentes qu'ils ont menées.

Les agents ont été utilisés pour diverses applications du commerce électronique : dans le but de réduire les coûts des recherches d'informations et ceux des transactions, pour réaliser les transactions de négociations telles que les ventes aux enchères. Les agents qui négocient doivent avoir des instructions concernant les règles et les stratégies de négociation efficaces.

Plusieurs recherches dans le domaine de l'intelligence artificielle distribuée et dans la théorie des jeux traitent des problèmes de coordination et de négociation en donnant des solutions pré-calculées à certains problèmes spécifiques.

a) Modèle de la théorie des jeux

Les modèles traditionnels de prise de décision utilisant un seul agent intelligent supposent que le décideur a une connaissance complète de l'ordre de ses préférences ou de sa fonction d'utilité ainsi que de la probabilité des différents résultats. Lorsque plusieurs agents sont impliqués, comme c'est le cas dans une négociation, l'introduction de l'interaction des stratégies des agents complique cette conception.

Pour éviter le problème d'interactions entre les stratégies des agents, les modèles de la théorie des jeux [17, 32] utilisent les hypothèses restrictives suivantes : le nombre de joueurs et leur identité sont supposés fixes et connus de tous, tous les joueurs sont supposés être totalement rationnels et chaque joueur sait que les autres sont rationnels. L'ensemble d'alternatives de chaque joueur est fixe et connu. La capacité de prise de risque de chaque joueur et les calculs de l'utilité sont également fixes et connus par tout le monde.

Ces hypothèses limitent l'applicabilité du modèle de la théorie des jeux dans le cadre de résolution de problèmes réels. Le caractère statique du modèle de la théorie des jeux constitue une limite à son application dans la résolution des problèmes réels. En effet, ils se concentrent principalement sur les résultats de la négociation par opposition au processus de négociation lui-même.

b) Intelligence artificielle distribuée

Les recherches en intelligence artificielle [17, 33] insistent sur le compromis tout en essayant de minimiser les interactions ou communications potentielles des agents impliqués. Sycara et Zeng [15] se sont intéressés à un autre ensemble de problèmes de recherche pour lequel ils considèrent que les agents peuvent avoir des mécanismes d'apprentissage dans le processus de négociation. Pour cela, ils ont adopté le modèle séquentiel de prise de décision (Sequential Decision Making) [17, 34, 35] pour construire un système de négociation.

Le modèle de décision séquentiel présente deux caractéristiques principales :

- une séquence de points (étapes) de prise de décision qui sont dépendants les uns des autres;

- le décideur a une chance de modifier sa connaissance après avoir exécuté une décision prise à une étape donnée : il aura ainsi plus d'informations pour prendre la décision à l'étape suivante.

Sycara et Zeng ont adopté ce modèle comme base pour le modèle de négociation car il se porte bien à la succession d'échanges de propositions et de contre-propositions.

Un modèle séquentiel de prise de décision fournit des composants prêts à être utilisés pour modéliser la nature itérative des interactions entre agents. De plus, les agents qui négocient reçoivent un feed-back, après avoir fait une proposition ou contre-proposition, sous forme de réponse en provenance d'agent(s) destinataire(s). Un modèle séquentiel de prise de décision supporte une approche ouverte du monde, c'est-à-dire qu'un agent ne doit pas avoir un modèle complet du monde au début de la négociation. Lorsqu'une nouvelle information arrive, l'agent peut utiliser la nouvelle connaissance à la prochaine étape de prise de décision. Enfin, l'apprentissage pourrait se faire facilement dans le modèle de prise de décision séquentielle. Ce type de comportement d'apprentissage incrémentale en ligne est très souhaitable dans un programme de négociation automatisée.

Sycara et Zeng ont proposé le modèle Bazaar [17], un modèle de prise de décision séquentielle en mesure d'apprendre. Leur objectif final de recherche est de développer un modèle de négociation adaptative qui pourra produire un riche ensemble de comportements de négociation sans beaucoup d'efforts statistiques.

Le MIT Media Laboratory [31] a, quant à lui, développé un système multi-agents appelé Kasbah destiné à conduire des transactions consumer-to-consumer en ligne. Si un utilisateur veut acheter ou vendre un produit, il crée un agent, lui donne des instructions sur la stratégie à suivre et l'envoie dans le marché centralisé des agents. Les agents de Kasbah vont chercher les acheteurs ou les vendeurs potentiels, et négocient avec eux à la place de leurs propriétaires. La négociation dans Kasbah est simple. Une fois que les agents des vendeurs et ceux des acheteurs sont associés, la seule action valide dans le protocole de négociation consiste, pour les agents des acheteurs, à faire une offre aux agents des vendeurs sans restrictions de temps ou de prix. Les agents des vendeurs ne peuvent répondre que par oui ou non. Le système Kasbah comporte également un mécanisme d'évaluation de la confiance et de la réputation des agents, le "Better Business Bureau". Ainsi un agent pourra choisir de négocier avec un autre agent sur base de sa réputation.

Chapitre 3 : Les ventes aux enchères

3.1. Introduction

3.1.1. Les ventes aux enchères dans la réalité

Une vente aux enchères est un marché avec un ensemble explicite de règles déterminant l'allocation des ressources et des prix sur base des offres en provenance des participants du marché [2]. C'est un mécanisme pour vendre des produits indivisibles aux enchérisseurs. Le mécanisme des enchères fournit les règles pour faire les offres et réserve les biens à l'enchérisseur (ou aux enchérisseurs) en se basant sur un ensemble des règles prédéfinies.

Les enchères sont intéressantes surtout lorsqu'on vend un article dont la valeur est indéterminée. La vente d'un article aux enchères est plus flexible que la vente à prix fixe sur l'article. Elle consomme également moins de temps et coûte moins cher que la négociation d'un prix. Dans la négociation sur le prix, chaque offre et contre-offre est considérée séparément, mais dans une vente aux enchères, les offres concurrentes sont offertes simultanément.

N'importe quel article peut être vendu aux enchères : cela va des propriétés publiques, du bétail, du vin, aux fleurs, en passant par des voitures, etc. Le dénominateur commun est que la valeur de chaque article varie suffisamment pour en exclure une fixation absolue du prix. On peut donc dire que la vente aux enchères est une méthode basée sur la compétition pour allouer des biens rares.

Comme sur tout marché, dans une vente aux enchères, le vendeur essaie de gagner le maximum d'argent possible et l'acheteur, quant à lui, veut sortir le moins d'argent possible. Une vente aux enchères offre l'avantage de la simplicité dans la détermination des prix. La particularité d'une vente aux enchères est le fait que le prix n'est pas déterminé par le vendeur mais par les enchérisseurs. Toutefois, c'est le vendeur qui détermine les règles à utiliser. Elle garantit que le produit revient à celui qui lui donne le plus de valeur. Elle assure aussi que le vendeur reçoit une estimation générale de la valeur de son bien.

Contrairement à d'autres méthodes de vente, dans la vente aux enchères, le commissaire-priseur n'est pas le propriétaire des biens qu'il vend mais agit plutôt comme un agent intermédiaire pour les propriétaires des biens : les acheteurs connaissent la valeur de l'article mieux que le vendeur. Un vendeur qui a peur de suggérer un premier prix à cause de son ignorance, choisit de vendre aux enchères pour extraire l'information qu'il ne pourrait avoir autrement.

3.1.2. Stratégies des ventes aux enchères

Il existe plusieurs types des ventes aux enchères dont les principaux sont : les enchères anglaises, les enchères hollandaises, les enchères aveugles et les enchères Vickrey. Nous analysons en détail ces différents types au point 3.5. Il n'y a pas de stratégie unique pour les ventes aux enchères. Cela dépend du type d'enchères dont il est question. Dans un type donné d'enchères, des stratégies différentes qui peuvent être appliquées. Plusieurs variables interviennent dans la détermination de la stratégie à appliquer dans un type de vente aux enchères donné : la possibilité pour l'enchérisseur de prendre un risque, la quantité d'exemplaires de l'article vendu, l'utilisation future de l'article par l'acheteur.

Le vendeur doit choisir le type d'enchères qui lui convient et doit donc anticiper le comportement des enchérisseurs. La meilleure stratégie pour le vendeur est de révéler l'information concernant l'article vendu et de lier la détermination du prix à d'autres facteurs externes à la valeur de l'article, par exemple la rareté de l'article. Par ailleurs, l'enchérisseur doit essayer de prévoir le comportement des autres enchérisseurs : chaque enchérisseur fait sa propre estimation de la valeur de l'article et fait également une estimation de la valeur que les autres enchérisseurs donneront à cet article. Une bonne enchère est souvent le résultat de bonnes prévisions sur le comportement des autres.

Il est à noter que les acheteurs ont généralement deux motivations qui les poussent à participer à une vente aux enchères de quelque type que ce soit :

- soit l'acheteur veut acquérir un article pour une utilisation personnelle, il donne alors une évaluation personnelle de l'article à acheter. Dans ce cas tous les enchérisseurs ont des évaluations personnelles et ils ont tendance à garder cette information secrète. Si le vendeur connaît d'avance la plus haute valeur que les acheteurs donnent à l'article, la vente aux enchères ne présente plus d'intérêt pour lui.

- soit l'acheteur veut acquérir un article pour le revendre ou pour une utilisation commerciale. Dans ce cas, une offre individuelle ne dépend pas seulement d'une évaluation personnelle mais aussi de l'estimation de la valeur du bien par les futurs acheteurs. Chaque enchérisseur, dans ce cas-ci, essaie de deviner le prix final de l'article. L'article a donc la même valeur pour tout le monde, mais le montant exact est inconnu. Par exemple, l'achat d'un terrain pour ses droits miniers. Chaque enchérisseur a une information différente et une évaluation différente du terrain, mais chacun doit deviner ce que le terrain pourrait finalement rapporter.

Le comportement des enchérisseurs change en fonction de la motivation qui les pousse à agir.

3.2. *Différents types d'enchères*

On classe les enchères en fonction de leur degré d'ouverture (enchères ouvertes, enchères secrètes), en fonction également de l'évolution du prix (enchères ascendantes et enchères descendantes) [2, 3, 4, 9, 10]. Généralement, on distingue quatre types d'enchères : les enchères anglaises (English auction), les enchères hollandaises (Dutch auction), les enchères aveugles (First-Price sealed-bid) et les enchères Vickrey (uniform second-price). Outre ces quatre types d'enchères, il en existe d'autres tel que le "reverse auction".

3.2.1. Les enchères anglaises

a) Description

Les enchères anglaises sont une forme la plus courante de vente aux enchères. C'est un type d'enchères ouvertes où le prix est annoncé à haute voix (à la criée) et évolue de façon ascendante.

b) Règle de fonctionnement

Le vendeur choisit le prix de départ appelé le prix de réserve. Ce prix de réserve représente le prix minimum auquel le vendeur est prêt à vendre son article. Les acheteurs doivent placer des enchères supérieures ou égales au prix de réserve. Le commissaire-priseur sollicite successivement des offres plus élevées des acheteurs jusqu'à ce que plus personne n'augmente le

prix. L'article est vendu à l'enchérisseur qui a offert le prix le plus élevé. L'enchérisseur qui gagne paye la plus grande valeur proposée.

Il existe une forme particulière d'enchères anglaises : **les enchères avec prix de réserve**. Les enchères avec prix de réserve sont des enchères normales pour lesquelles le vendeur choisit un prix de réserve qui peut être différent du prix de départ. Les acheteurs peuvent placer des enchères inférieures au prix de réserve, mais le vendeur n'est pas obligé de vendre tant que le prix demeure inférieur au prix de réserve. Si l'enchère finale est inférieure au prix de réserve, l'enchère ferme sans gagnant. Les vendeurs utilisent le format des enchères avec réserve quand ils ne savent pas quel prix de départ utiliser pour leurs articles. Cela leur permet d'évaluer la demande pour leurs articles. Les vendeurs peuvent baisser le prix de réserve à tout moment avant la fermeture de l'enchère (mais ils ne peuvent pas l'augmenter). Si plusieurs exemplaires du même bien sont proposés, tous les exemplaires sont vendus au même prix.

Le commissaire-priseur peut ne pas dévoiler le prix de réserve et commencer les enchères sans annoncer le prix le plus bas acceptable. L'une des raisons qui pousse à faire ce choix est de contrecarrer les acheteurs qui pourraient se mettre ensemble pour ne pas se surenchérir mutuellement dans l'objectif de baisser la valeur de l'offre gagnante.

c) Stratégie de l'acheteur

La stratégie d'un enchérisseur est fonction de sa précédente estimation sur l'évaluation des autres enchérisseurs. Son enchère peut être modifiée si l'information change. La meilleure stratégie est de surenchérir d'un petit montant par rapport à la plus haute enchère précédente jusqu'à ce qu'on atteigne la valeur maximale que l'on s'est fixé. Cette stratégie est optimale parce qu'elle permet à un enchérisseur de payer le produit vendu à un prix inférieur à sa valeur réelle.

d) Avantages/ Inconvénients

L'avantage des enchères anglaises est de permettre à l'enchérisseur d'obtenir de l'information en observant le prix à partir duquel les autres participants abandonnent les enchères. Cet atout permet à un enchérisseur de connaître l'évaluation que les autres avaient sur la valeur du bien et par conséquent de revoir sa propre évaluation.

Ce type d'enchères peut être moins avantageux pour le vendeur car il ne lui permet pas de gagner la valeur maximale d'un bien étant donné que les acheteurs n'augmentent leurs mises qu'à petits coups. Un autre désavantage de ce système est l'obligation pour les acheteurs d'être présents pendant la vente pour pouvoir surenchérir.

3.2.2. Les enchères hollandaises

a) Description

C'est la forme inversée du type d'enchères anglaises. C'est un type d'enchères ouvertes où le prix est annoncé à la criée et évolue de façon descendante.

b) Règle de fonctionnement

Le vendeur fixe un prix de départ beaucoup plus élevé que son prix de réserve. Le prix proposé par le commissaire-priseur baisse progressivement tout au long de la vente jusqu'au moment où un des participants décide d'acheter, et donc d'interrompre la vente. Si le prix baisse au delà de la limite que s'est fixée le vendeur (son prix de réserve), le produit est retiré de la vente. Si plusieurs exemplaires du produit sont proposés, le premier gagnant paye plus cher que les suivants, la vente se termine lorsque tous les exemplaires sont vendus.

c) Stratégie de l'acheteur

L'acheteur doit décider du montant maximum qu'il est prêt à offrir. Il doit décider à quel moment stopper les enchères en se basant sur sa propre évaluation de l'objet et de ses précédentes suppositions sur l'évaluation des autres enchérisseurs.

d) Avantages/ Inconvénients

L'avantage de ce type d'enchères, pour le vendeur, est que si un acheteur désire absolument obtenir le produit, il peut offrir une grosse somme et arrêter la vente. Ce montant peut dépasser la valeur maximale estimée du bien.

3.2.3. Les enchères aveugles

a) Description

Les enchères aveugles représentent le type d'enchères le plus simple et aussi le plus excitant. C'est un type d'enchères secrètes et, par conséquent, on ne tient pas compte de l'évolution du prix.

b) Règle de fonctionnement

Chaque acheteur n'émet qu'une seule offre, unique et secrète. A la clôture des enchères, l'offre la plus élevée remporte la vente si elle est au moins égale au prix de réserve. Il y a deux parties dans l'organisation de ce type d'enchères, la phase pendant laquelle les enchérisseurs proposent leurs prix et la phase de résolution pendant laquelle on compare toutes les offres d'achat et on détermine le gagnant.

Dans ce type d'enchères, le nombre d'exemplaires de l'article vendu détermine la manière dont le prix sera fixé. Si un seul exemplaire de l'article est mis aux enchères, il est vendu au prix le plus élevé proposé (first-price). Par contre, si on a plusieurs exemplaires du même produit, tous les enchérisseurs gagnants ne payent pas le même prix : on parlera de vente aux enchères discriminatoire. Les offres d'achat sont triées de manière décroissante : les exemplaires sont vendus au prix le plus élevé jusqu'à épuisement du stock. Les gagnants payent le montant exact qu'ils ont proposé.

c) Stratégie de l'acheteur

Il est difficile de spécifier une stratégie unique pour ce type d'enchères : en effet, la maximisation du profit d'une enchère dépend des actions des autres. L'acheteur peut faire des offres très élevées pour être sûr de gagner ou, à l'inverse, il peut faire des offres très basses en espérant tirer un bénéfice important si son offre est gagnante à la clôture des enchères. La meilleure stratégie pour ce type est de faire des offres qui approchent le consensus du marché par le bas.

d) Avantages/ Inconvénients

Ce type présente un avantage pour les enchérisseurs : une enchère élevée augmente la probabilité de gagner mais diminue d'autant le profit si l'enchérisseur gagne.

3.2.4. Les enchères Vickrey

a) Description

Les enchères Vickrey tirent leur appellation de William Vickrey, un Prix Nobel des Sciences Economiques. Ce type d'enchères est assez similaire aux enchères aveugles, dans le sens où chaque enchérisseur ignore la mise des autres enchérisseurs (enchères secrètes).

b) Règle de fonctionnement

Le fonctionnement des enchères Vickrey est similaire à celui des enchères aveugles. Elles se différent des enchères aveugles dans le sens où le gagnant, celui dont l'enchère est la plus élevée, ne paiera que le prix de la deuxième enchère la plus élevée, c'est-à-dire moins que ce qu'il a proposé.

Par exemple, si nous avons trois offres d'achat, l'une A de 100 F, l'autre B de 200 F et le troisième C de 250 F, l'enchérisseur de C va gagner mais il ne payera que 200 F. Dans le cas où le produit est proposé en plusieurs exemplaires, tous les gagnants payent le même prix : 200 F dans notre exemple.

c) Stratégie de l'acheteur

La stratégie dominante pour l'acheteur dans une vente aux enchères Vickrey est de faire une offre équivalente à son prix de réserve (c'est-à-dire le maximum qu'il est prêt à offrir). Il augmente ainsi ses chances de gagner si toutes les autres offres sont inférieures à son prix de réserve.

d) Avantages/ inconvénients

Ce type d'enchères est intéressant pour les enchérisseurs. Ils savent qu'ils ne pourront pas payer le prix le plus élevé. Les enchérisseurs les plus agressifs reçoivent certainement l'objet vendu mais payent un prix proche du consensus du marché. Le prix que l'enchérisseur gagnant

paye n'est déterminé que par les enchères des concurrents et ne dépend d'aucune action qu'il pourrait entreprendre.

3.2.5. Reverse Auction

a) Description

La "Reverse Auction" est une vente aux enchères particulière dans laquelle ce sont les acheteurs qui exposent leurs attentes en terme de produits, quant aux vendeurs, ils s'efforcent d'offrir le meilleur prix. C'est un type d'enchères ouvertes où les prix proposés par les vendeurs le sont de manière descendante (comme pour les enchères hollandaises, à la différence qu'ici ce sont les vendeurs qui diminuent le prix alors que dans les enchères hollandaises, ce sont les acheteurs).

b) Règle de fonctionnement

L'acheteur place une demande, les vendeurs font leurs offres par rapport à la demande de l'acheteur. Les enchères sont faites de façon décroissante sur le prix. A la clôture des enchères, l'acheteur choisit le gagnant, celui qui a offert le prix le plus bas. L'acheteur sélectionne le gagnant en se basant sur ses propres critères. Les facteurs qui influencent le choix de l'acheteur sont les suivants : le prix proposé par le vendeur, les termes de la vente, la réputation du vendeur, sa qualité, la date de livraison, la localisation, etc.

c) Stratégie du vendeur

Le vendeur peut influencer le choix de l'acheteur en proposant des conditions attirantes en dehors du prix de l'article.

d) Avantages/ Inconvénients

L'avantage de ce type d'enchères réside dans le fait que les vendeurs peuvent élargir leur capacité de distribution. La flexibilité de la fixation des prix est un des autres avantages de ce type d'enchères. Les acheteurs, quant à eux, peuvent faire leurs achats ou proposer leurs demandes et trouver ce dont ils cherchent à un prix très compétitif.

3.3. Les enchères en ligne ou virtuelles

3.3.1. Mécanisme des enchères en ligne

Contrairement aux enchères "réelles" (dans des salles de vente) qui ne durent que quelques heures au maximum, les enchères virtuelles peuvent durer plusieurs jours. Par conséquent, les sites qui proposent des ventes aux enchères mettent souvent à la disposition des acheteurs un système automatique (robot d'enchères ou agent d'enchères). Ce dernier surenchérit à la place des acheteurs afin de leur éviter de rester en ligne en permanence pour suivre les enchères minute par minute. Il suffit à l'acheteur de confier au robot d'enchères les instructions sur le prix maximum qu'il est prêt à miser sur la vente, et le robot émettra des offres d'achat contre les autres utilisateurs ou leurs robots, jusqu'au prix plafond de l'acheteur. L'acheteur indique également au robot la marche à suivre si les enchères dépassent son prix plafond : il peut, par exemple, avertir l'intéressé par e-mail, afin de lui permettre de changer sa position et augmenter son prix plafond.

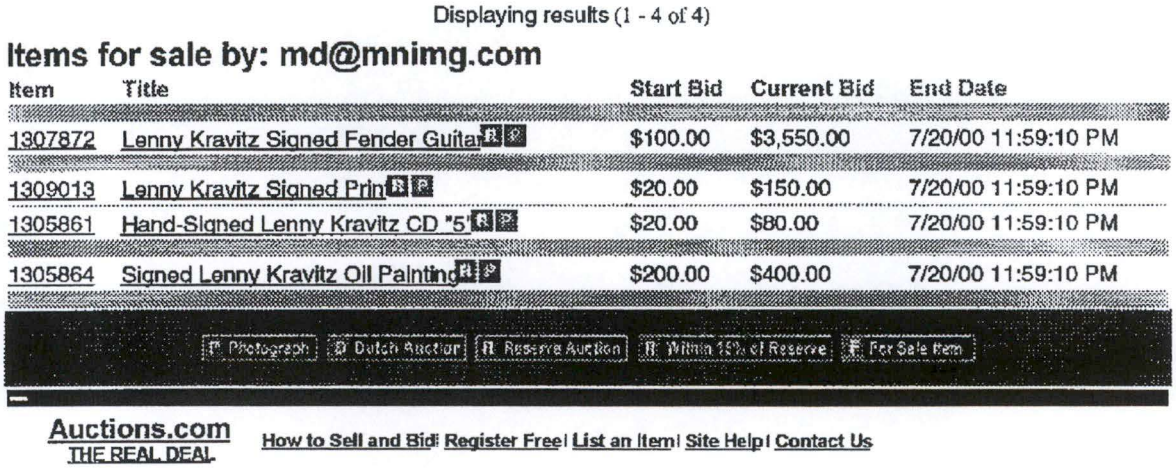


Figure 3.1. : Vente aux enchères en ligne

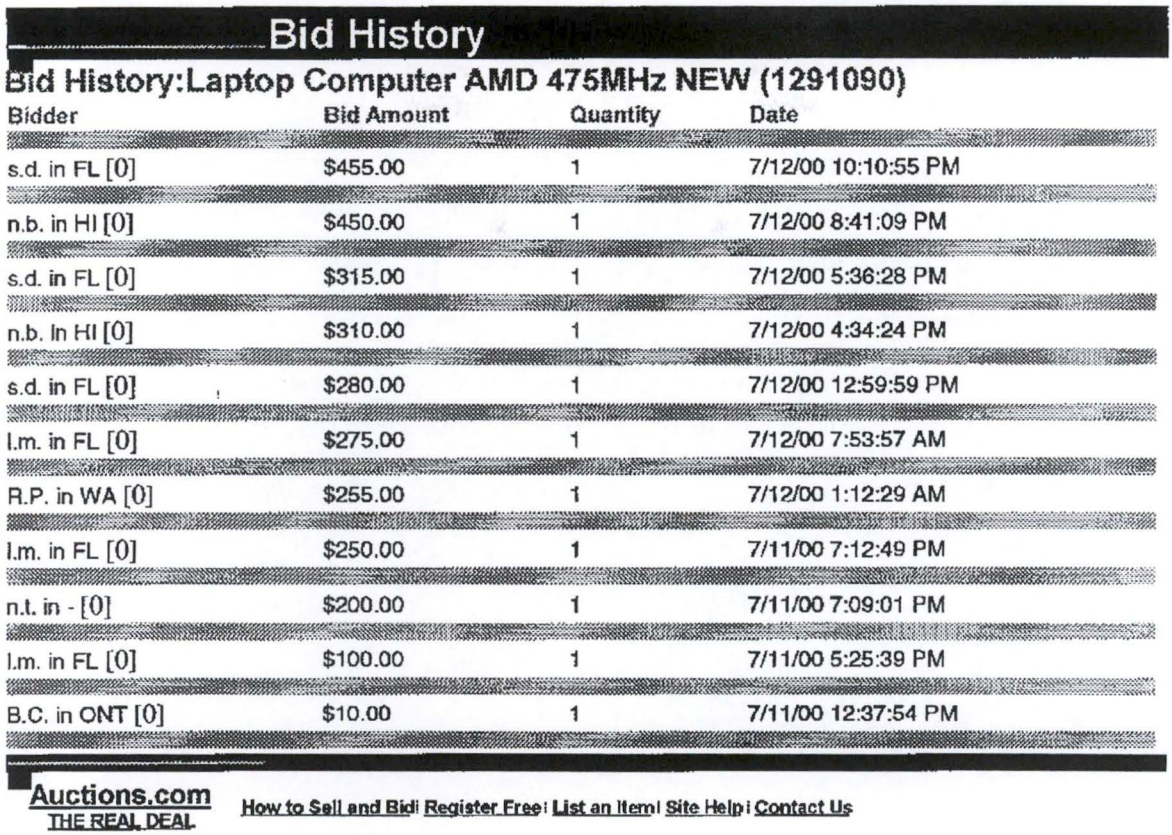
Comme la vente aux enchères sur le Web peut durer plusieurs jours, l'enchérisseur peut augmenter à tout moment son offre maximale même s'il est le gagnant à un moment donné avant la fin de la vente. Il est évident que le montant maximum de chaque enchérisseur est gardé secret vis-à-vis des autres enchérisseurs.

La Figure 3.1. représente un exemple d'une proposition de vente des produits aux enchères avec une description du type d'enchères à côté de chaque produit.

Dans certaines ventes aux enchères en ligne, on fixe le montant des incréments. Un enchérisseur qui veut surenchérir doit augmenter les enchères du montant fixe des incréments. Si un enchérisseur place une offre maximale supérieure à ce montant, cette offre doit être un multiple du montant des incréments. Par exemple, si à moment donné l'offre la plus élevée d'un article est de 100 F et le montant des incréments fixé à 25 F. Si un enchérisseur place une offre maximale de 140 F, cet enchérisseur sera l'actuel gagnant mais l'offre gagnante ne vaudra pas 140 F mais plutôt 125 F, et son offre maximale sera réduite à 125 F.

La durée de la vente est toujours déterminée à l'avance. Elle peut être déterminée soit par le système automatique, soit par le vendeur, soit par l'acheteur (reverse auction). La vitesse à laquelle se déroulent les enchères dépendra de la nature du produit vendu. Pour les denrées périssables et autres produits à durée de vie courte, comme les fleurs, par exemple, on optera pour des enchères rapides. A la fermeture de la vente, l'acheteur ayant fait l'offre la plus élevée emporte la vente et le gagnant est alors averti par e-mail.

L'historique des enchères est disponible pour que les enchérisseurs puissent analyser l'évolution des enchères. La figure 3.2. est un exemple de l'historique d'une vente aux enchères en ligne.



Bid History

Bid History:Laptop Computer AMD 475MHz NEW (1291090)

Bidder	Bid Amount	Quantity	Date
s.d. in FL [0]	$455.00	1	7/12/00 10:10:55 PM
n.b. in HI [0]	$450.00	1	7/12/00 8:41:09 PM
s.d. in FL [0]	$315.00	1	7/12/00 5:36:28 PM
n.b. in HI [0]	$310.00	1	7/12/00 4:34:24 PM
s.d. in FL [0]	$280.00	1	7/12/00 12:59:59 PM
l.m. in FL [0]	$275.00	1	7/12/00 7:53:57 AM
R.P. in WA [0]	$255.00	1	7/12/00 1:12:29 AM
l.m. in FL [0]	$250.00	1	7/11/00 7:12:49 PM
n.t. in - [0]	$200.00	1	7/11/00 7:09:01 PM
l.m. in FL [0]	$100.00	1	7/11/00 5:25:39 PM
B.C. in ONT [0]	$10.00	1	7/11/00 12:37:54 PM

Figure 3.2. : Historique d'offres dans une vente aux enchères en ligne

Certains sites des ventes aux enchères en ligne proposent également un système de classement les utilisateurs. Ce système permet d'évaluer la confiance que l'on peut faire à un utilisateur et d'exclure du système des personnes qui ne sont pas crédibles.

3.3.2. Caractéristiques importantes dans une vente aux enchères en ligne

Comme nous l'avons vu dans le chapitre précédent, les enchères structurées sont facilement automatisables et conviennent parfaitement à la négociation électronique sur l'Internet. En effet, elles présentent des caractéristiques qui conviennent à la mise en place d'agents intelligents :

- la négociation y est limitée à une seule dimension : le prix, ce qui est relativement facile à manipuler pour un agent logiciel;

- le problème de l'ontologie de l'offre n'intervient pas ici car l'article à vendre est exposé et les enchérisseurs peuvent l'observer pour analyser ses spécifications;

- les règles de la négociation sont clairement définies;

- la stratégie de négociation peut être résolue à partir du point de vue du vendeur qui peut rendre sa stratégie publique sans pour autant se retrouver désavantagé par rapport aux enchérisseurs.

Les articles qui conviennent pour les enchères sur le Web doivent respecter les caractéristiques suivantes :

- leur ontologie doit être relativement simple, c'est-à-dire que l'article doit pouvoir être décrit entièrement par un humain ou par une machine;
- la négociation ne peut porter que sur le prix;
- la valeur de l'article n'est pas suffisamment élevée pour nécessiter une négociation humaine qui, elle coûte cher;
- chaque client a tendance à évaluer l'article différemment : en conséquence, les différences entre les évaluations des clients pourraient justifier la nécessité d'une négociation;
- les clients ont tendance à être géographiquement dispersés, ce qui rendrait la négociation de face-à-face difficile.

Les articles qui peuvent être téléchargés conviennent pour les ventes aux enchères sur l'Internet car le gagnant des enchères peut recevoir le produit directement.

Pour les biens physiques, il faut pouvoir intégrer dans le mécanisme des enchères un système d'inventaire des biens et mettre en place des mécanismes d'expédition pour faire parvenir à un client, qui se trouverait physiquement très loin du vendeur, le produit acquis.

3.3.3. Fonctionnement sur le Web des enchères anglaises et hollandaises

a) Les enchères anglaises

Les enchères anglaises sont très utilisées sur le Web. Le mécanisme utilisé sur le Web est identique à celui utilisé dans les salles de vente. Le système automatique affiche le prix de réserve ou le prix de départ. Au fur et à mesure que les offres sont proposées, elles sont également affichées pour que tous les enchérisseurs en prennent connaissance et puissent surenchérir. Les surenchères

peuvent se faire automatiquement en utilisant le robot d'enchères ou manuellement. Les enchères anglaises qui proposent un robot d'enchères pour surenchérir à la place des acheteurs permettent à ces derniers de ne pas rester constamment en ligne pour suivre l'évolution des enchères. La durée des enchères peut ainsi être déterminée sans se préoccuper de l'immobilisation des acheteurs.

Dans les enchères avec prix de réserve pour lesquelles on utilise un robot d'enchères, si le prix maximum proposé par un enchérisseur est inférieur au prix de réserve du vendeur, alors le robot placera ce maximum comme offre actuelle de l'enchérisseur. Mais si ce maximum est supérieur au prix de réserve, alors le robot proposera le prix de réserve comme offre actuelle de l'enchérisseur.

La figure 3.3. est un exemple d'une vente aux enchères de type anglaise sur le Web et la figure 3.4. l'historique de la vente représentée par la figure 3.3.

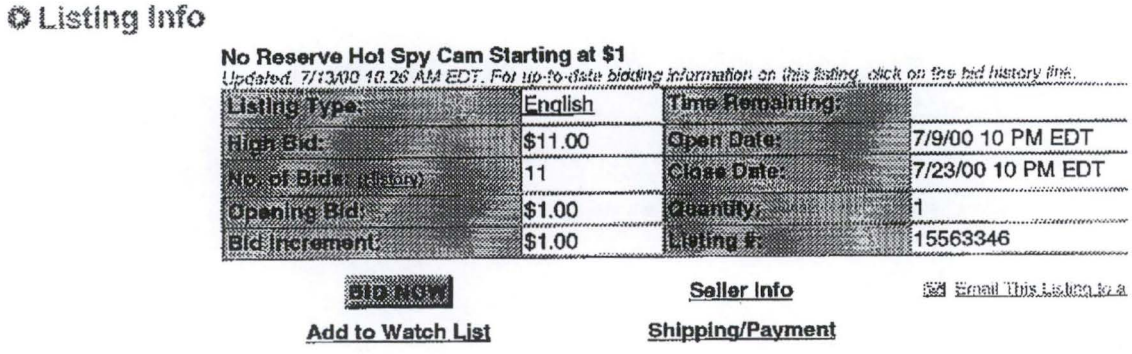


Figure 3.3. : Etat d'une vente aux Enchères anglaises

No Reserve Hot Spy Cam Starting at $1

Winning Bid for this item: $11.00
Opening Bid: $1.00

Bid Now

Member	Bid Amount	Qty(Won)	Bid Date (EDT)	Status
christophertan88 (0)				
	$11.00	1(1)	7/13/00 10:20 AM	Winning
THARD1234 (0)				
	$10.00	1(0)	7/13/00 12:29 AM	Losing
JOSE.MATA (0)				
	$9.00	1(0)	7/12/00 8:31 AM	Losing
kwolfsn69 (0)				
	$9.00	1(0)	7/12/00 12:02 PM	Losing
huskeycb (0)				
	$7.00	1(0)	7/12/00 3:50 AM	Losing
RYAN-LLOYD (0)				
	$3.00	1(0)	7/11/00 2:10 PM	Losing

Number of bids may reflect multiple bids by the same bidder. Bids are ordered by amount then quantity then time of bid.

Figure 3.4. : Historique d'une vente aux enchères anglaises

b) Les enchères hollandaises

Sur le Web, nous avons remarqué que le fonctionnement des enchères hollandaises est dans certains cas différent de celui que nous avons décrit au point 3.2.2. Dans les enchères hollandaises sur le Web, un prix de départ est fixé ainsi que le montant des incréments. Les enchérisseurs font leurs offres de manière ascendante. A la fermeture de la vente, le gagnant est celui dont l'offre est la plus élevée mais il paie un montant égal à la deuxième offre la plus élevée.

Prenons un exemple pour mieux comprendre comment fonctionnent ces enchères sur le Web. Si 10 articles sont offerts dans une vente en ligne et le prix de réserve fixé à 10 F la pièce. Si à la clôture des enchères il y a 20 offres de 10 F chacune, les dix premiers enchérisseurs vont être déclarés gagnants. On départage les ex æquo en suivant l'ordre chronologique des placements

d'enchères. Mais si, par contre, avant que les enchères ne soient clôturées, un nouvel enchérisseur place une enchère de 30 F pour un article, comme cette personne a placé l'enchère la plus élevée, elle sera gagnante mais ne payera que 10 F pour son article. Les 9 articles restants vont être attribués aux 9 premiers enchérisseurs qui ont placé 10 F. Tout le monde paye le même prix mais en donnant priorité à ceux qui offrent le montant le plus élevé et, en cas d'ex æquo, aux premiers enchérisseurs.

Pour ce type d'enchères, il n'existe pas de système automatique de surenchérissement à la place des acheteurs.

La figure 3.5. est un exemple de l'état d'une vente aux enchères de type hollandaise à un moment donné avant la fin de la vente. La figure 3.6. l'historique de cette vente.

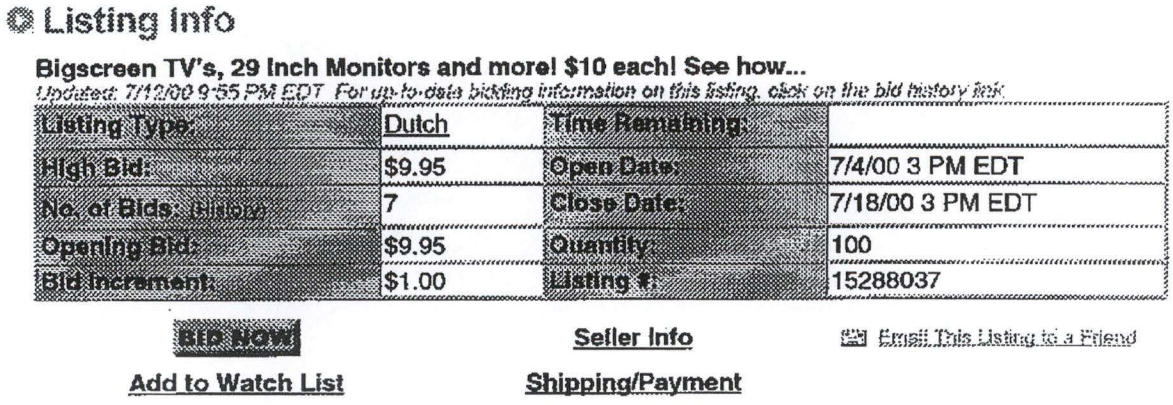


Figure 3.5. : Etat d'une vente aux enchères hollandaises

High Bid for this item: $39.95
Opening Bid: $9.95

Bid Now

Member	Bid Amount	Qty(Won)	Bid Date (EDT)	Status
sbhe911 (0)				
	$39.95	1(1)	7/12/00 9:44 PM	Winning
gonzalez69 (0)				
	$14.95	3(3)	7/7/00 7:51 PM	Winning
dduchess18 (0)				
	$14.95	1(1)	7/11/00 12:03 PM	Winning
JMAWAD (0)				
	$11.95	1(1)	7/5/00 11:59 PM	Winning
BILLY187_99 (0)				
	$9.95	1(1)	7/4/00 3:53 PM	Winning
ronhavingfun@aol.com (0)				
	$9.95	1(1)	7/7/00 8:49 AM	Winning
skatemuska22 (0)				
	$9.95	1(1)	7/10/00 1:12 PM	Winning

Number of bids may reflect multiple bids by the same bidder. Bids are ordered by amount then quantity then time of bid.

Figure 3.6. : historique d'une vente aux enchères hollandaises

Dans le monde financier, le fonctionnement des ventes aux enchères du type hollandais (Dutch Auction) est identique à celui des ventes aux enchères Vickrey. Les sites Internet se sont sans doute inspirés de l'appellation du monde financier plutôt que de l'appellation scientifique pour déterminer les règles de fonctionnement des ventes aux enchères hollandaises.

3.3.4. Evaluation comparative des différents types des ventes aux enchères

Déterminer le meilleur type de vente dépend de plusieurs variables. Le vendeur et l'acheteur ont des perspectives différentes. Certaines ventes aux enchères encouragent la tricherie, d'autres la rendent plus difficile : en fonction de cet élément, acheteurs et vendeurs opteront pour

l'un ou l'autre type de vente. Un vendeur estimera que la vente est un succès s'il perçoit que le produit mis aux enchères suscite une vive compétition chez les acheteurs potentiels.

Tous les types d'enchères sont susceptibles d'être manipulés. Les enchères anglaises sont plus susceptibles de collusions entre enchérisseurs que les enchères aveugles. Ceci pourrait expliquer la popularité des enchères aveugles, même si les enchères anglaises génèrent des revenus plus importants. Dans les enchères aveugles, les collusions peuvent se former entre le commissaire-priseur et un ou plusieurs enchérisseurs. Ce type d'enchères est moins prédisposé aux collusions entre enchérisseurs parce que les enchères secrètes poussent les participants à offrir des enchères élevées plutôt que de s'entendre sur le prix pour tromper les autres. La même raison vaut pour les enchères hollandaises, même si les offres sont prononcées à la criée, parce que le premier enchérisseur qui décide d'acheter arrête la vente.

L'automatisation des ventes aux enchères permet d'éviter ce genre de comportement car les robots ne sont pas programmés pour créer des collusions.

S'il est possible d'utiliser le mécanisme de confidentialité sur l'Internet, de sorte que les acheteurs potentiels ne sachent pas entrer en contact entre eux pour former des groupes, le risque de collusions pourrait être ainsi évité dans les enchères en ligne.

Chapitre 4 : L'e-contracting

4.1. Introduction

Du point de vue légal (juridique), on définit l'e-contracting comme étant un terme pour décrire l'application des technologies de l'information avancées à la réalisation des contrats [1]. Les technologies de l'information sont utilisées pour rendre la réalisation des contrats sur l'Internet conforme aux obligations juridiques (obligations de signature, d'information claire et précise, etc.).

L'e-contracting implique l'échange, entre vendeurs et acheteurs, de messages structurés conformément à un format pré-arrangé pour que leur contenu soit traitable par une machine (automatisable) et entraîne automatiquement des obligations contractuelles [2]. L'e-contracting est une application spécifique basée sur la messagerie électronique.

Cette définition du terme "Electronic contracting" considère principalement le processus de négociation, d'échange des messages ou des accords entre parties utilisant les technologies de l'information ou les outils du commerce électronique. La signature du contrat négocié n'est pas prise en compte malgré le rôle important qu'elle joue.

Pour être plus complet, l'e-contracting" devrait couvrir deux activités principales (Figure 4.1.) :

1. la négociation proprement dite : elle concerne la situation des négociations, l'échange interactif des messages entre les acheteurs et vendeurs;

2. la signature des contrats négociés électroniquement.

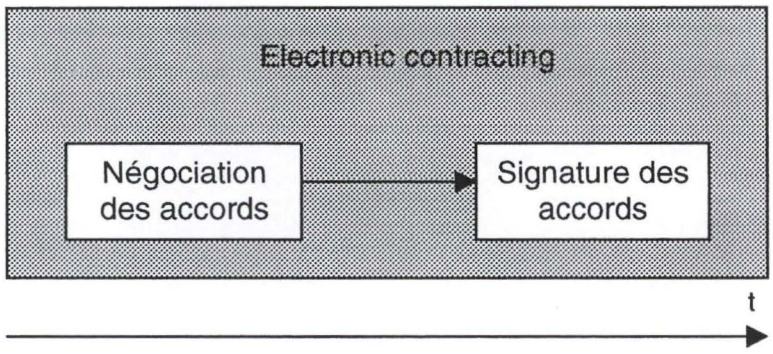


Figure 4.1. : Processus de négociation des contrats par l'Internet

4.2. Besoins des utilisateurs

Il est important d'identifier les besoins réels des utilisateurs en terme d'outils d'aide à la négociation sur le Web. Selon une étude menée par A. Runge [1] (Figure 4.2.), 65,83% des utilisateurs d'Internet (acheteurs et vendeurs confondus) considèrent l'e-contracting comme étant important, voire très important.

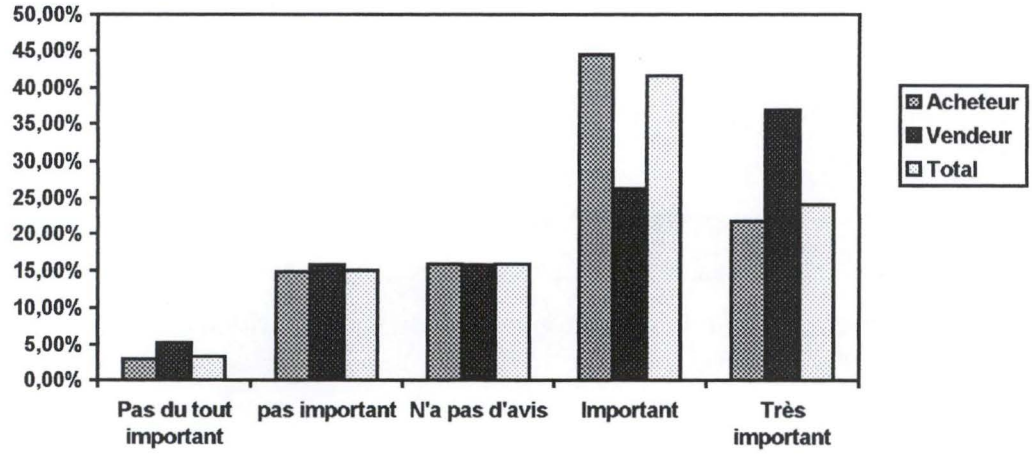


Figure 4.2. : Les besoins des utilisateurs pour l'e-contracting

Lorsqu'on les interroge sur la prise en charge de la négociation par un outil automatique (un "Négociateur"), 65.52% des utilisateurs déclarent préférer négocier eux-mêmes. Seulement Les négociations 17.93% des utilisateurs accepteraient d'utiliser des négociations faites par un

ordinateur. Ils ne sont plus que 13.10% à souhaiter avoir recours à un tiers de confiance (Figure 4.3.).

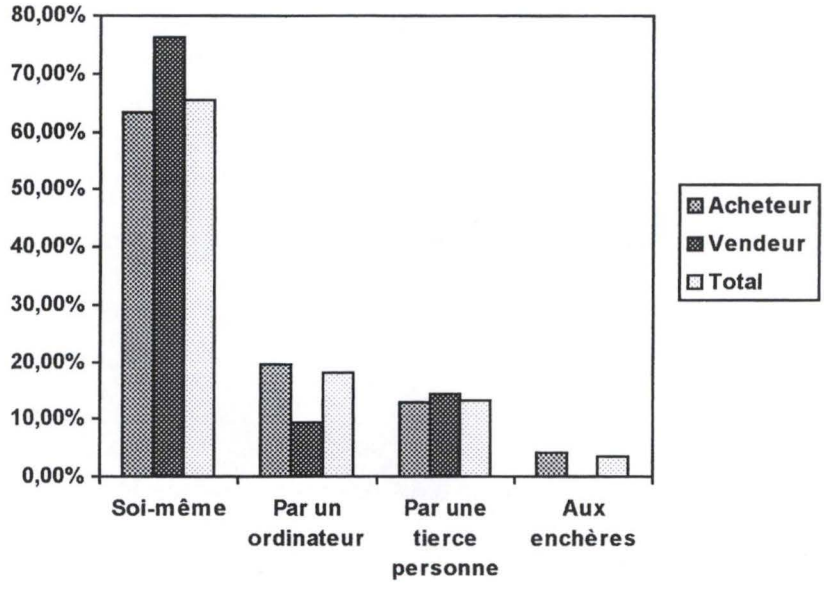


Figure 4.2. : Prise en charge de la négociation

Cette étude nous permet de conclure au fait que les utilisateurs ne font pas encore suffisamment confiance aux outils électroniques pour réaliser la tâche de la négociation. Cela est sans doute dû au fait qu'il n'existe pas encore de système de négociation automatisée suffisamment performant (susceptible de réaliser la négociation mieux qu'un humain). Mais comme nous l'avons vu dans le chapitre 2, les recherches dans ce domaine sont en cours et nous serons peut être en mesure, dans le futur, d'attribuer la tâche de négociation à un ordinateur.

4.3. Aspect juridique de l'e-contracting

La négociation sur le Web doit pouvoir se faire en respectant les lois en vigueur en matière de contrats à distance. La capacité de négocier de façon automatique sur l'Internet n'aura pas beaucoup de valeur sur le plan juridique si les accords obtenus à la suite d'une telle négociation ne peuvent être reconnus de façon légale. Un contrat n'est authentique légalement que s'il est signé par les parties contractantes.

4.3.1. Problèmes liés à la nature de l'Internet

Le problème de la négociation de contrats sur l'Internet est lié à la méconnaissance réciproque des parties. Les utilisateurs n'ont pas confiance non plus dans le système pour la sécurité des échanges. Afin de pallier ces imperfections et pour renforcer la confiance des utilisateurs, on voit apparaître de nouveaux intermédiaires : les tiers de confiance.

La Figure 4.4. illustre le cas d'un site de vente aux enchères nommé Auctions.com et qui porte le sigle ***Trust-e*** attribué par le tiers de confiance. Auctions.com en présentant le sigle ***Trust-e*** sur ses pages Web, s'engage à protéger les données portant sur la confidentialité de ses clients.

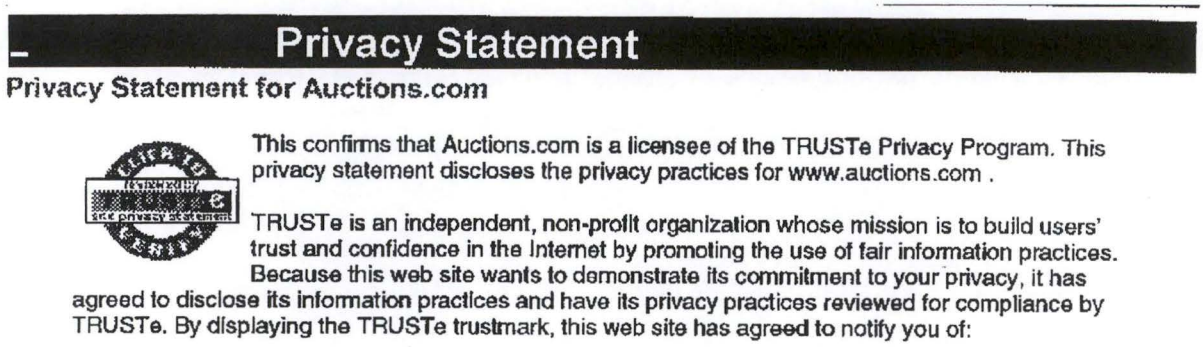


Figure 4.4. : Sigle représentant une licence obtenue auprès d'un tiers de confiance

L'insuffisance de réglementation générale et universelle ne contribue pas non plus à renforcer la confiance. Etant donné l'aspect mondial du commerce électronique, les utilisateurs se trouvent souvent confrontés à des réglementations différentes d'un pays à un autre.

L'absence de reconnaissance de la signature électronique pose le problème de la preuve dans le commerce électronique, les accords conclus lors des négociations l'Internet risquant de ne pas avoir de valeur juridique.

4.3.2. Solutions proposées par le législateur

Etant donné que les activités commerciales se multiplient sur l'Internet, les législateurs des différents pays prennent des mesures pour reconnaître et réglementer les activités qui se réalisent sur le Web. Nous n'allons traiter dans ce travail que le cas de la réglementation de la Commission Européenne.

La proposition de directive du 1er septembre 1999 [12] sur le commerce électronique de la Commission Européenne a pour but d'offrir un nouveau cadre juridique aux contrats conclus par voie électronique. L'objectif de cette réglementation est de supprimer les obstacles juridiques à la conclusion de contrats électroniques qui existent dans le droit des contrats des Etats membres. Elle vise aussi à établir des obligations spécifiques pour les contrats électroniques, dans le but de promouvoir la transparence des rapports commerciaux en ligne et, en particulier, protéger les consommateurs.

L'article 9 § 1 de cette proposition de directive vise à éliminer les exigences de forme qui existent dans le processus contractuel habituel et qui peuvent empêcher l'utilisation des contrats électroniques. Cet article ordonne aux Etats membres de supprimer tous les obstacles qui peuvent soit empêcher l'utilisation effective des contrats électroniques, soit les priver d'effet ou de validité. En particulier, en ce qui concerne la suppression des exigences de forme des contrats, les Etats membres devront supprimer la nécessité d'utiliser un support papier, par exemple. Ils peuvent aussi exprimer ces exigences de façon suffisamment ouverte de manière à admettre les équivalents électroniques, tel l'écrit sur support électronique. Cette obligation s'applique à tout le processus contractuel, qui s'étend de la phase pré-contractuelle à celle de la conclusion et de l'exécution du contrat.

L'article 5 § 1 de la proposition de directive vient résoudre le problème posé par les signatures électroniques. Il stipule que les signatures électroniques qui remplissent certaines conditions devront être traitées de manière équivalente aux signatures manuscrites. Cette directive reconnaît, par cet article, que la signature électronique peut satisfaire les fonctions de la signature manuscrite, notamment l'identification du signataire et la manifestation de volonté de ce dernier de s'approprier le contenu de l'acte auquel se réfère la signature, par diverses méthodes d'authentification. Les Etats membres devront donc introduire, dans leur législation, une équivalence entre la signature électronique et la signature manuscrite.

Une des exigences fondamentales en droit des contrats est le principe selon lequel la validité d'un contrat suppose l'existence d'une volonté de contracter. Ce principe entraîne le respect du principe de transparence en vertu duquel les parties contractantes doivent disposer de toutes les informations relatives au contrat, y compris les conditions générales, préalablement à la conclusion de celui-ci. Cette obligation de transparence est traduite dans l'article 10 de la proposition de directive et a pour objectif de s'assurer que le consommateur qui veut conclure un contrat par des moyens électroniques dispose de tous les moyens nécessaires pour prendre connaissance de l'existence du contrat et de son contenu, de façon à ce que le consentement soit donné librement et sans erreur. Ces dispositions visent à éviter un engagement non souhaité, par la suite d'un consentement affecté d'un vice, ce qui serait le cas si un consommateur "cliquait" par erreur sur l'icône "j'accepte" au lieu de l'icône "je n'accepte pas".

Le moment de la conclusion du contrat a des effets juridiques importants. Il détermine la prise d'effet du contrat. En matière de vente, aussitôt le contrat formé, l'acheteur devient en principe propriétaire du bien vendu et supporte la charge des risques. Dès cet instant, l'offre et l'acceptation ne sont plus révocables. Dans la plupart des Etats membres, il est prévu qu'entre parties éloignées l'une de l'autre, un contrat est conclu dès l'instant où l'offrant a eu la possibilité de prendre connaissance de l'acceptation de son offre par son interlocuteur.

Le législateur fait des efforts pour adapter la législation à la nouvelle forme du commerce sur l'Internet. Il est donc important lorsqu'on réalise des systèmes informatiques d'aide à la négociation sur l'Internet de tenir compte des obligations imposées par le législateur sur la réalisation des contrats sur l'Internet. Dans la réalisation du prototype que nous proposons dans la deuxième partie de ce travail, nous essayons de prendre en compte ces exigences juridiques.

DEUXIEME PARTIE

Cette partie ne comprend qu'un seul chapitre dans lequel nous décrivons un prototype pour un outil de communication destiné à supporter une négociation 1:N sur l'Internet. Le vendeur négocie avec un ou plusieurs clients de façon indépendante et confidentielle : Chaque client a ainsi l'impression qu'il est le seul à négocier avec le vendeur. Dans ce prototype propose une forme structurée d'offre (ou de contre-offre) qui va permettre aux parties qui négocient de formaliser leur proposition de manière claire. Il offre également aux parties la possibilité de s'exprimer de manière non structurée lorsqu'elles le désirent, grâce à des messages sous forme de texte libre.

Ce prototype ne vise pas à mettre en place un mécanisme de négociation automatisée mais plutôt à offrir une structure formelle aux parties pour réaliser une négociation. Il tient également compte des exigences juridiques en matière de réalisation de contrat sur l'Internet.

Chapitre 5 : Prototype pour un support à la négociation 1:N sur l'Internet

5.1. Description du prototype

Le schéma de la Figure 5.1. décrit la conversation lors d'une négociation entre un vendeur et un acheteur.

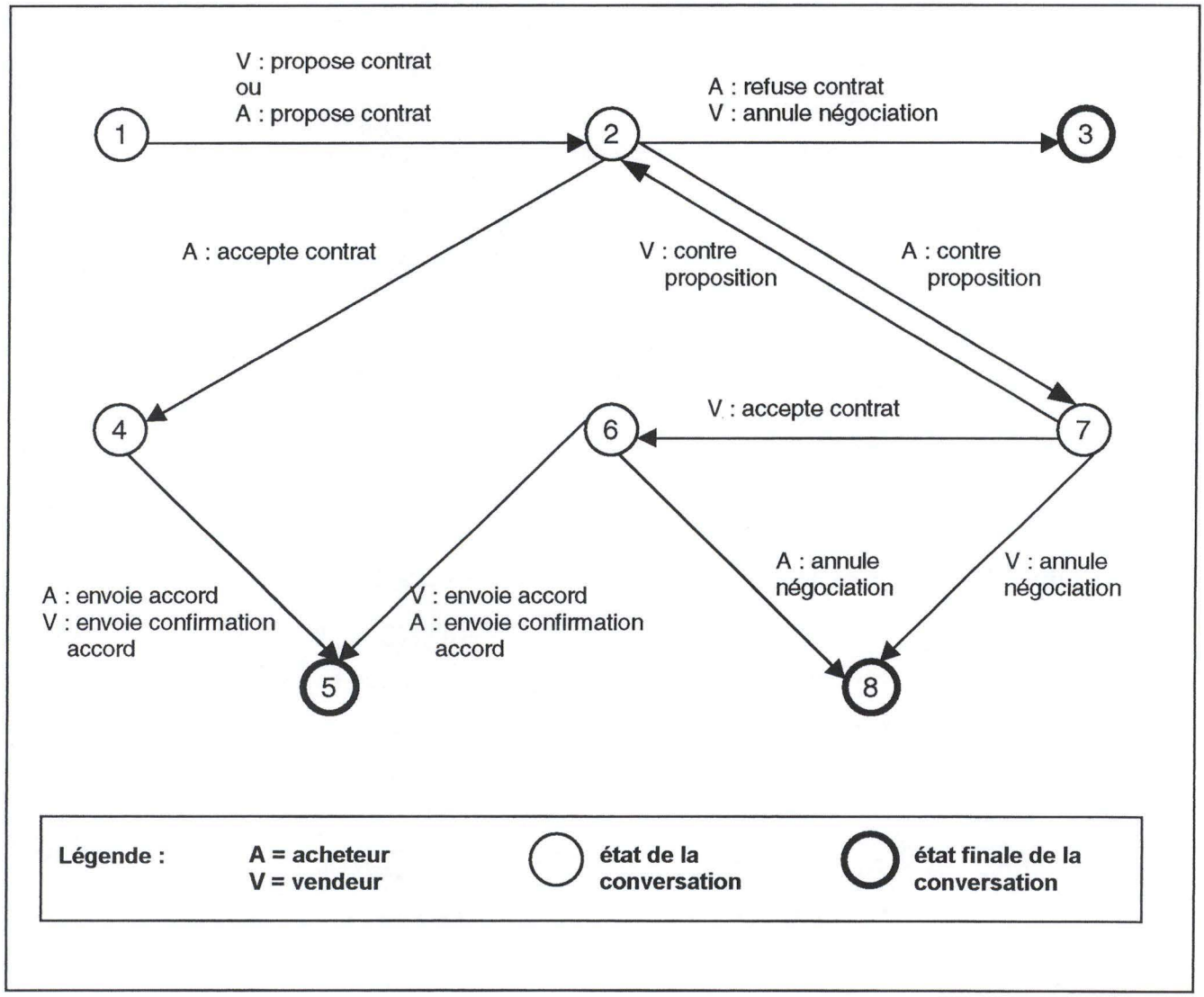


Figure 5.1. Conversation entre un acheteur et un vendeur

5.1.1. Définition du contexte

La négociation s'effectue sur l'Internet entre un vendeur (professionnel ou non) et un client. Le vendeur dépose son offre sur un serveur qui sert de plate-forme d'intermédiation. Le produit faisant l'objet de cette négociation ainsi que l'identité du vendeur sont décrits avec le plus de précisions possibles sur la page Web qui contient son offre.

Notre application a pour objectif de mettre à disposition des utilisateurs un support à la négociation sur l'Internet. Cette application vise à guider les parties dans leur négociation en s'appuyant sur une structure prédéfinie d'offre (ou de contre-offre). Elle offre aux parties d'autres outils tels que l'historique de la négociation et un système de messagerie. L'historique va permettre aux parties de pouvoir consulter à tout moment l'évolution de la négociation. Les parties peuvent également communiquer librement en dehors d'offres et des contre-offres en utilisant des messages de demande d'information. La négociation peut porter sur le prix, les conditions de retour, la date de livraison et toute autre condition que les parties peuvent vouloir ajouter au cours de l'évolution de la négociation.

Un vendeur peut négocier avec plusieurs clients en parallèle, mais de façon indépendante : aucun client n'est au courant de l'évolution de la négociation du vendeur avec d'autres clients. Le client a l'impression qu'il est le seul à négocier avec le vendeur.

Nous ne tiendrons pas compte ici de l'enregistrement du vendeur, nous supposons qu'il s'est enregistré au moment du dépôt de son offre sur le serveur : lorsqu'il veut commencer une session de négociation, il s'identifie en donnant son nom d'utilisateur et son mot de passe. Pour les acheteurs, nous devons faire une distinction entre les nouveaux et les anciens clients. Un nouveau client doit s'enregistrer avant de commencer une session de négociation. Un ancien client commencera une session de négociation en s'identifiant par son nom d'utilisateur et son mot de passe.

Le serveur joue le rôle d'intermédiaire de négociation entre l'acheteur et le vendeur. Les deux parties sont des clients du serveur utilisant toutes deux les services du serveur.

5.1.2. Structure des objets du dialogue

a) Structure d'une offre

Une offre est composée d'un contenu statique (dont le type est le même pendant toute l'évolution de la négociation) et d'un contenu dynamique (dont le contenu varie en fonction des choix du client et du vendeur) :

- ***contenu statique :*** le prix, le délai de livraison et les conditions de retour.
- ***contenu dynamique :*** ce sont les autres conditions que le client et le vendeur peuvent souhaiter ajouter, pour que la négociation soit ouverte. Les parties peuvent ajouter autant de conditions qu'elles voudront au cours de la négociation.

b) Structure d'une contre-offre

La structure d'une contre-offre est exactement la même que celle d'une offre, la différence se situe au niveau du contenu. Dans une offre, le contenu est entièrement nouveau c'est-à-dire qu'il s'agit d'une première proposition faite par l'une des parties. A l'inverse, dans une contre-offre, l'une des parties réagit à une proposition faite par l'autre partie. La partie qui fait la contre-offre peut modifier tout ou partie des conditions proposées par l'offrant.

Lorsqu'une partie fait une contre-offre, les attributs qui ont été modifiés sont mis en évidence pour que la partie qui reçoit la proposition visualise aisément les attributs qui ont été modifiés (en utilisant une couleur différente, par exemple).

c) Structure d'une demande d'information

La structure d'une offre présente le défaut de limiter la conversation des parties à un certain nombre d'attributs prédéfinis. Pour résoudre ce problème, nous offrons aux parties la possibilité de s'envoyer mutuellement des messages.

Une demande d'information est une conversation libre entre un vendeur et un client. La conversation peut porter ou non sur la négociation en cours. La demande d'information permettra aux parties de discuter de manière libre pour enrichir leur conversation, en se posant des questions

qui ne sont pas forcément des contre-propositions, mais qui ont pour objet, par exemple, la clarification de certains points de la négociation.

Dans notre prototype, la structure de la demande d'information prévoira une partie visant à rappeler le contexte actuel de la négociation sur lequel l'expéditeur voudrait avoir l'information et une autre partie qui contiendra le message qu'il voudrait transmettre à son partenaire.

d) Structure de l'accord sur une offre

L'accord sur une offre doit contenir l'offre sur lequel elle porte pour éviter toute ambiguïté. Ce volet contiendra également un message spécifiant l'acceptation de l'offre référencée. Ceci pour répondre également à l'obligation d'information qu'impose la loi en ce qui concerne les contrats réalisés sur l'Internet. La partie qui donne son accord doit savoir sur quoi il porte. Avant d'envoyer l'accord définitif manifesté par une partie, le système demande confirmation à la partie concernée.

e) Structure d'une confirmation d'un accord

C'est la réponse de la partie dont l'offre a été acceptée. Tout comme l'accord et pour les mêmes raisons, la confirmation d'une offre doit également contenir l'offre référencée. La confirmation d'une offre termine la négociation. Le système demandera confirmation avant d'enregistrer la fin de la négociation. Cette confirmation est importante car elle assure à la partie qui a envoyé son accord sur une offre que le contrat est conclu.

f) Structure de l'historique de la négociation

L'historique contiendra toutes les étapes de la négociation classées par ordre chronologique. Les informations contenues dans l'historique vont conserver leur forme originale c'est-à-dire la forme d'une demande d'information, d'une offre ou d'une contre-offre. Par exemple, les attributs d'une contre-offre qui ont été modifiés seront mis en évidence (par une couleur différente, par exemple). L'historique garde ainsi une trace de tout le déroulement de la négociation. En cas de litige, les parties peuvent y recourir pour vérification. Une copie de l'historique doit donc être envoyée aux parties, par e-mail, à la fin d'une négociation.

g) Structure de l'annulation d'une négociation

L'annulation d'une négociation consiste en la suppression du client concerné par cette négociation dans la liste des clients. Ce qui met fin à la négociation entre le client concerné et le vendeur. Le partenaire de la partie qui annule la négociation sera automatiquement averti par le serveur qui l'informera de la fin de la négociation pour cause d'abandon.

5.2. Spécifications des dialogues

Nous avons sept principaux types de dialogues dans notre prototype de négociation : le dialogue de l'offre, le dialogue de la contre-offre, le dialogue de la demande d'information, le dialogue de l'accord d'une offre, le dialogue de la confirmation de l'offre, le dialogue de l'abandon de la négociation et le dialogue de la consultation de l'historique.

5.2.1. Le dialogue de l'offre

Nous avons deux sortes de dialogues d'offre selon qu'il s'agit d'une offre faite par le client ou d'une offre faite par le vendeur :

- Si l'offre est faite par le client :

  - ***Phase d'identification :***
    - Si le client est nouveau, il reçoit un formulaire d'identification qu'il remplit et renvoie. Ce formulaire comprend les cases suivantes : nom d'utilisateur, mot de passe, nom et prénom du client, adresse postale du client, adresse e-mail du client et choix du mode de payement (par chèque, carte de crédit ou virement). Si le login choisi est déjà utilisé par un ancien client, l'utilisateur reçoit un message pour lui demander de changer le login. Si le mot de passe choisi n'est pas valable, il reçoit un message pour lui dire que son mot de passe n'est pas valide. Si le login et mot de passe sont acceptés, le client est enregistré.

    - Si c'est un ancien client, il introduit son login et son mot de passe pour s'identifier.

- ***Phase d'ouverture de session :*** la validation de l'identification du client crée une session et il reçoit un message l'invitant à choisir une action (un dialogue).

- ***Phase du choix de l'action :*** le client choisit l'action "faire une offre" à la suite de laquelle il reçoit le formulaire de l'offre.

- ***Phase d'entrée de l'offre :*** le client doit remplir le formulaire de l'offre et le renvoyer. Il reçoit ensuite un message lui demandant de confirmer les informations introduites dans cette offre. S'il répond par l'affirmative, son offre est enregistrée et la session se termine. Sinon, il reçoit le formulaire de l'offre avec les informations introduites pour y opérer une vérification ou une éventuelle modification des informations avant d'envoyer de nouveau le formulaire. Il peut également annuler son action et terminer la session ou choisir une autre action et continuer la session.

- Si l'offre est faite par le vendeur :

  - ***Phase d'identification :*** il introduit son nom d'utilisateur ainsi que son mot de passe. Si l'un des deux n'est pas correct, il reçoit un message lui demandant de recommencer l'identification car le mot de passe ou le login est incorrect.

  - ***Phase d'ouverture de session :*** la validation de l'identification du vendeur crée une session pour le vendeur et il reçoit un message l'invitant à choisir une action.

  - ***Phase du choix de l'action :*** le vendeur choisit l'action "faire une offre" à la suite de laquelle il reçoit le formulaire de l'offre.

  - ***Phase de l'entrée de l'offre :*** le vendeur doit remplir le formulaire de l'offre et le renvoyer. Il reçoit ensuite un message lui demandant de confirmer les informations introduites dans cette offre. S'il répond par l'affirmative, son offre est enregistrée et la session se termine. Sinon, il reçoit de nouveau le formulaire de l'offre avec les informations introduites pour y opérer une vérification ou une éventuelle modification des informations avant d'envoyer

de nouveau le formulaire. Il peut également annuler son action et terminer la session ou choisir une autre action et continuer la session.

5.2.2. Le dialogue de la contre-offre

Le dialogue est le même qu'il s'agisse du vendeur ou de l'acheteur. L'utilisateur ne passe par les phases d'identification et d'ouverture de session que s'il n'avait pas encore ouvert une session pour réaliser un autre dialogue :

- ***Phase d'identification :*** l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

- ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit et reçoit un message l'invitant à choisir une action;

- ***Phase du choix de l'action :*** l'utilisateur choisit l'action "faire une contre-offre" à la suite de laquelle il reçoit un message qui lui demande s'il veut consulter l'historique de la négociation avant d'introduire sa proposition

- ***Phase d'entrée de la contre-offre :*** l'utilisateur peut répondre par l'affirmative ou par la négative au message reçu,

    - S'il répond par l'affirmative, il reçoit l'historique qu'il consulte. Après quoi, il reçoit l'offre de la partie adverse, introduit les modifications éventuelles et renvoie sa proposition. Il reçoit ensuite un message lui demandant de confirmer les informations introduites. S'il répond "oui", alors sa proposition est envoyée à son partenaire et la session est fermée. Sinon, le formulaire de contre-offre lui est renvoyé, il vérifie les informations qu'il contient et les modifie éventuellement avant de l'envoyer et terminer la session. Il peut aussi annuler son action et revenir à dernière étape de la négociation pour choisir une autre action.

- S'il ne veut pas consulter l'historique avant d'introduire son offre, il reçoit l'offre de la partie adverse qu'il modifie et renvoie. Il reçoit ensuite un message qui lui demande de confirmer les informations introduites. S'il répond oui, alors sa proposition est envoyée à son partenaire et la session est fermée. Sinon, il revient sur le formulaire de contre-offre, vérifie les informations qu'il contient et les modifie éventuellement avant de l'envoyer et terminer la session. Il peut aussi annuler son action et choisir une autre action pour continuer.

5.2.3. Le dialogue de demande d'information

Le dialogue est le même pour l'acheteur et le vendeur. L'utilisateur peut soit demander l'information pendant une session, soit commencer une session pour demander une information.

- Si l'utilisateur a déjà ouvert une session, alors il passe directement à la phase d'introduction de la demande d'information décrite plus bas;

- Si l'utilisateur n'avait pas encore ouvert une session, alors :

  - ***Phase d'identification*** : l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

  - ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit un message l'invitant à choisir une action;

  - ***Phase du choix de l'action :*** l'utilisateur choisit l'action "demande d'information" à la suite de laquelle il reçoit le formulaire de demande d'information ;

  - ***Phase d'introduction de la demande :*** l'utilisateur remplit le formulaire de demande d'information. Ce dernier comprend une partie pour rappeler le contexte actuel de la négociation (pas le contexte historique), et une partie pour écrire sa demande d'information proprement dite. Il peut envoyer son

message puis clôturer la session ou envoyer son message et continuer la session par un autre dialogue.

5.2.4. Le dialogue de l'acceptation de l'offre

Le dialogue est le même pour l'acheteur et pour le vendeur. L'utilisateur ne passe par les phases d'identification et d'ouverture de session que s'il n'avait pas encore ouvert une session pour réaliser un autre dialogue :

- ***Phase d'identification :*** l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

- ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit le dernier état de la négociation;

- ***Phase du choix de l'action :*** l'utilisateur choisit l'action "accepter offre" à la suite de laquelle il reçoit un formulaire pour l'accord de la dernière offre de son partenaire;

- ***Phase de l'entrée de l'accord :*** l'utilisateur introduit le message d'acceptation de l'offre et renvoie l'accord;

- ***Phase de confirmation de l'action :*** l'utilisateur reçoit un message qui lui demande de confirmer son action, s'il répond "oui", son message est envoyé et la session se termine. Sinon, il reçoit de nouveau le formulaire de l'accord pour une vérification et son envoie ou pour annuler son action d'acceptation et terminer la session.

5.2.5. Le dialogue de la confirmation de l'offre

Le dialogue est le même pour l'acheteur et le vendeur. L'utilisateur ne passe par les phases d'identification et d'ouverture de session que s'il n'avait pas encore ouvert une session pour réaliser un autre dialogue :

- ***Phase d'identification :*** l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

- ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit un message l'invitant à choisir une action;

- ***Phase du choix de l'action :*** l'utilisateur choisi l'action "confirmation de l'offre" à la suite de laquelle il reçoit un formulaire pour la confirmation de son offre.

- ***Phase de l'entrée de la confirmation :*** l'utilisateur introduit le message de confirmation de son offre et le renvoie;

- ***Phase de confirmation de l'action :*** l'utilisateur reçoit un message qui lui demande de confirmer son action. S'il répond "oui", son message de confirmation est envoyé et la session se termine. Sinon, il reçoit de nouveau le formulaire de confirmation pour une vérification et son envoie ou pour annuler son action et terminer la session.

5.2.6. Le dialogue de l'abandon de la négociation

Le dialogue est le même pour l'acheteur et pour le vendeur. L'utilisateur ne passe par les phases d'identification et d'ouverture de session que s'il n'avait pas encore ouvert une session pour réaliser un autre dialogue :

- ***Phase d'identification :*** l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

- ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit un message l'invitant à choisir une action;

- ***Phase du choix de l'action :*** l'utilisateur choisi l'action "annulation de la négociation" à la suite de laquelle il reçoit un message qui lui demande de confirmer son action.;

- ***Phase de confirmation de l'action :*** l'utilisateur répond soit par l'affirmative au message qu'il reçoit et la session se termine, soit il répond par la négative et il reçoit de nouveau le dernier état de la négociation pour choisir une autre action.

5.2.7. Le dialogue de la consultation de l'historique

Le dialogue est le même pour l'acheteur et le vendeur. L'utilisateur peut soit consulter l'historique pendant une session, soit commencer une session pour consulter l'historique.

- Si l'utilisateur avait déjà ouvert une session, alors il passe directement à la phase du choix de l'action décrite plus bas;

- Si l'utilisateur n'avait pas encore ouvert une session, alors :

  - ***Phase d'identification*** : l'utilisateur introduit son nom d'utilisateur ainsi que son mot de passe, si l'un des deux n'est pas correct, il reçoit un message qui lui demande de recommencer l'identification car le mot de passe ou le login est incorrect;

  - ***Phase d'ouverture de session :*** si l'identité de l'utilisateur est validée, alors la session est ouverte et il reçoit un message l'invitant à choisir une action;

  - ***Phase du choix de l'action :*** l'utilisateur choisit l'action "consulter l'historique de la négociation" à la suite de laquelle il reçoit l'historique de la négociation;

  - ***Phase de consultation de l'historique :*** l'utilisateur consulte l'historique de la négociation. Lorsqu'il termine la consultation, il peut soit continuer la session en choisissant une autre action, soit terminer la session.

5.3. Architecture logicielle du prototype

La négociation étant une tâche très interactive, nous nous sommes inspiré de la méthodologie Trident pour réaliser l'architecture logicielle de notre prototype.

5.3.1. Analyse de la tâche

La tâche interactive de notre prototype est de réaliser une négociation. Le but principal de cette tâche est décomposé en huit sous-buts auxquels on peut associer huit sous-tâches : identifier l'utilisateur, faire une offre, faire une contre-offre, faire une demande d'information, consulter l'historique de la négociation, accepter une offre, confirmer l'acceptation d'une offre et préciser l'abandon d'une négociation.

a) Diagramme des buts et sous-buts

Le graphique de la Figure 5.2. représente la décomposition de la tâche en buts et sous-buts.

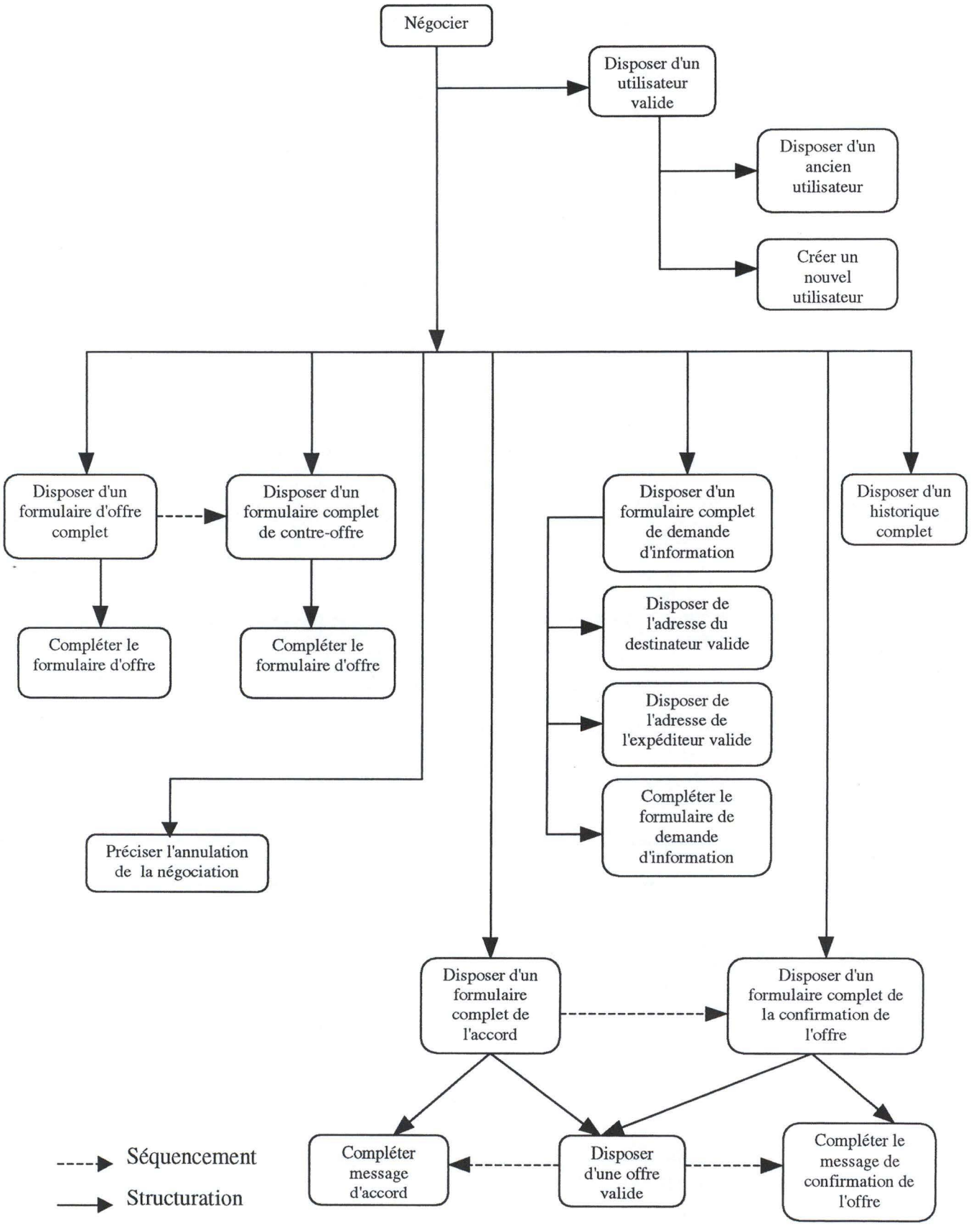


Figure 5.2. : Diagramme des buts et sous-buts

b) Décomposition en actions

- ***Décomposition en actions de la première sous-tâche :***

client_enreg : = faux;

type_client : = demande (client);

SI type_client = nouveau

ALORS

nom : = demande (client);

prénom : = demande (client);

nom_utilisateur : = demande (client);

mot_de_passe : = demande (client);

confirmer_mot_passe : = demande (client);

adresse : = demande (client);

adresse_e-mail : = demande (client);

mode_paiement : = demande (client);

SI nom_utilisateur_valide (nom_utilisateur, →utilisateurs_bd, ← login_valide)
= ok

ALORS

SI valider_mot_passe (→mot_de_passe, →confirmer_mot_passe,
←mot_passe_valide) = ok

ALORS

enregistrer_utilisateur (→nom, → prénom, →nom_utilisateur,
→mot_de_passe, →adresse, →adresse_e-mail, →mode_paiement);
client_enreg : = vrai;

FINSI

FINSI

FINSI

SI type_client = ancien

ALORS

nom_utilisateur : = demande (client);

mot_de_passe : = demande (client);

SI verifier_mot_passe (→mot_passe, →nom_utilisateur, →bd_client,
←mot_passe_correct) = ok

ALORS client_enreg : = vrai

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type_dialogue = offre

ALORS

prix : = demande (client);

delai_livraison : = demande (client);

conditions_retour : = demande (client);

mode_apiement : = demande (client);

autres_conditions : = demande (client);

enregistrer_offre (→prix, →delai_livraison, →conditions_retour, →mode_paiement, →autres concditions);

offre_enregistre : = vrai

FINSI

- ***Décomposition en actions de la troisième sous-tâche :***

type_dialogue : = demande (client);

SI type_dialogue = contre-offre

ALORS

SI offre_enregistre : = vrai;

ALORS

prix : = demande (client);

delai_livraison : = demande (client);

conditions_retour : = demande (client);

mode_apiement : = demande (client);

autres_conditions : = demande (client);

SI modif_offre (→prix, →delai_livraison, →conditions_retour, →mode_paiement, →autres concditions, →bd_offre, ←offre_modifie) = ok

ALORS

enreg_contre-offre (→prix, →delai_livraison, conditions_retour, →mode_paiement, →autres concditions);

offre_enregistre : = vrai

FINSI

FINSI

FINSI

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type_dialogue = message

ALORS

rappel_contexte : = demande (client);

contenu_message : = demande (client);

enregistrer_message (→rappel_contexte, →contenu_message);

FINSI

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type de dialogue = accord;

ALORS

SI offre_enregistre = vrai

ALORS

message_accord : = demande (client);

enregistrer_accord (→message_accord, ←accord_enreg)

FINSI

FINSI

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type de dialogue = confirmation_accord

ALORS

SI enregistrer_accord (→message_accord, ←accord_enreg) = ok

ALORS

message_conf_accord : = demande(client);

enregistrer_conf (→message_conf_accord, ← conf_accord)

FINSI

FINSI

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type de dialogue = abandon_nego **ALORS enregistrer_abandon** ()

- ***Décomposition en actions de la deuxième sous-tâche :***

type_dialogue : = demande (client);

SI type de dialogue = consult_historique **ALORS consulter_hist** (←historique)

c) Identification des objets de la tâche

Les objets manipulés par les actions ci-dessus sont les suivants : un objet UTILISATEUR qui est décomposé en deux sous-types (ACHETEUR et VENDEUR), un objet OFFRE, un objet ACCORD, un objet ABANDON, un objet CONFIRMATION, un objet HISTORIQUE et un objet MESSAGE. Ces objets sont spécifiés sous forme d'un schéma entité-association (voir annexe).

d) Fonctions sémantiques

- **enregistrer_offre** (→prix, →delai_livraison,, →conditions_retour, →mode_paiement, →autres_conditions)

- **valide_modif_offre** (→prix, →delai_livraison, →conditions_retour, →mode_paiement, →autres_conditions)

- **valider_offre** (→prix, →delai_livraison,, →conditions_retour, →mode_paiement, →autres_conditions, ← offre_valide)

- **verifier_mot_passe** (→mot_passe, →login, ←mot_passe_correct)

- **verifier_login** (→mot_passe, →login, ←login_correct)

- **valider_login** (→mot_passe, →login, ←login_valide)

- **valider_données** (→nom, → prénom, →nom_utilisateur, →mot_passe, →adresse, →adresse_e-mail, →mode_paiement)

- **enregistrer_utilisateur** (→nom, → prénom, →nom_utilisateur, →mot_passe, →adresse, →adresse_e-mail, →mode_paiement)

- **enregistrer_accord** (→offre, →message_acc)

- **enregistrer_conf** (→accord, →message_conf)

- **consulter_historique** (←historique)

- **enregistrer_dde_information** (→adr_mail_exp, →adr_mail_dest, →date, →heure, →objet, →contenu)

- **enregistrer_abandon** ( )

- **valider_mot_passe** (→mot_passe, ← mot_passe_valide)

- **valider_adr_dest** (→adr_dest, ← adr_dest_valide)

- **valider_adr_exp** (→ adr_exp, ← adr_exp_valide)

e) Graphe d'enchaînement des fonctions

La figure 5.3. représente l'enchaînement des fonctions sémantiques.

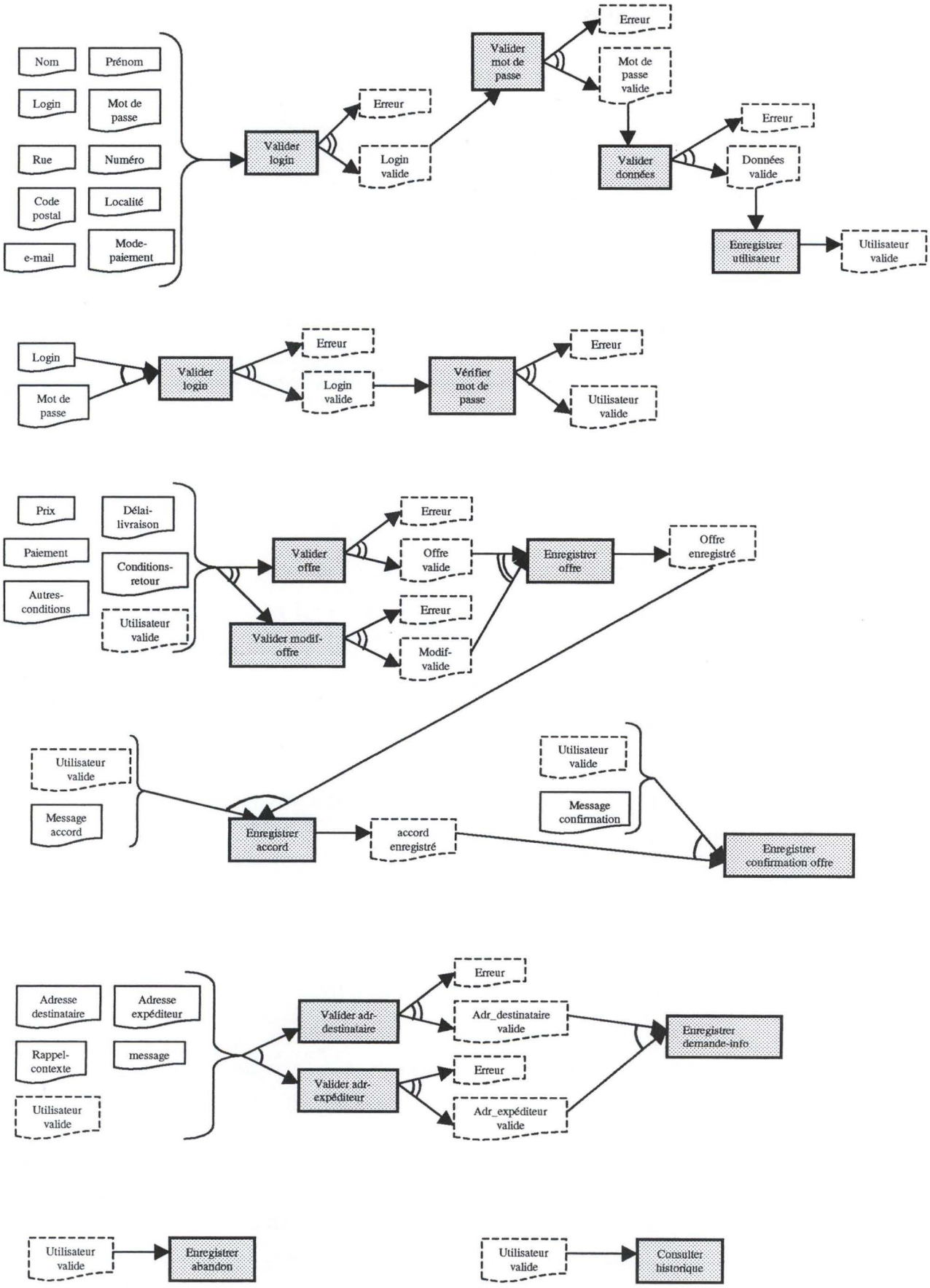
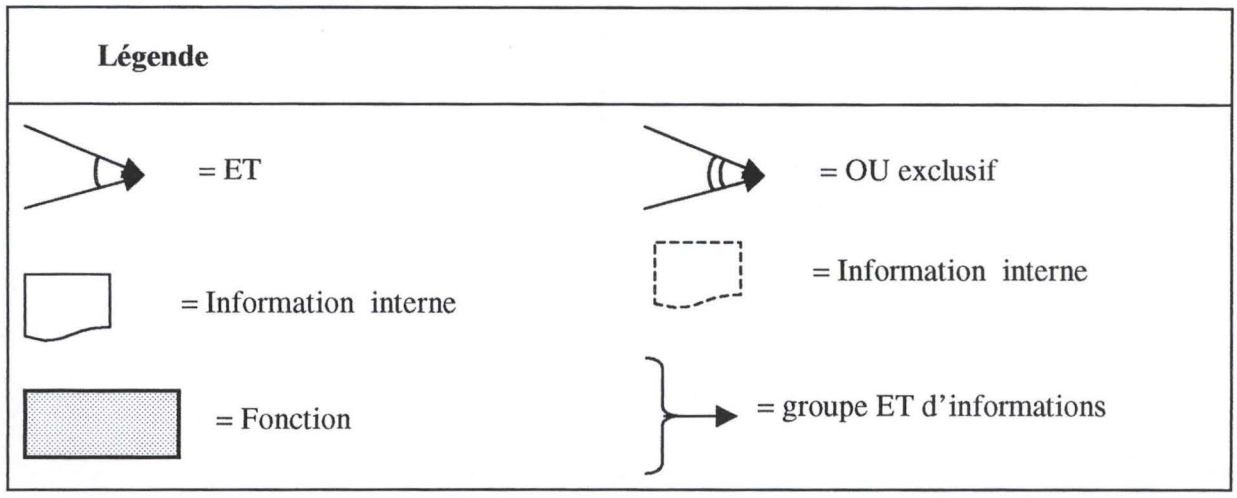


Figure 5.3. : Graphe d'enchaînement des fonctions



f) Choix des attributs de dialogue

Le contrôle du dialogue est externe pour la tâche car l'initiative vient de l'utilisateur. Le mode de dialogue est asynchrone au sein de chaque sous-tâche car l'ordre d'exécution des actions n'est pas prédéterminé. Toutefois, il est synchrone entre les sous-tâches d'identification de l'utilisateur et les autres sous-tâches car l'utilisateur doit d'abord s'identifier avant de réaliser n'importe quelle autre sous-tâche. Il est également synchrone entre :

- la sous-tâche "faire une offre" et la sous-tâche "faire une contre-offre", car on doit avoir fait une offre pour faire une contre-offre;
- la sous-tâche "faire une offre" ou "faire une contre-offre" et la tâche "accepter une offre", car on doit avoir accepté une offre avant d'en accepter une;
- la sous-tâche "accepter une contre offre" et la sous-tâche "confirmer l'accord d'une offre", car on ne peut confirmer qu'un accord qui a été fait.

Le mode de déclenchement des fonctions est manuel explicite affiché car les fonctions doivent être déclenchées à l'initiative de l'utilisateur. La métaphore est celle du mini-monde car l'utilisateur agit en tant qu'acteur.

Ces différents attributs nous permettent de dériver les styles d'interaction : le remplissage de forme, la manipulation directe, l'interaction iconique et le multi-fenêtrage pour les tâches indépendantes.

g) Identification des unités de présentation (UP)

Une unité de présentation est définie comme l'ensemble de fenêtres logiques nécessaires à l'exécution d'une sous tâche de premier niveau. Une fenêtre logique se définit comme l'ensemble des objets interactifs nécessaires à l'exécution d'une sous-tâche de dernier niveau, celui des actions. Nous dégageons du graphe d'enchaînement présenté à la figure 5.3. sept unités de présentation, dont une unité de présentation pour chaque sous-tâche sauf pour les sous-tâches "faire une offre" et "faire une contre-offre" que nous avons regroupées en une seule unité de présentation (Figure 5.4.).

h) Identification des fenêtres (FL)

Nous avons opté pour une identification fonctionnelle pour découper les unités de présentation en fenêtres logiques, c'est-à-dire que nous avons regroupé au sein d'une seule et même fenêtre les informations externes de chaque fonction. Pour l'UP2 on regroupe les fonctions "valider-offre" et "valider-modif-offre" pour respecter la règle ergonomique selon laquelle les informations (objets interactifs abstraits) logiquement liées doivent être groupés. La découpe des unités de présentation en fenêtres logiques est représentée par la figure 5.5.

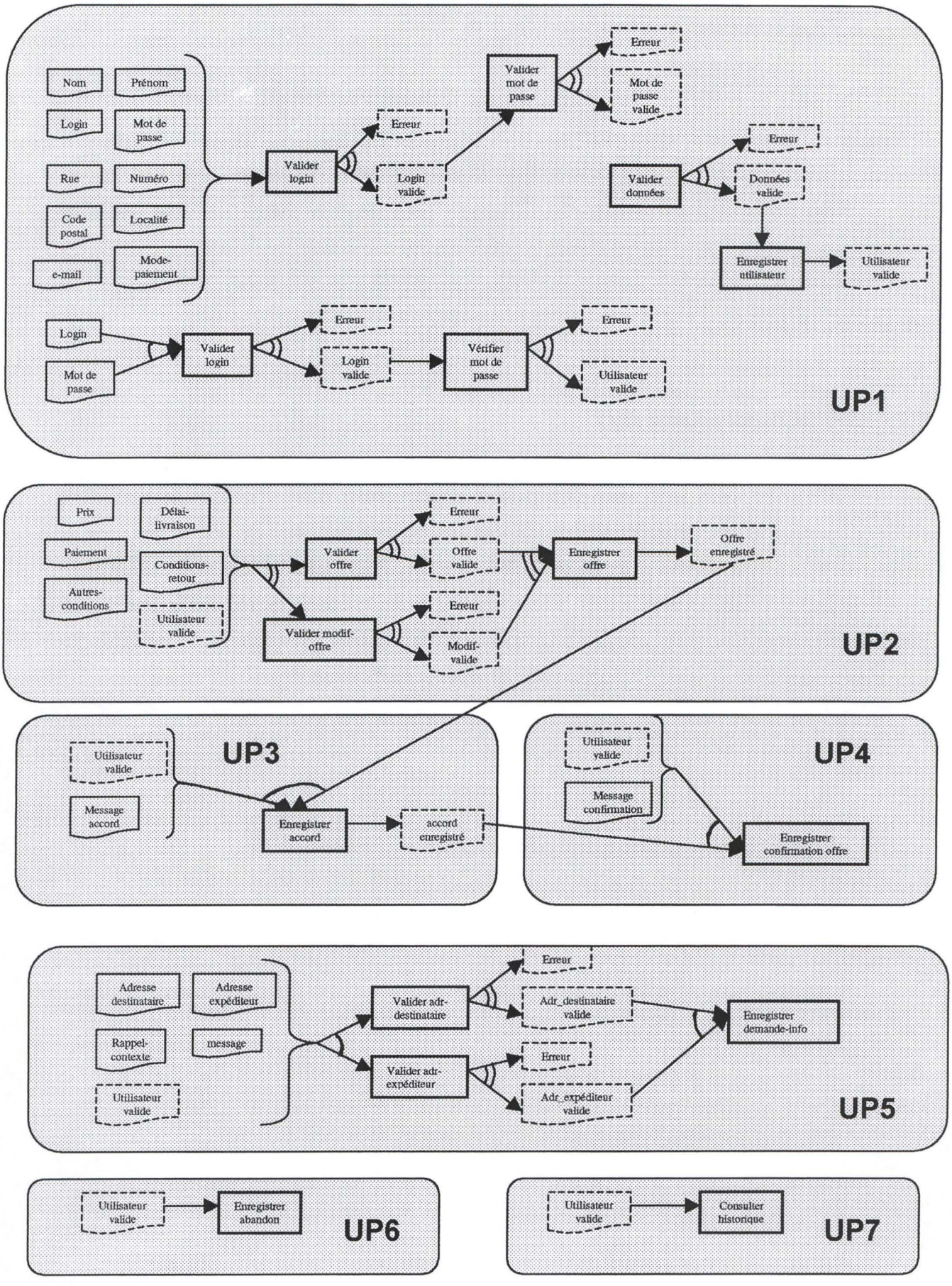


Figure 5.4. : Représentation des unités de présentation (UP)

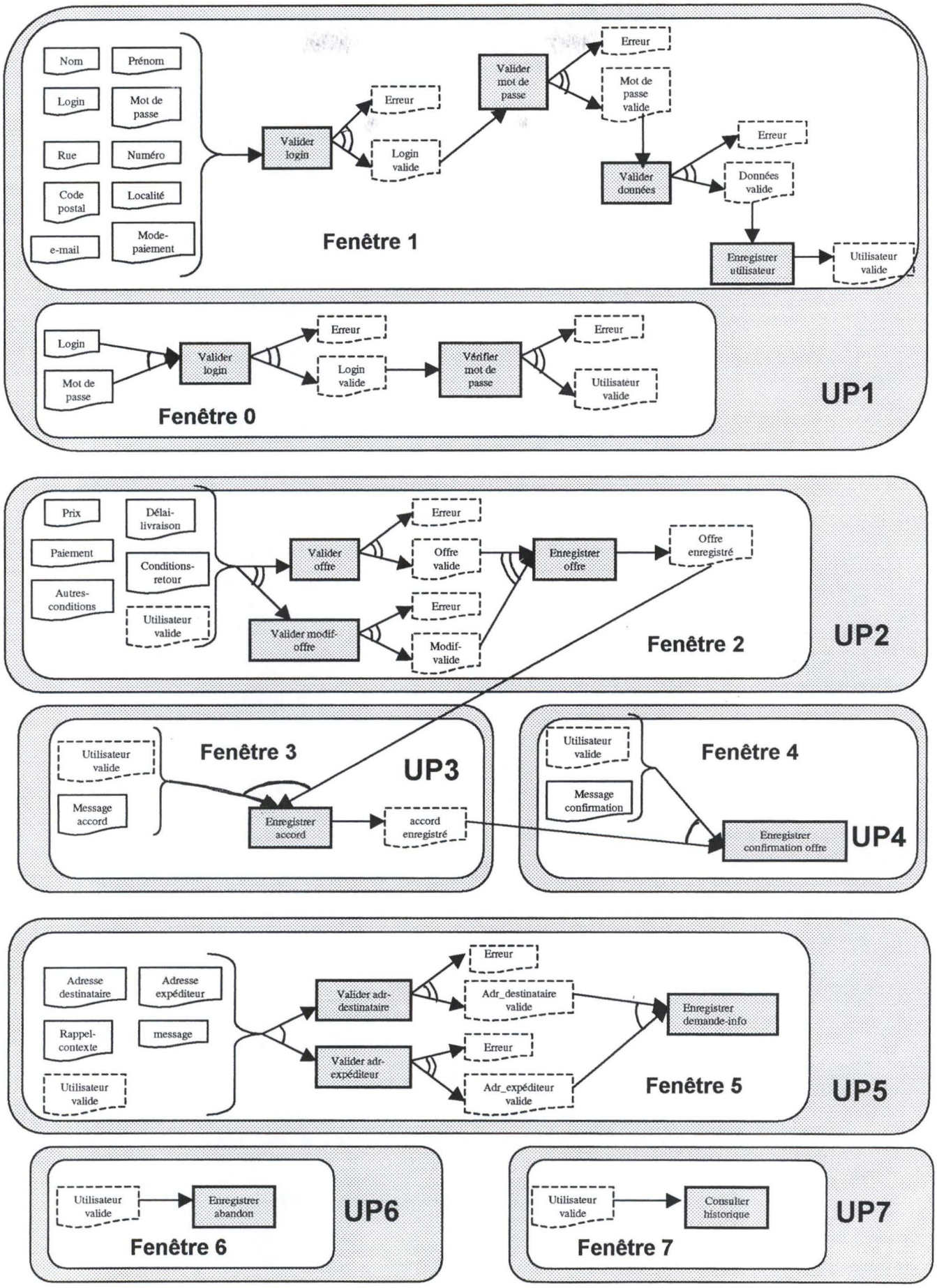


Figure 5.5. : Découpe des UP en fenêtres logiques

i) Structuration des objets du dialogue

Les objets du dialogue sont structurés en trois types de composants suivant la fonction qu'ils assument : les objets de l'application, les objets de contrôle et les objets interactifs.

Hiérarchie des objets d'application (OA)

Les objets d'application réalisent la machine fonctionnelle. Ils pilotent l'exploitation des services de l'application.

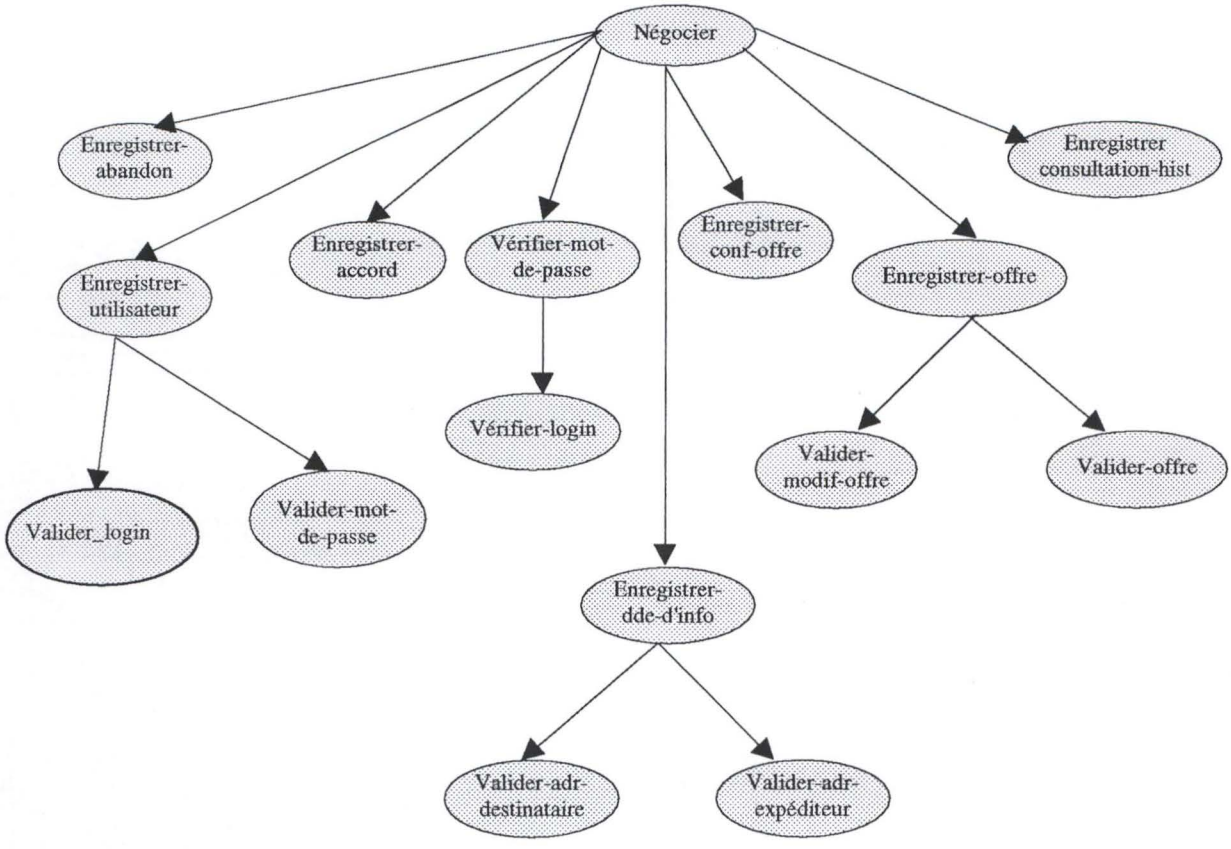


Figure 5.6. : Hiérarchie des objets d'application

Hiérarchie des objets de contrôle (OC)

Les objets de contrôle assurent la conversation entre les données strictement syntaxiques de la présentation et des données purement sémantiques de l'application. Ils pilotent l'exécution en cordonnant les échanges entre la présentation et les services de l'application. L'objet de contrôle tâche interactive (OC_TI) initialise le dialogue de la tâche interactive et ordonnance l'exécution des UP sur base éventuelle d'une sélection par l'utilisateur. Les objets interactifs unités de présentation (OC_UP) ordonnancent l'exécution des fenêtres logiques sur base éventuelle d'une sélection par l'utilisateur. Les objets de contrôle fenêtres logiques (OC_FL) gèrent le dialogue d'exécution de la sous-tâche associée à une fenêtre logique.

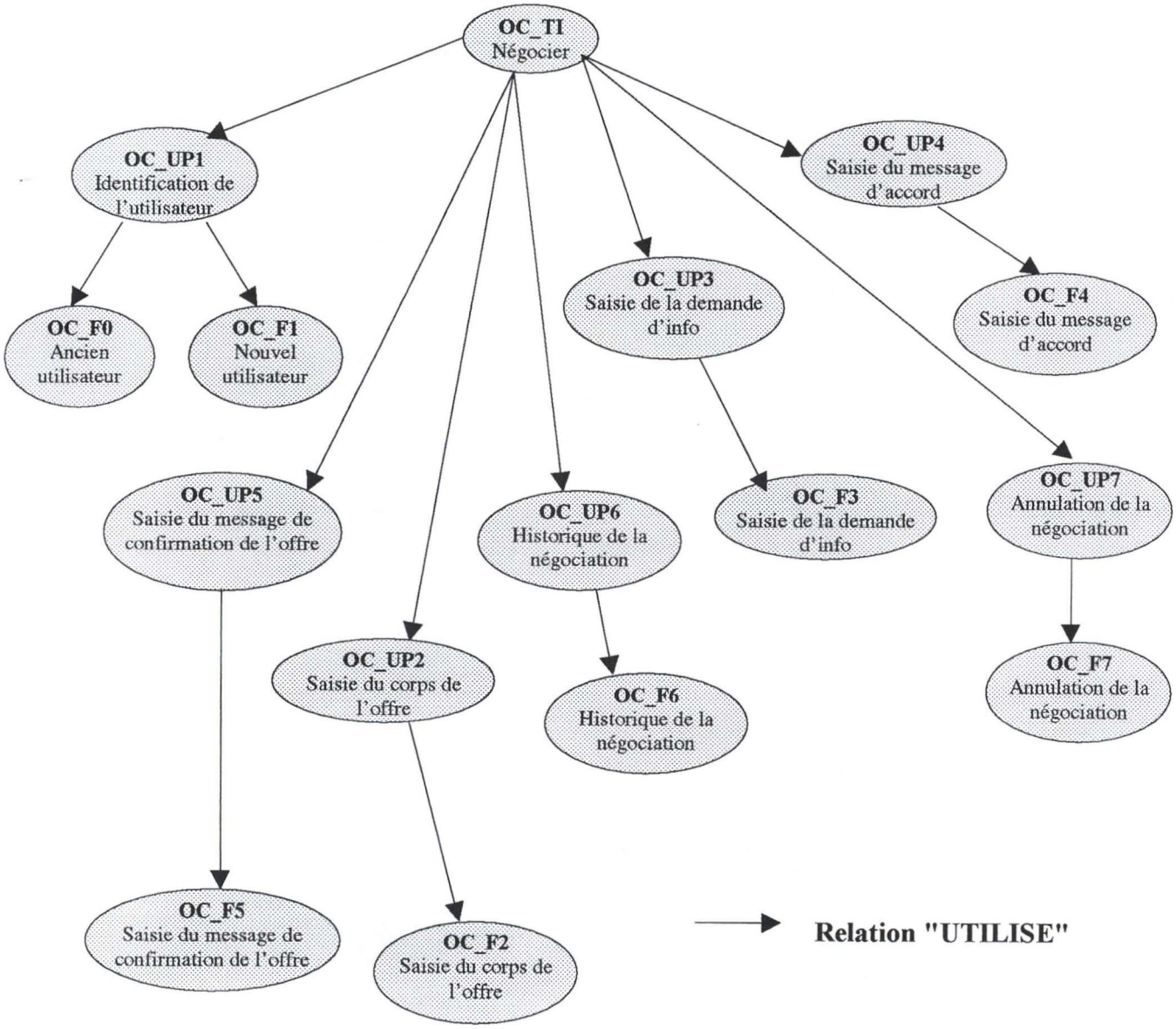
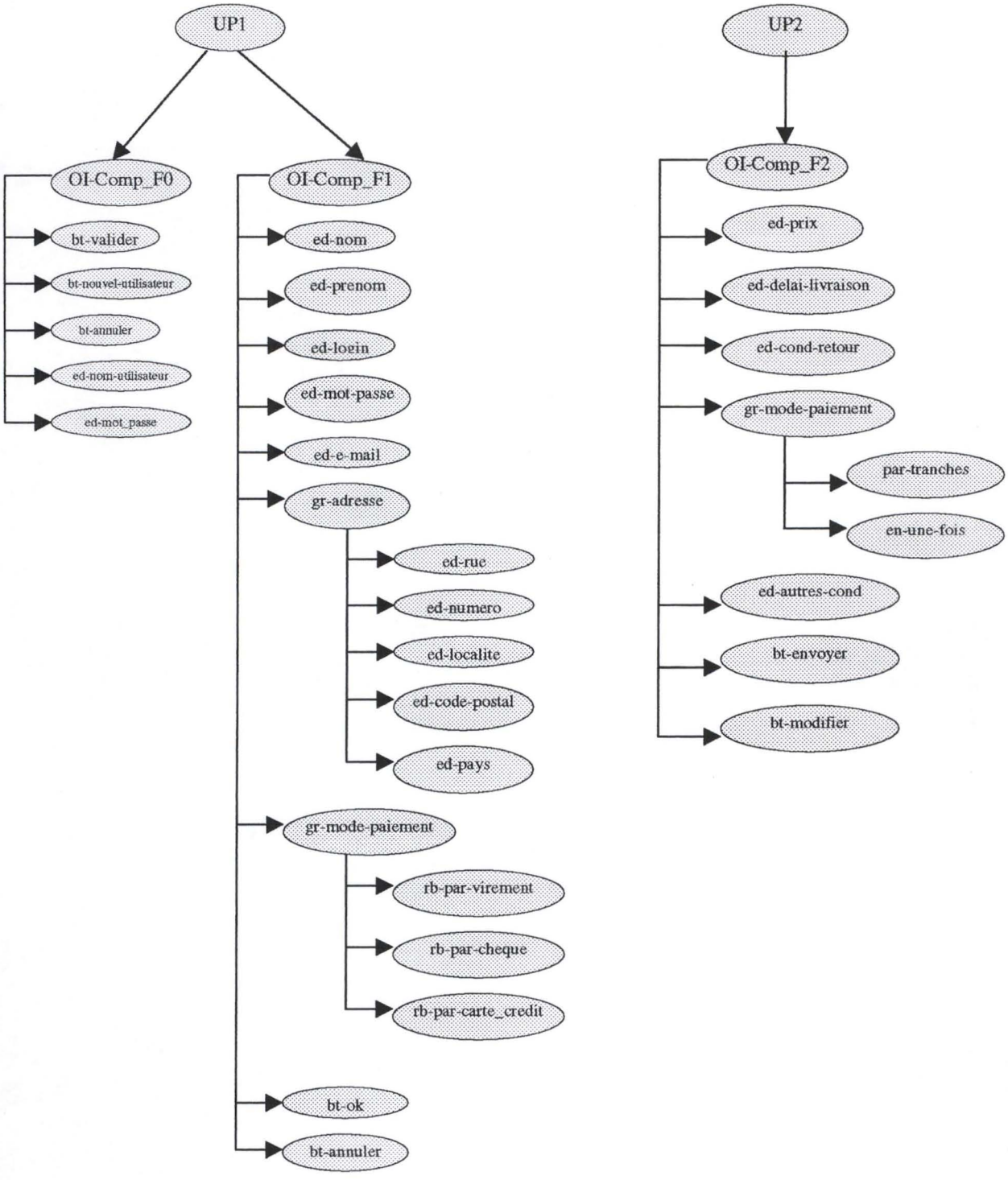


Figure 5.7. : Hiérarchie des objets de contrôle

Hiérarchie des objets Interactifs (OI)

Les objets interactifs assurent la présentation de l'interface de saisie et d'affichage. Ils réalisent le dialogue entre l'utilisateur et le système d'information associé à l'exécution de la tâche proprement dite.



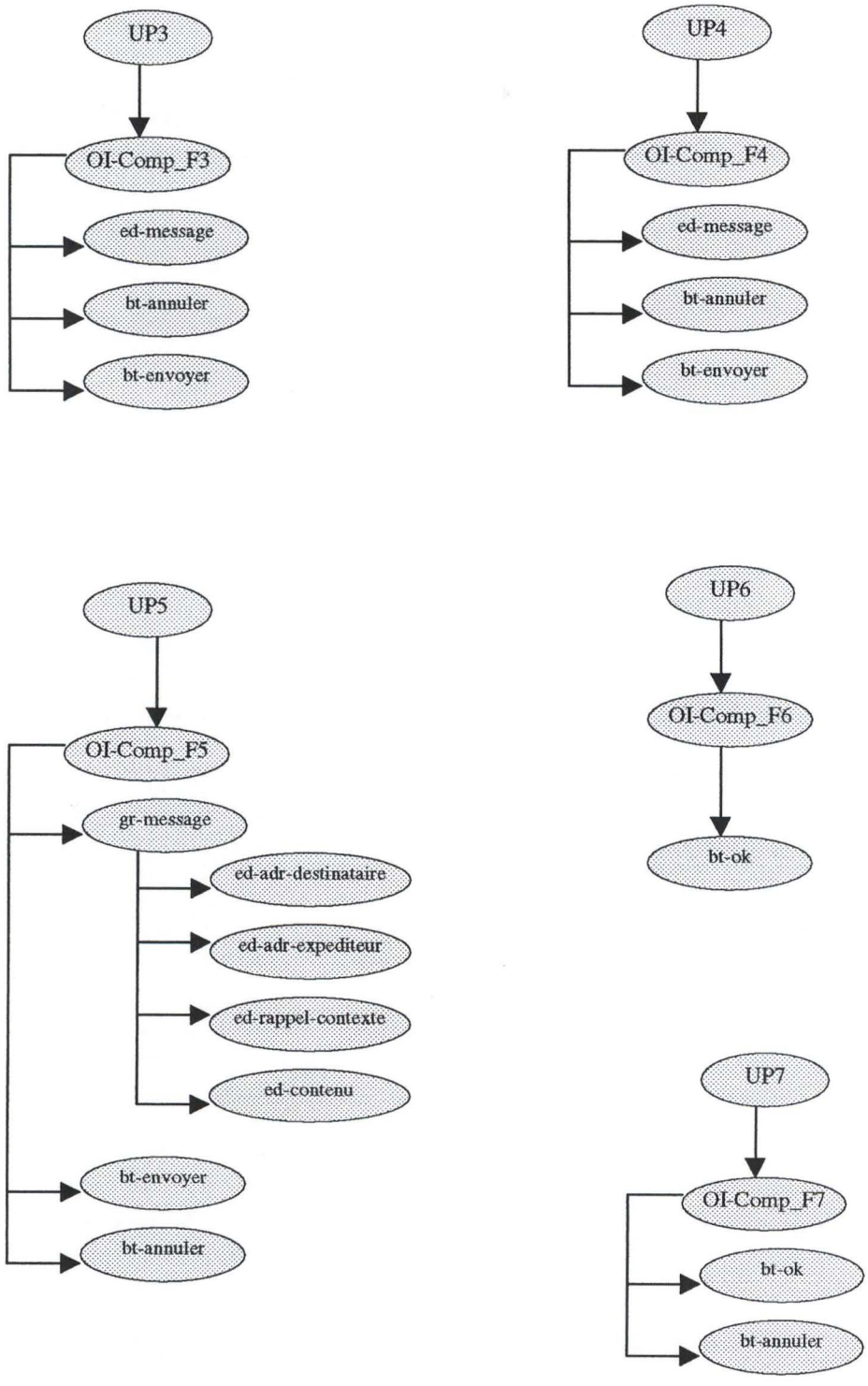


Figure 5.8. : Représentation de la hiérarchie des objets interactifs

Intégration des trois hiérarchies

Les objets de conversation (OC) utilisent les objets de l'interface (OI) ainsi que les objets d'application. L'OC_TI n'utilise pas d'OA, les OC_UP n'utilisent des OA que s'ils sont confondus avec les OC_FL. Chaque OC_FL utilise les OA qui encapsulent les fonctions capturées par l'OC. Pour assurer l'indépendance entre la partie fonctionnelle et la partie présentation, les OI n'utilisent pas les OA (Figure 5.9.).

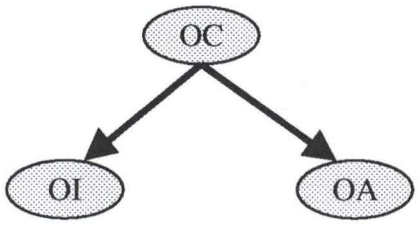


Figure 5.9. : Relation "utilise" entre les trois classes

Pour compléter cette architecture nous ajoutons la classe des données qui est utilisée par la classe des OA (Figure 5.10.).

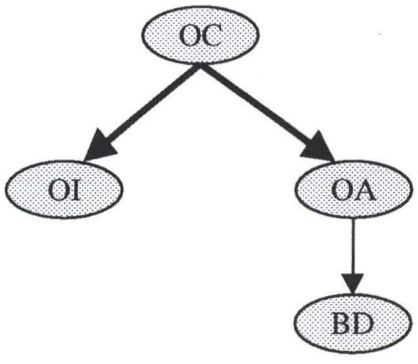


Figure 5.10. : Relation "utilise" entre les OA et les objets des données

La figure 5.11. illustre une intégration partielle des trois hiérarchies. Nous ne pouvons pas représenter le schéma complet de l'intégration sur une page car nous avons plusieurs objets qui interviennent dans les différentes classes.

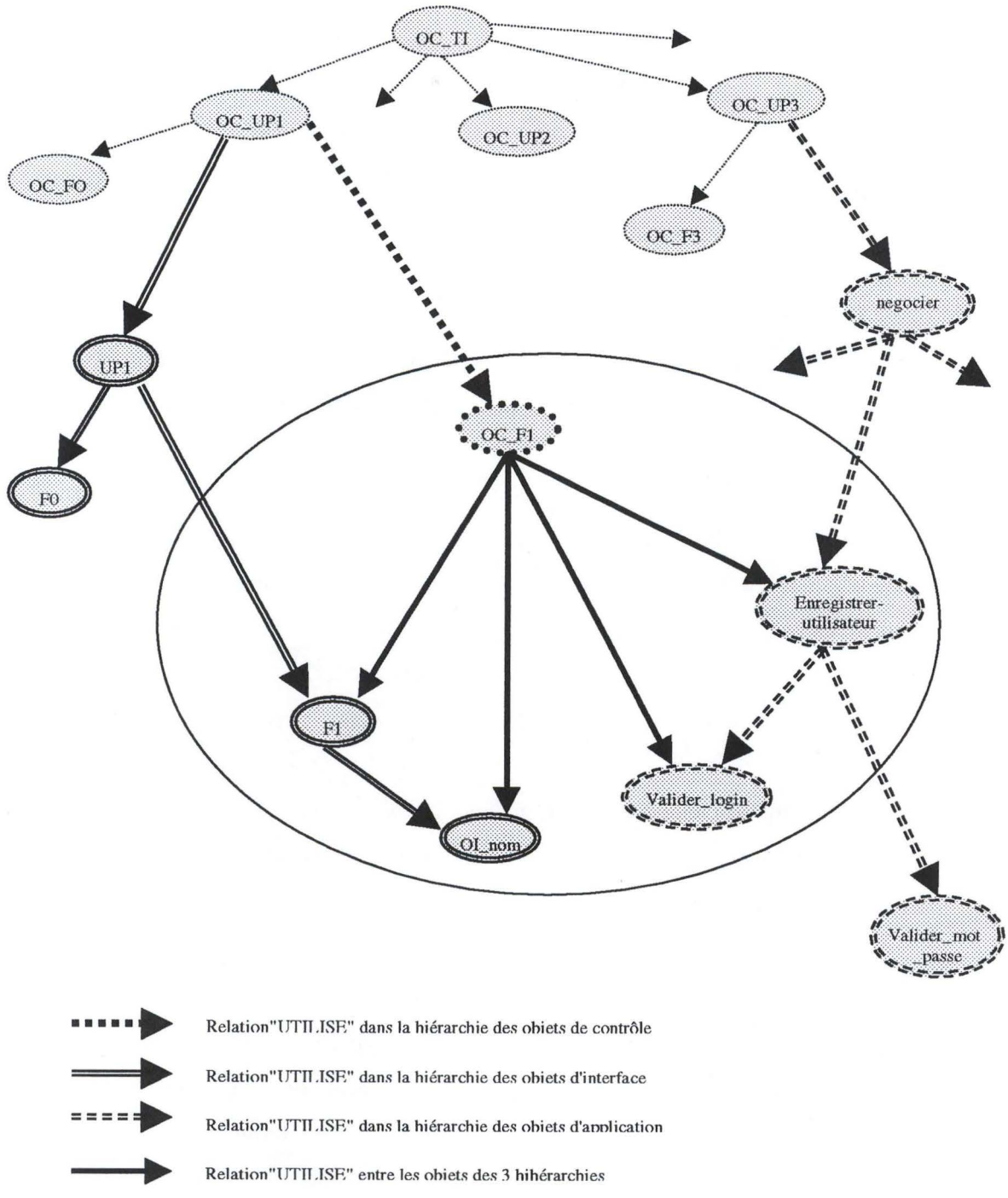


Figure 5.11. : Schéma d'intégration des trois hiérarchies

5.4. Intérêt du prototype

Ce prototype est plus intéressant qu'un e-mail traditionnel car il offre une structure prédéfinie de négociation pour guider les parties dans leur négociation et les aider ainsi à respecter les obligations contractuelles.

Ce prototype est semblable aux systèmes de négociation INSS et CBSS que nous avons décrits au chapitre 2 (2.5.1.) en ce sens qu'il offre une structure de communication aux parties. L'apport de ce prototype par rapport à l'INSS est qu'il permet aux parties de discuter sur les termes du contrat de manière flexible et dynamique.

Dans l'INSS, les attributs de la négociation doivent être fixés avant de commencer la négociation. La négociation entre les parties est donc limitée aux seuls attributs prédéfinis. Dans le prototype que nous proposons permet de contourner cette limite en offrant deux parties dans las structure d'une offre. Une partie fixe dont les attributs sont prédéfinis et une partie dynamique dans laquelle les parties peuvent ajouter ou supprimer des attributs.

Le CBSS offre un outil de communication flexible mais pas d'historique de la négociation. Notre prototype offre un historique de la négociation, en plus d'une structure flexible.

5.5. Réalisation du prototype

Pour développer une application complète de ce prototype, il faut réaliser une application Client-Serveur. Le serveur étant la plate-forme d'intermédiation qui va gérer les différents dialogues entre l'acheteur et le vendeur. Nous n'avons pas eu le temps de développer une application complète dans le cadre de ce mémoire. Nous nous sommes donc limités à développer une petite application java qui simule l'enchaînement des dialogues pour un seul utilisateur. Voir la présentation des interfaces dans les annexes.

5.6. Extension possible du prototype

Toujours dans le soucis d'assister l'utilisateur dans la réalisation de sa tâche de négociation, on pourrait utiliser des agents pour aider les utilisateurs dans leur prise de décision

face à une offre. Ces agents pourraient évaluer l'offre reçue et faire des propositions à l'utilisateur pour l'aider à faire ses choix.

On pourrait également ajouter un mécanisme de signature électronique de sorte que le contrat négocié en utilisant ce support soit valable légalement.

Conclusion

Ce travail montre l'importance de la négociation dans la nouvelle économie qu'implique le commerce électronique. Le manque de sécurité sur l'Internet freine encore beaucoup de clients potentiels à réaliser leurs achats en ligne. Il y a un réel besoin de commencer par satisfaire les attentes des utilisateurs en matière de sécurité des transactions sur l'Internet. Une fois la sécurité assurée, les utilisateurs pourront alors avoir de l'intérêt pour les systèmes susceptibles de les aider à réaliser leurs transactions commerciales sur l'Internet. Les transactions qui ne nécessitent pas une négociation sont les plus courantes à l'heure actuelle sur l'Internet. Toutefois, elles souffrent encore du manque de confiance des utilisateurs quant à la sécurité des transactions financières (paiement par carte de crédit).

La tendance actuelle est de créer des systèmes automatiques susceptibles de réaliser par eux-mêmes les tâches sur l'Internet. L'idée n'est pas mauvaise, toutefois, il nous semble que certaines tâches pourraient difficilement être automatisables. La négociation est une tâche qui est difficile à automatiser car elle nécessite certaines aptitudes telle que l'intuition qu'il est difficile à faire acquérir à un agent logiciel. Nous pouvons dès lors nous poser des questions philosophiques sur l'intelligence artificielle, à savoir "une machine peut-elle être intelligente"?. Les avis divergent sur la réponse à cette question et cela sort du cadre de notre travail. Mais la question reste ouverte pour les chercheurs qui tentent de créer des systèmes de négociation automatisée. Nous avons donc préféré utiliser, dans ce travail, le terme "support à la négociation" plutôt que d'utiliser le terme qui est repris dans la littérature. Nous estimons que l'intervention humaine sera toujours nécessaire dans des systèmes d'aide à la négociation.

Annexes

1. Spécifications fonctionnelles

ou = ou non exclusif, **ou** (= ou exclusif)

1. Module OFFRE

1.1) **enregistrer_offre** (→prix, →delai_livraison,, →conditions_retour,
→mode_paiement, →autres_conditions)

Cette fonction enregistre l'offre dans la base de données

Pré-condition:
Prix = " " ***et*** delai_livraison = " " ***et*** conditions_retour = " " ***et***
mode_paiement = " " ***et*** autres_conditions = " "

Post-condition:
Prix ≠ " "***et*** delai_livraison ≠ " " ***et*** conditions_retour ≠ " " ***et***
mode_paiement ≠ " " ***et*** (autres_conditions ≠ " " **ou** autres_conditions = " ")

1.2) **verif_modif_offre** (→prix, →delai_livraison, →conditions_retour,
→mode_paiement, →autres_conditions, ←offre_modifie)

Cette fonction vérifie si l'offre a été modifiée

Pré-condition:
$Prix_i$ ≠ " "***et*** $delai_livraison_i$ ≠ " " ***et*** $conditions_retour_i$ ≠ " " ***et***
$mode_paiement_i$ ≠ " " ***et*** ($autres_conditions_i$ ≠ " " **ou** $autres_conditions_i$ = " ")

Post-condition:
offre_modifie = vrai **si** $Prix_j$ ≠ $Prix_i$ ***ou*** $delai_livraison_j$ ≠ $delai_livraison_i$ ***ou*** $conditions_retour_j$ ≠ $conditions_retour_i$ ***ou*** $mode_paiement_j$ ≠ $mode_paiement_i$ ***ou*** ($autres_conditions_j$ ≠ $autres_conditions_i$)
offre_modifie = faux sinon

1.3) **valider_offre** (→prix, →delai_livraison,, →conditions_retour,
→mode_paiement, →autres_conditions, ← offre_valide)

Cette fonction vérifie si les cases obligatoires de l'offre sont remplies

Pré-condition:
offre_valide = faux

Post-condition:
offre_valide = faux **si** Prix = " "***ou*** delai_livraison = " " ***ou*** conditions_retour = " " ***ou***
mode_paiement = " "
ou
offre_valide = vrai **si** Prix ≠ " "***et*** delai_livraison ≠ " " ***et*** conditions_retour ≠ " " ***et***
mode_paiement ≠ " "

2. Module VENDEUR

2.1) **verifier_mot_passe** (→mot_passe, →login, ←mot_passe_correct)

Cette fonction vérifie si le mot de passe fournit correspond au login introduit

Pré-condition:
mot_passe_correct = faux

Post-condition:
mot_passe_correct = vrai **si** mot_passe correspond à login
ou
mot_passe_correct = faux **si** mot_passe ne correspond pas à login → message erreur

2.2) **verifier_login** (→mot_passe, →login, ←login_correct)

Cette fonction vérifie si le nom d'utilisateur introduit correspond à un utilisateur existant

Pré-condition:
login_correct = faux

Post-condition:
login_correct = vrai **si** login existe dans BD_VENDEUR
ou
login_correct = faux **si** login n'existe pas dans BD_VENDEUR → message erreur

3. Module ACHETEUR

3.1) **enregistrer_acheteur** (→nom, → prénom, →nom_utilisateur, →mot_passe, →adresse, →adresse_e-mail, →mode_paiement)

Cette fonction enregistre un nouvel utilisateur dans la base de données

Pré-condition:
Acheteur ∉ à la BD

Post-condition:
Acheteur ∈ à la BD

3.3) **verifier_mot_passe** (→mot_passe, →login, ←mot_passe_correct)

Cette fonction vérifie si le mot de passe fournit correspond au login introduit

Pré-condition:
mot_passe_correct = faux

Post-condition:
mot_passe_correct = true **si** mot_passe correspond à login
ou
mot_passe_correct = faux **si** mot_passe ne correspond pas à login → message erreur

3.4) **valider_login** (→mot_passe, →login, ←login_valide)

Cette fonction vérifie si nouvel utilisateur n'utilise pas un login déjà utilisé par un autre utilisateur déjà existant dans la base de données

Pré-condition:
login_valide = faux

Post-condition:
login_valide = vrai **si** login existe dans BD_ACHETEUR
ou
login_valide = faux **si** login n'existe pas dans BD_ACHETEUR → message erreur

3.5) **valider_données** (→nom, → prénom, →nom_utilisateur, →mot_passe, →adresse, →adresse_e-mail, →mode_paiement)

Cette fonction vérifie si l'utilisateur a introduit toutes les informations obligatoires pour son enregistrement

Pré-condition:
données_valides = faux

Post-condition:
données_valides = vrai **si** nom ≠ " ", prénom ≠ " " , nom_utilisateur ≠ " " , mot_passe ≠ " " , adresse ≠ " " , adresse_e-mail ≠ " " , mode_paiement ≠ " "
ou
données_valides = faux **si** (nom = " " ***ou*** prénom = " " ***ou*** nom_utilisateur = " " ***ou*** mot_passe = " " ***ou*** adresse = " " ***ou*** adresse_e-mail = " " ***ou*** mode_paiement = " ") → message erreur

4. Module ACCORD

4.1) **enregistrer_accord** (→offre, →message_acc)

Cette fonction enregistre un accord dans la base des données.

Pré-condition:

Post-condition: accord enregistré dans BD_ACCORD

5. Module CONF_ACCORD

5.1) **enregistrer_conf** (→accord, →message_conf)

Cette fonction enregistre la confirmation d'une offre dans la base de données

Pré-condition:

Post-condition: confirmation enregistré dans BD_CONF_ACCORD

5.2) **vérifier_accord** (←accord_enregistre)

Cette fonction vérifie si un accord a été enregistré dans la base de données

Pré-condition:

Post-condition: accord_enregistre = vrai **si** accord enregistré dans BD_ACCORD
ou
acccord_enregistre = faux **si** accord pas enregistré dans BD_ACCORD

6. Module HISTORIQUE

6.1) **enregistrer_offre** (→offre)

Cette fonction enregistre une offre dans l'historique de la négociation

Pré-condition:

Post-condition: offre enregistré dans BD_HISTORIQUE

6.2) **enregistrer_accord** (→accord)

Cette fonction enregistre un accord dans l'historique de la négociation

Pré-condition:

Post-condition: accord enregistré dans BD_HISTORIQUE

6.3) **enregistrer_dde_information** (→dde_info)

Cette fonction enregistre une demande d'information (un message) dans la l'historique de la négociation

Pré-condition:

Post-condition: dde_info enregistré dans BD_HISTORIQUE

6.4) **enregistrer_conf_acc** (→conf_accord)

Cette fonction enregistre la confirmation d'une offre dans l'historique de la négociation

Pré-condition:

Post-condition: conf_acc enregistré dans BD_HISTORIQUE

6.5) **enregistrer_abandon** (→abandon)

Cette fonction enregistre un abandon dans l'historique de la négociation

Pré-condition:

Post-condition: abandon enregistré dans BD_HISTORIQUE

6.6) **consulter_historique** (→historique)

Cette fonction permet d'afficher l'historique

Pré-condition:

Post-condition: afficher historique

7. Module DEMANDE D'INFO

7.1) **enregistrer_dde_information** (→adr_mail_exp, →adr_mail_dest, →date, →heure, →objet, →contenu)

Cette fonction enregistre une demande d'information dans la base de données

Pré-condition:

Post-condition: dde_info enregistré dans BD_DEMANDE_INFO

8. Module ABANDON_NEGO

8.1) **enregistrer_abandon** ( )

Cette fonction enregistre un abandon dans la base de données

Pré-condition:

Post-condition: abandon enregistré dans BD_ABANDON_NEGO

9. Module BD

Ce module offre ses services à tous les autres modules précédents pour la gestion des données

9.1) **enregistrer_offre** (→prix, →delai_livraison,, →conditions_retour, →mode_paiement, →autres_conditions)

Pré-condition:

Post-condition: offre enregistré dans BD_OFFRE

9.2) **enregistrer_accord** (→offre, →message_acc)

Pré-condition:

Post-condition: accord enregistré dans BD_ACCORD

9.3) **enregistrer_dde_information** (→adr_mail_exp, →adr_mail_dest, →date, →heure, →objet, →contenu)

Pré-condition:

Post-condition: dde_info enregistré dans BD_DEMANDE_INFO

9.4) **enregistrer_conf_acc** (→accord, →message_conf)

Pré-condition:

Post-condition: confirmation enregistré dans BD_CONF_ACCORD

9.5) **enregistrer_abandon** ( )

Pré-condition:

Post-condition: abandon enregistré dans BD_ABANDON_NEGO

9.6) **consulter_historique** (→historique)

Pré-condition:

Post-condition: afficher historique

9.7) **supprimer_acheteur** (→acheteur)

Pré-condition: acheteur existe dans BD_ACHETEUR

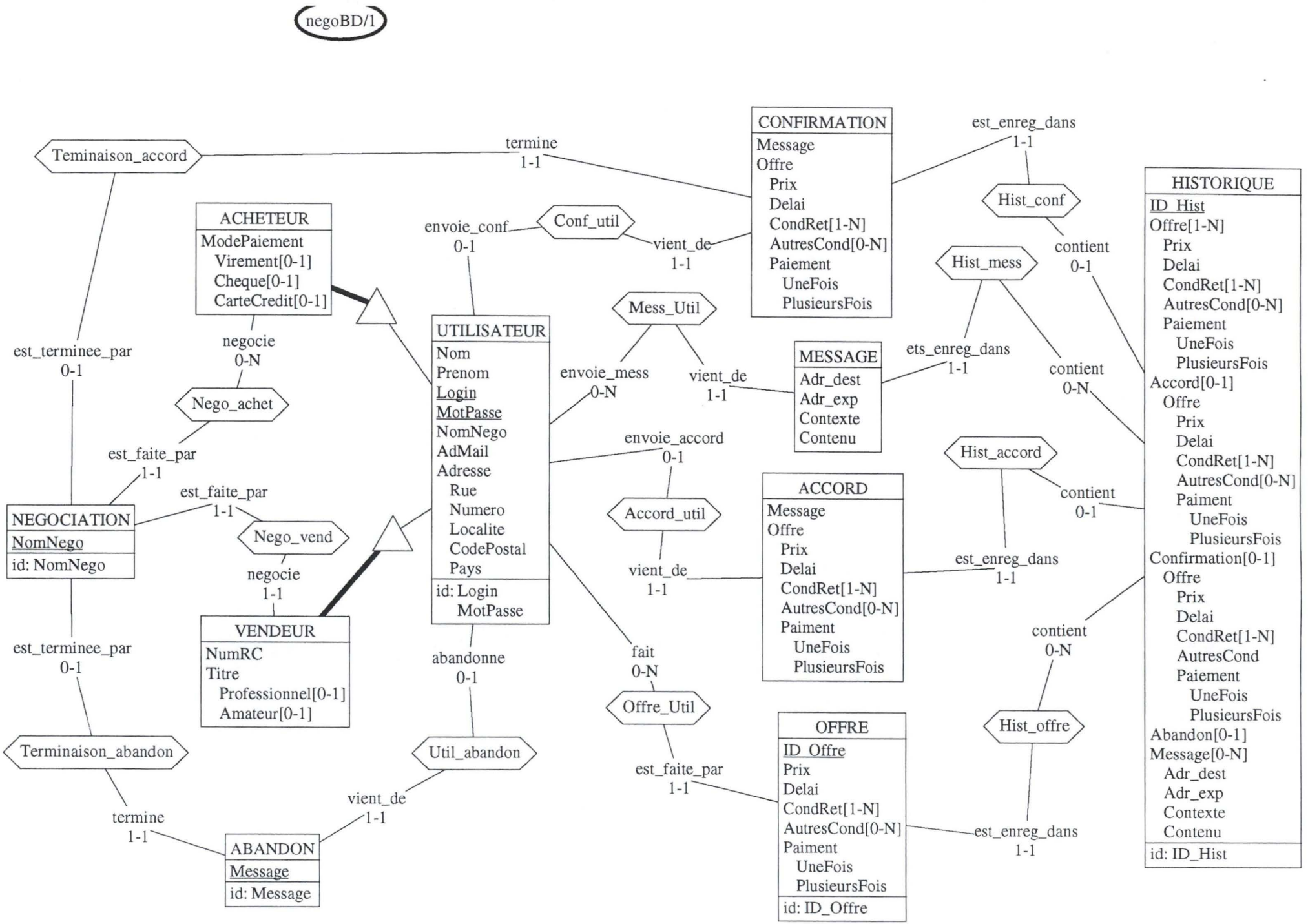
Post-condition: acheteur n'existe pas dans BD_ACHETEUR

9.8) **ajouter_acheteur** (→acheteur)

Pré-condition: acheteur n'existe pas dans BD_ACHETEUR

Post-condition: acheteur existe dans BD_ACHETEUR

2. *Schéma entité-association*


negoBD/1
Teminaison_accord
termine
1-1
CONFIRMATION
Message
Offre
Prix
Delai
CondRet[1-N]
AutresCond[0-N]
Paiement
UneFois
PlusieursFois
est_enreg_dans
1-1
Hist_conf
contient
0-1
HISTORIQUE
ID_Hist
Offre[1-N]
Prix
Delai
CondRet[1-N]
AutresCond[0-N]
Paiement
UneFois
PlusieursFois
Accord[0-1]
Offre
Prix
Delai
CondRet[1-N]
AutresCond[0-N]
Paiment
UneFois
PlusieursFois
Confirmation[0-1]
Offre
Prix
Delai
CondRet[1-N]
AutresCond
Paiement
UneFois
PlusieursFois
Abandon[0-1]
Message[0-N]
Adr_dest
Adr_exp
Contexte
Contenu
id: ID_Hist
ACHETEUR
ModePaiement
Virement[0-1]
Cheque[0-1]
CarteCredit[0-1]
envoie_conf
0-1
Conf_util
vient_de
1-1
Hist_mess
contient
0-N
Mess_Util
UTILISATEUR
Nom
Prenom
Login
MotPasse
NomNego
AdMail
Adresse
Rue
Numero
Localite
CodePostal
Pays
id: Login
MotPasse
envoie_mess
0-N
vient_de
1-1
MESSAGE
Adr_dest
Adr_exp
Contexte
Contenu
ets_enreg_dans
1-1
contient
0-N
negocie
0-N
Nego_achet
est_faite_par
1-1
envoie_accord
0-1
Hist_accord
contient
0-1
est_terminee_par
0-1
NEGOCIATION
NomNego
id: NomNego
est_faite_par
1-1
Nego_vend
negocie
1-1
Accord_util
vient_de
1-1
ACCORD
Message
Offre
Prix
Delai
CondRet[1-N]
AutresCond[0-N]
Paiment
UneFois
PlusieursFois
est_enreg_dans
1-1
contient
0-N
VENDEUR
NumRC
Titre
Professionnel[0-1]
Amateur[0-1]
est_terminee_par
0-1
abandonne
0-1
fait
0-N
Offre_Util
Hist_offre
Terminaison_abandon
Util_abandon
vient_de
1-1
est_faite_par
1-1
OFFRE
ID_Offre
Prix
Delai
CondRet[1-N]
AutresCond[0-N]
Paiment
UneFois
PlusieursFois
id: ID_Offre
est_enreg_dans
1-1
termine
1-1
ABANDON
Message
id: Message

3. Code SQL de la base de données

```
Database Section
create database negoBD;

create table ABANDON (
    Message char(50) not null,
    Abandonne_Login char(15) not null,
    Abandonne_MotPasse char(15) not null,
    Est_terminee_par char(15) not null,
    primary key (Message),
    unique (Abandonne_Login, Abandonne_MotPasse),
    unique (Est_terminee_par));

create table ACCORD (
    Contient char(6) not null,
    Envoie_accord_Login char(15) not null,
    Envoie_accord_MotPasse char(15) not null,
    Message char(300) not null,
    Off_Prix char(12) not null,
    Off_Delai char(20) not null,
    Off_Pai_UneFois char not null,
    Off_Pai_PlusieursFois char not null,
    primary key (Contient),
    unique (Envoie_accord_Login, Envoie_accord_MotPasse));

create table ACHETEUR (
    Est_faite_par char(15),
    Login char(15),
    MotPasse char(15),
    Mod_Virement char,
    Mod_Cheque char,
    Mod_CarteCredit char,
    unique (Est_faite_par),
    unique (Login, MotPasse));

create table AutresCond (
    ID_Offre char(1) not null,
    AutresCond char(500) not null,
    primary key (ID_Offre, AutresCond));

create table CondRet (
    ID_Offre char(1) not null,
    CondRet char(200) not null,
    primary key (ID_Offre, CondRet));

create table CONFIRMATION (
    Est_terminee_par char(15) not null,
    Contient char(6) not null,
    Envoie_conf_Login char(15) not null,
```

```
    Envoie_conf_MotPasse char(15) not null,
    Message char(12) not null,
    Off_Prix char(12) not null,
    Off_Delai char(20) not null,
    Off_Pai_UneFois char not null,
    Off_Pai_PlusieursFois char not null,
    primary key (Est_terminee_par),
    unique (Contient),
    unique (Envoie_conf_Login, Envoie_conf_MotPasse));

create table HISTORIQUE (
    ID_Hist char(6) not null,
    Accord -- Compound attribute --,
    Confirmation -- Compound attribute --,
    Abandon char(15),
    primary key (ID_Hist));

create table MESSAGE (
    Adr_dest char(15) not null,
    Adr_exp char(15) not null,
    Contexte char(200) not null,
    Contenu char(500) not null,
    Envoie_mess_Login char(15) not null,
    Envoie_mess_MotPasse char(15) not null,
    Contient char(6) not null);

create table Message_1 (
    ID_Hist char(6) not null,
    Adr_dest char(15) not null,
    Adr_exp char(15) not null,
    Contexte char(200) not null,
    Contenu char(500) not null,
    primary key (ID_Hist, Adr_dest, Adr_exp, Contexte, Contenu));

create table NEGOCIATION (
    NomNego char(15) not null,
    primary key (NomNego));

create table OFFRE (
    ID_Offre char(1) not null,
    Prix char(12) not null,
    Delai char(20) not null,
    Pai_UneFois char not null,
    Pai_PlusieursFois char not null,
    Fait_Login char(15) not null,
    Fait_MotPasse char(15) not null,
    Contient char(6) not null,
    primary key (ID_Offre));

create table Offre_1 (
    ID_Hist char(6) not null,
    Offre -- Compound attribute -- not null,
```

```
    primary key (ID_Hist, Offre -- Compound attribute --));

create table UTILISATEUR (
    Nom char(15) not null,
    Prenom char(15) not null,
    Login char(15) not null,
    MotPasse char(15) not null,
    NomNego char(15) not null,
    AdMail char(15) not null,
    Adr_Rue char(15) not null,
    Adr_Numero char(6) not null,
    Adr_Localite char(15) not null,
    Adr_CodePostal char(15) not null,
    Adr_Pays char(15) not null,
    primary key (Login, MotPasse));

create table VENDEUR (
    Est_faite_par char(15) not null,
    Login char(15),
    MotPasse char(15),
    NumRC char(15) not null,
    Tit_Professionnel char,
    Tit_Amateur char,
    unique (Login, MotPasse),
    primary key (Est_faite_par));

create table Off_AutresCond (
    Est_terminee_par char(15) not null,
    Off_AutresCond char(500) not null,
    primary key (Est_terminee_par, Off_AutresCond));

create table Off_AutresCond_1 (
    Contient char(6) not null,
    Off_AutresCond char(500) not null,
    primary key (Contient, Off_AutresCond));

create table Off_CondRet (
    Est_terminee_par char(15) not null,
    Off_CondRet char(300) not null,
    primary key (Est_terminee_par, Off_CondRet));

create table Off_CondRet_1 (
    Contient char(6) not null,
    Off_CondRet char(300) not null,
    primary key (Contient, Off_CondRet));


-- Constraints Section
-- ____________________

alter table ABANDON add constraint FKUtil_abandon
    foreign key (Abandonne_Login, Abandonne_MotPasse)
```

```
    references UTILISATEUR;

alter table ABANDON add constraint FKTerminaison_abandon
    foreign key (Est_terminee_par)
    references NEGOCIATION;

--alter table ACCORD add constraint
--    check(exists(select * from Off_CondRet_1
--            where Off_CondRet_1.Contient = Contient));

alter table ACCORD add constraint FKHist_accord
    foreign key (Contient)
    references HISTORIQUE;

alter table ACCORD add constraint FKAccord_util
    foreign key (Envoie_accord_Login, Envoie_accord_MotPasse)
    references UTILISATEUR;

alter table ACHETEUR add constraint FKNego_achet
    foreign key (Est_faite_par)
    references NEGOCIATION;

alter table ACHETEUR add constraint FKACH_UTI
    foreign key (Login, MotPasse)
    references UTILISATEUR;

--alter table ACHETEUR add constraint FKACH_UTI
--    check((Login is not null and MotPasse is not null)
--        or (Login is null and MotPasse is null));

alter table AutresCond add constraint FKOFF_Aut
    foreign key (ID_Offre)
    references OFFRE;

alter table CondRet add constraint FKOFF_Con
    foreign key (ID_Offre)
    references OFFRE;

--alter table CONFIRMATION add constraint
--    check(exists(select * from Off_CondRet
--            where Off_CondRet.Est_terminee_par = Est_terminee_par));

alter table CONFIRMATION add constraint FKTeminaison_accord
    foreign key (Est_terminee_par)
    references NEGOCIATION;

alter table CONFIRMATION add constraint FKHist_conf
    foreign key (Contient)
    references HISTORIQUE;

alter table CONFIRMATION add constraint FKConf_util
    foreign key (Envoie_conf_Login, Envoie_conf_MotPasse)
```

```
    references UTILISATEUR;

--alter table HISTORIQUE add constraint
--    check(exists(select * from Offre_1
--            where Offre_1.ID_Hist = ID_Hist));

alter table MESSAGE add constraint FKMess_Util
    foreign key (Envoie_mess_Login, Envoie_mess_MotPasse)
    references UTILISATEUR;

alter table MESSAGE add constraint FKHist_mess
    foreign key (Contient)
    references HISTORIQUE;

alter table Message_1 add constraint FKHIS_Mes
    foreign key (ID_Hist)
    references HISTORIQUE;

--alter table NEGOCIATION add constraint
--    check(exists(select * from ACHETEUR
--            where ACHETEUR.Est_faite_par = NomNego));

--alter table NEGOCIATION add constraint
--    check(exists(select * from VENDEUR
--            where VENDEUR.Est_faite_par = NomNego));

--alter table OFFRE add constraint
--    check(exists(select * from CondRet
--            where CondRet.ID_Offre = ID_Offre));

alter table OFFRE add constraint FKOffre_Util
    foreign key (Fait_Login, Fait_MotPasse)
    references UTILISATEUR;

alter table OFFRE add constraint FKHist_offre
    foreign key (Contient)
    references HISTORIQUE;

alter table Offre_1 add constraint FKHIS_Off
    foreign key (ID_Hist)
    references HISTORIQUE;

--alter table UTILISATEUR add constraint
--    check(exists(select * from ACHETEUR
--            where ACHETEUR.Login = Login and ACHETEUR.MotPasse = MotPasse));

--alter table UTILISATEUR add constraint
--    check(exists(select * from VENDEUR
--            where VENDEUR.Login = Login and VENDEUR.MotPasse = MotPasse));

alter table VENDEUR add constraint FKVEN_UTI
    foreign key (Login, MotPasse)
```

```
    references UTILISATEUR;

--alter table VENDEUR add constraint FKVEN_UTI
--    check((Login is not null and MotPasse is not null)
--          or (Login is null and MotPasse is null));

alter table VENDEUR add constraint FKNego_vend
    foreign key (Est_faite_par)
    references NEGOCIATION;

alter table Off_AutresCond add constraint FKCON_Off_1
    foreign key (Est_terminee_par)
    references CONFIRMATION;

alter table Off_AutresCond_1 add constraint FKACC_Off_1
    foreign key (Contient)
    references ACCORD;

alter table Off_CondRet add constraint FKCON_Off
    foreign key (Est_terminee_par)
    references CONFIRMATION;

alter table Off_CondRet_1 add constraint FKACC_Off
    foreign key (Contient)
    references ACCORD;


-- Index Section
-- _____

create unique index IDABANDON
    on ABANDON (Message);

create unique index FKUtil_abandon
    on ABANDON (Abandonne_Login, Abandonne_MotPasse);

create unique index FKTerminaison_abandon
    on ABANDON (Est_terminee_par);

create unique index FKHist_accord
    on ACCORD (Contient);

create unique index FKAccord_util
    on ACCORD (Envoie_accord_Login, Envoie_accord_MotPasse);

create unique index FKNego_achet
    on ACHETEUR (Est_faite_par);

create unique index FKACH_UTI
    on ACHETEUR (Login, MotPasse);

create unique index IDAutresCond
```

```
    on AutresCond (ID_Offre, AutresCond);

create index FKOFF_Aut
    on AutresCond (ID_Offre);

create unique index IDCondRet
    on CondRet (ID_Offre, CondRet);

create index FKOFF_Con
    on CondRet (ID_Offre);

create unique index FKTeminaison_accord
    on CONFIRMATION (Est_terminee_par);

create unique index FKHist_conf
    on CONFIRMATION (Contient);

create unique index FKConf_util
    on CONFIRMATION (Envoie_conf_Login, Envoie_conf_MotPasse);

create unique index IDHISTORIQUE
    on HISTORIQUE (ID_Hist);

create index FKMess_Util
    on MESSAGE (Envoie_mess_Login, Envoie_mess_MotPasse);

create index FKHist_mess
    on MESSAGE (Contient);

create unique index IDMessage_1
    on Message_1 (ID_Hist, Adr_dest, Adr_exp, Contexte, Contenu);

create index FKHIS_Mes
    on Message_1 (ID_Hist);

create unique index NEGOCIATION
    on NEGOCIATION (NomNego);

create unique index IDOFFRE
    on OFFRE (ID_Offre);

create index FKOffre_Util
    on OFFRE (Fait_Login, Fait_MotPasse);

create index FKHist_offre
    on OFFRE (Contient);

create unique index IDOffre_1
    on Offre_1 (ID_Hist, Offre -- Compound attribute --);

create index FKHIS_Off
    on Offre_1 (ID_Hist);
```

```
create unique index IDUTILISATEUR
   on UTILISATEUR (Login, MotPasse);

create unique index FKVEN_UTI
   on VENDEUR (Login, MotPasse);

create unique index FKNego_vend
   on VENDEUR (Est_faite_par);

create unique index IDOff_AutresCond
   on Off_AutresCond (Est_terminee_par, Off_AutresCond);

create index FKCON_Off_1
   on Off_AutresCond (Est_terminee_par);

create unique index IDOff_AutresCond_1
   on Off_AutresCond_1 (Contient, Off_AutresCond);

create index FKACC_Off_1
   on Off_AutresCond_1 (Contient);

create unique index IDOff_CondRet
   on Off_CondRet (Est_terminee_par, Off_CondRet);

create index FKCON_Off
   on Off_CondRet (Est_terminee_par);

create unique index IDOff_CondRet_1
   on Off_CondRet_1 (Contient, Off_CondRet);

create index FKACC_Off
   on Off_CondRet_1 (Contient);
```

4. Codes sources

```
//Identification d'un utilisateur

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;


//
public class Identification extends JPanel
        {
        private Action action;
        GridBagConstraints constraints = new GridBagConstraints();
        JTextField ed_login = new JTextField(15);
        JTextField ed_pwd = new JTextField(15);
        //
        public Identification ()
                {
                this.setLayout(new GridBagLayout());
                this.setBackground(SystemColor.info);
                constraints.fill = GridBagConstraints.NONE;
                constraints.anchor = GridBagConstraints.CENTER;
                constraints.insets = new Insets(5, 5, 5, 5);
                constraints.weightx = 1.0;
                constraints.weighty = 1.0;
                int x, y;
                constraints.gridheight = 1;
                constraints.gridwidth = 2;
                JLabel label1 = new JLabel("Si vous êtes un nouvel utilisateur, veuillez cliquer ");
                label1.setFont(new java.awt.Font("Dialog", 1, 14));
                label1.setForeground(Color.blue);
                addGB(label1, x=0, y=0);
                JLabel label2 = new JLabel("sur le bouton nouvel utilisateur pour vous
                enregistrer");
                label2.setFont(new java.awt.Font("Dialog", 1, 14));
                label2.setForeground(Color.blue);
                addGB(label2, x=0, y=1);
                JLabel label3 = new JLabel("Si vous êtes un ancien utilisateur, veuillez
                introduire");
                label3.setFont(new java.awt.Font("Dialog", 1, 14));
                label3.setForeground(Color.blue);
                addGB(label3, x=0, y=3);
                JLabel label4 = new JLabel("votre nom d'utilisateur et votre mot de passe");
                label4.setFont(new java.awt.Font("Dialog", 1, 14));
                label4.setForeground(Color.blue);
                addGB(label4, x=0, y=4);
                //constraints.gridwidth = 1;
```

```
        JButton bt_NewUser = new JButton("Nouvel utilisateur");
        bt_NewUser.setFont(new java.awt.Font("Dialog", 1, 14));
        bt_NewUser.setForeground(Color.blue);
        bt_NewUser.addActionListener(new bt_NewUserListener());
        addGB(bt_NewUser, x=0, y=2);

        constraints.weightx = 0.5;
        constraints.gridwidth = 1;
        //constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.anchor = GridBagConstraints.EAST;
        JLabel nom = new JLabel("Nom d'utilisateur :");
        nom.setFont(new java.awt.Font("Dialog", 1, 14));
        nom.setForeground(Color.blue);
        addGB(nom, x=0, y=5);
        JLabel pwd = new JLabel("Mot de passe :");
        pwd.setFont(new java.awt.Font("Dialog", 1, 14));
        pwd.setForeground(Color.blue);
        addGB(pwd, x=0, y=6);
        //constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.anchor = GridBagConstraints.WEST;
        constraints.weightx = 0.5;
        addGB(ed_login, x=1, y=5);
        addGB(ed_pwd, x=1, y=6);
        //constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.anchor = GridBagConstraints.EAST;
        JButton bt_valider = new JButton("Valider");
        bt_valider.setFont(new java.awt.Font("Dialog", 1, 14));
        bt_valider.setForeground(Color.blue);
        bt_valider.addActionListener(new bt_validerListener());
        addGB(bt_valider, x=0, y=7);

        constraints.anchor = GridBagConstraints.WEST;
        JButton bt_annuler = new JButton("Annuler");
        bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
        bt_annuler.setForeground(Color.blue);
        bt_annuler.addActionListener(new bt_annulerListener());
        addGB(bt_annuler, x=1, y=7);
        }

//
void addGB(Component component, int x, int y)
        {
        constraints.gridx = x;
        constraints.gridy = y;
        add(component, constraints);
        }
//
public String getLogin()
        {
        return ed_login.getText();
        }
```

```
        //
        public void setLogin(String login)
                {
                ed_login.setText(login);
                }

        //
        public String getPwd()
                {
                return ed_pwd.getText();
                }

        //
        public void setPwd(String password)
                {
                ed_pwd.setText(password);
                }

        //
        public void addLoginListener(ActionListener listener)
                {
                ed_login.addActionListener(listener);
                }

        public class bt_validerListener implements ActionListener
                {
                public void actionPerformed(ActionEvent e)
                        {
                        if ((getLogin() != null) & (getPwd() != null))
                                {
                                Action action = new Action();
                                Frame f = new Frame("Choix de l'action à effectuer ");
                                f.addWindowListener(new WindowAdapter()
                                        {
                                        public void windowClosing(WindowEvent we)
                                                {
                                                System.exit(0);
                                                }
                                        });
                                f.setSize(400,400);
                                f.setLocation(200, 50);
                                f.add("Center", action);
                                f.setVisible(true);
                                }
                        else JOptionPane.showMessageDialog(null, "Nom d'utilisateur ou mot de
passe incorrect");
                        }
                }

        //

        public class bt_annulerListener implements ActionListener
```

```
        {
        public void actionPerformed(ActionEvent e)
                {
                System.exit(0);
                }
        }

//
public class bt_NewUserListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Utilisateur user = new Utilisateur();
                Frame f = new Frame("Enregistrement d'un nouvel utilisateur");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(700,500);
                f.add("Center",user);
                f.setLocation(80, 100);
                f.setVisible(true);
                }
        }
//
public static void main(String[] args)
        {
        Identification ident = new Identification();
        Frame f = new Frame("Formulaire de l'identification");
        f.addWindowListener(new WindowAdapter()
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
                });
        f.setSize(400,350);
        f.setLocation(200, 50);
        f.add("Center", ident);
        f.setVisible(true);
        }
}
```

```
//Affichage de l'adresse

import java.awt.*;
import java.awt.event.*;
import java.lang.*;
```

```
import javax.swing.border.*;
import javax.swing.*;
import java.util.*;

//
public class Adresse extends JPanel
        {
        //attributs
        String Rue;
        String Numero;
        String CodePostal;
        String Localite;
        String Pays;
        static GridBagConstraints constraints = new GridBagConstraints();
        //static Container content=new JPanel();
        //
        void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                add(component, constraints);
                }
        //
        /*public Adresse(String rue, String numero, String codePostal, String localite, String pays)
                {
                this.Rue=rue;
                this.Numero=numero;
                this.CodePostal=codePostal;
                this.Localite=localite;
                this.Pays=pays;
                }*/
        public Adresse()
                {
                JTextField ed_rue = new JTextField(15);
                JLabel Lrue = new JLabel("Rue :");
                Lrue.setFont(new java.awt.Font("Dialog", 1, 14));
                Lrue.setForeground(Color.blue);
                JTextField ed_numero = new JTextField(15);
                JLabel Lnumero = new JLabel("Numéro :");
                Lnumero.setFont(new java.awt.Font("Dialog", 1, 14));
                Lnumero.setForeground(Color.blue);
                JTextField ed_CP = new JTextField(15);
                JLabel LcodePostal = new JLabel("Code Postal :");
                LcodePostal.setFont(new java.awt.Font("Dialog", 1, 14));
                LcodePostal.setForeground(Color.blue);
                JTextField ed_loc = new JTextField(15);
                JLabel Llocalite = new JLabel("Localité :");
                Llocalite.setFont(new java.awt.Font("Dialog", 1, 14));
                Llocalite.setForeground(Color.blue);
                JTextField ed_pays = new JTextField(15);
                JLabel Lpays = new JLabel("Pays :");
                Lpays.setFont(new java.awt.Font("Dialog", 1, 14));
```

```
Lpays.setForeground(Color.blue);

//JPanel content = new JPanel();
GridBagLayout gridbag = new GridBagLayout();
this.setLayout(gridbag);
this.setBackground(SystemColor.info);
JPanel Adr = new JPanel();
Adr.setBorder(new TitledBorder(new EtchedBorder(),"Adresse"));
//Adr.setLayout(gridbag);

constraints.fill = GridBagConstraints.VERTICAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets=new Insets(0,0,0,0);
constraints.weightx = 1;
constraints.weighty = 1;
gridbag.setConstraints(Adr,constraints);
//content.add(Adr);
this.add(Adr);

GridBagLayout gb1=new GridBagLayout();
GridBagConstraints constraints1=new GridBagConstraints();
Adr.setLayout(gb1);
Adr.setBackground(SystemColor.info);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.NORTHWEST;
constraints1.insets=new Insets(5,5,5,5);
constraints1.gridwidth = 1;

constraints1.gridx = 0;
constraints1.gridy = 0;
gb1.setConstraints(Lrue, constraints1);

constraints1.gridx = 1;
constraints1.gridy = 0;
gb1.setConstraints(ed_rue, constraints1);

constraints1.gridx = 0;
constraints1.gridy = 1;
gb1.setConstraints(Lnumero, constraints1);

constraints1.gridx = 1;
constraints1.gridy = 1;
gb1.setConstraints(ed_numero, constraints1);

constraints1.gridx = 0;
constraints1.gridy = 2;
gb1.setConstraints(LcodePostal, constraints1);

constraints1.gridx = 1;
constraints1.gridy = 2;
gb1.setConstraints(ed_CP, constraints1);
```

```
        constraints1.gridx = 0;
        constraints1.gridy = 3;
        gb1.setConstraints(Llocalite, constraints1);

        constraints1.gridx = 1;
        constraints1.gridy = 3;
        gb1.setConstraints(ed_loc, constraints1);

        constraints1.gridx = 0;
        constraints1.gridy = 4;
        gb1.setConstraints(Lpays, constraints1);

        constraints1.gridx = 1;
        constraints1.gridy = 4;
        gb1.setConstraints(ed_pays, constraints1);

        Adr.add(Lrue);
        Adr.add(ed_rue);
        Adr.add(Lnumero);
        Adr.add(ed_numero);
        Adr.add(LcodePostal);
        Adr.add(ed_CP);
        Adr.add(Llocalite);
        Adr.add(ed_loc);
        Adr.add(Lpays);
        Adr.add(ed_pays);

        int x, y;

        addGB(Adr, x=0, y=0);
        }

//
void setRue(String rue)
        {
        Rue = rue;
        }

//
String getRue()
        {
        return Rue;
        }

//
void setNumero(String num)
        {
        Numero = num;
        }

//
```

```
String getNumero()
        {
        return Numero;
        }

//
void setCodePostal(String code)
        {
        CodePostal = code;
        }

//
String getCodePostal()
        {
        return CodePostal;
        }

//
void setLocalite(String loc)
        {
        Localite = loc;
        }

//
String getLocalite()
        {
        return Localite;
        }

//
void setPays(String pays)
        {
        Pays = pays;
        }

//
String getPays()
        {
        return Pays;
        }

//
public static void main(String[] args)
        {
        Adresse adresse = new Adresse();
        Frame f=new Frame();
        f.add("Center", adresse);
        //f.add("Center", content);
        f.show();
        }

}
```

```
==========================================================================

//Identification d'un nouvel utilisateur

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Utilisateur extends JFrame
        {
        // attributs
        private String Login;
        private String MotPasse;
        private String Nom;
        private String Prenom;
        private String eMail;
        private Adresse adr = new Adresse();

        static Container content = new JPanel();
        GridBagConstraints constraints = new GridBagConstraints();


        JTextField ed_nom = new JTextField(15);
        JTextField ed_prenom = new JTextField(15);
        JTextField ed_login = new JTextField(15);
        JTextField ed_password1 = new JTextField(15);
        JTextField ed_password2 = new JTextField(15);
        JTextField ed_e_mail = new JTextField(15);


        //
        public Utilisateur ()
                {
                super("Enregistrement d'un nouvel utilisateur");

                JLabel message1 = new JLabel("Veuillez remplir cette fiche pour votre
                enregistrement");
                JLabel message2 = new JLabel("en tant que nouvel utilisateur");
                message1.setFont(new java.awt.Font("Dialog", 1, 14));
                message1.setForeground(Color.blue);
                message2.setFont(new java.awt.Font("Dialog", 1, 14));
                message2.setForeground(Color.blue);

                JLabel nom = new JLabel("Nom :");
                nom.setFont(new java.awt.Font("Dialog", 1, 14));
                nom.setForeground(Color.blue);
                JLabel prenom = new JLabel("Prénom :");
                prenom.setFont(new java.awt.Font("Dialog", 1, 14));
                prenom.setForeground(Color.blue);
                JLabel login = new JLabel("Chosir votre nom d'utilisateur :");
```

```
login.setFont(new java.awt.Font("Dialog", 1, 14));
login.setForeground(Color.blue);
JLabel pwd1 = new JLabel("Choisir votre mot de passe :");
pwd1.setFont(new java.awt.Font("Dialog", 1, 14));
pwd1.setForeground(Color.blue);
JLabel pwd2 = new JLabel("Confirmer votre mot de passe :");
pwd2.setFont(new java.awt.Font("Dialog", 1, 14));
pwd2.setForeground(Color.blue);
JLabel admail = new JLabel("Adresse e-mail :");
admail.setFont(new java.awt.Font("Dialog", 1, 14));
admail.setForeground(Color.blue);
JCheckBox cb1 = new JCheckBox("Par virement");
cb1.setFont(new java.awt.Font("Dialog", 1, 14));
cb1.setForeground(Color.blue);
cb1.setBackground(SystemColor.info);
JCheckBox cb2 = new JCheckBox("Par chèque");
cb2.setFont(new java.awt.Font("Dialog", 1, 14));
cb2.setForeground(Color.blue);
cb2.setBackground(SystemColor.info);
JCheckBox cb3 = new JCheckBox("Par carte de crédit");
cb3.setFont(new java.awt.Font("Dialog", 1, 14));
cb3.setForeground(Color.blue);
cb3.setBackground(SystemColor.info);
JButton bt_enregistrer = new JButton("Enregistrer");
bt_enregistrer.setFont(new java.awt.Font("Dialog", 1, 14));
bt_enregistrer.setForeground(Color.blue);
bt_enregistrer.addActionListener(new btEnregListener());
JButton bt_annuler = new JButton("Annuler");
bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
bt_annuler.setForeground(Color.blue);
bt_annuler.addActionListener(new btAnnulerListener());
JPanel idPanel = new JPanel();
JPanel modePanel = new JPanel();
JPanel btPanel = new JPanel();

idPanel.setBorder(new TitledBorder(new EtchedBorder(),"Identification"));
modePanel.setBorder(new TitledBorder(new EtchedBorder(),"Paiement"));
btPanel.setBorder(new EtchedBorder());

GridBagLayout gb1 = new GridBagLayout();
GridBagConstraints constraints1 = new GridBagConstraints();
idPanel.setLayout(gb1);
idPanel.setBackground(SystemColor.info);

constraints1.fill = GridBagConstraints.NONE;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx = 0;
constraints1.gridy = 0;
constraints1.weightx =20;
constraints1.weighty =1;
gb1.setConstraints(nom, constraints1);
```

```
constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.gridx = 1;
constraints1.gridy = 0;
constraints1.weightx =80;
constraints1.weighty =1;
gb1.setConstraints(ed_nom, constraints1);

constraints1.fill = GridBagConstraints.NONE;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx = 0;
constraints1.gridy = 1;
constraints1.weightx =20;
constraints1.weighty =1;
gb1.setConstraints(prenom, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.gridx = 1;
constraints1.gridy = 1;
constraints1.weightx =80;
constraints1.weighty =1;
gb1.setConstraints(ed_prenom, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx =2 ;
constraints1.gridy = 0;
constraints1.weightx =20;
constraints1.weighty =1;
gb1.setConstraints(login, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.gridx = 3;
constraints1.gridy = 0;
constraints1.weightx =80;
constraints1.weighty =1;
gb1.setConstraints(ed_login, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx = 0;
constraints1.gridy = 2;
constraints1.weightx =20;
```

```
constraints1.weighty =1;
gb1.setConstraints(admail, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.gridx = 1;
constraints1.gridy = 2;
constraints1.weightx =80;
constraints1.weighty =1;
gb1.setConstraints(ed_e_mail, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx = 2;
constraints1.gridy = 1;
constraints1.weightx =20;
constraints1.weighty =1;
gb1.setConstraints(pwd1, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.gridx = 3;
constraints1.gridy = 1;
constraints1.weightx =80;
constraints1.weighty =1;
gb1.setConstraints(ed_password1, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 5, 5, 0);
constraints1.gridx = 2;
constraints1.gridy = 2;
constraints1.weightx =20;
constraints1.weighty =1;
gb1.setConstraints(pwd2, constraints1);

constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(5, 0, 5, 5);
constraints1.weightx =80;
constraints1.weighty =1;
constraints1.gridx = 3;
constraints1.gridy = 2;
gb1.setConstraints(ed_password2, constraints1);

idPanel.add(nom);
idPanel.add(ed_nom);
idPanel.add(prenom);
idPanel.add(ed_prenom);
```

```
idPanel.add(login);
idPanel.add(ed_login);
idPanel.add(pwd1);
idPanel.add(ed_password1);
idPanel.add(pwd2);
idPanel.add(ed_password2);
idPanel.add(admail);
idPanel.add(ed_e_mail);

GridBagLayout gb2 = new GridBagLayout();
GridBagConstraints constraints2 = new GridBagConstraints();
modePanel.setLayout(gb2);
modePanel.setBackground(SystemColor.info);
constraints2.fill = GridBagConstraints.VERTICAL;
constraints2.anchor = GridBagConstraints.CENTER;
constraints2.insets = new Insets(5, 5, 5, 5);
constraints2.gridwidth = 1;
constraints2.gridheight = 1;
constraints2.gridx = 0;
constraints2.gridy = 0;
gb2.setConstraints(cb1, constraints2);

constraints2.gridx = 1;
constraints2.gridy = 0;
gb2.setConstraints(cb2, constraints2);

constraints2.gridx = 2;
constraints2.gridy = 0;
gb2.setConstraints(cb3, constraints2);

modePanel.add(cb1);
modePanel.add(cb2);
modePanel.add(cb3);

GridBagLayout gb3 = new GridBagLayout();
GridBagConstraints constraints3 = new GridBagConstraints();
btPanel.setLayout(gb3);
btPanel.setBackground(SystemColor.info);
constraints3.fill = GridBagConstraints.VERTICAL;
constraints3.anchor = GridBagConstraints.CENTER;
constraints3.insets = new Insets(5, 5, 5, 5);
constraints3.gridwidth = 1;
constraints3.gridheight = 1;

constraints3.gridx = 0;
constraints3.gridy = 0;
gb3.setConstraints(bt_enregistrer, constraints3);

constraints3.gridx = 1;
constraints3.gridy = 0;
gb3.setConstraints(bt_annuler, constraints3);
```

```
btPanel.add(bt_enregistrer);
btPanel.add(bt_annuler);

int x, y;
content.setLayout(new GridBagLayout());
content.setBackground(SystemColor.info);
constraints.fill = GridBagConstraints.HORIZONTAL;
//content.add(message);

/*constraints.fill=GridBagConstraints.BOTH;
constraints.anchor=GridBagConstraints.CENTER;
constraints.insets=new Insets(5,5,5,5);
addGB(message1, x=0, y=0);*/

constraints.fill=GridBagConstraints.NONE;
constraints.anchor=GridBagConstraints.CENTER;
constraints.insets=new Insets(0,0,0,0);
constraints.weighty = 1;
constraints.weightx = 1;
constraints.gridwidth = 2;
constraints.gridheight =1;
addGB(message1, x=0, y=0);
addGB(message2, x=0, y=1);

constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(5, 5, 5, 5);
constraints.weighty = 1;
constraints.weightx = 1;
constraints.gridwidth = 2;
constraints.gridheight =1;
addGB(idPanel, x=0, y=2);

constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(0, 5, 0, 0);
constraints.weighty = 30;
constraints.weightx = 40;
constraints.gridwidth = 1;
constraints.gridheight =2;
addGB(adr, x=0, y=3);

constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.EAST;
constraints.insets = new Insets(0, 5, 0, 5);
constraints.weighty = 30;
constraints.weightx = 40;
constraints.gridwidth = 1;
constraints.gridheight =1;
addGB(modePanel, x=1, y=3);

constraints.fill = GridBagConstraints.NONE;
```

```
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(0, 5, 5, 5);
        constraints.weighty = 30;
        constraints.weightx = 60;
        constraints.gridwidth = 1;
        constraints.gridheight =1;
        addGB(btPanel, x=1, y=4);
        }

//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                content.add(component, constraints);
                }
//
void setLogin(String log)
        {
        ed_login.setText(log);
        }
String getLogin()
        {
        return ed_login.getText();
        }
//
void setMotPasse(String pwd)
        {
        ed_password1.setText(pwd);
        }
//
String getMotPasse()
        {
        return ed_password1.getText();
        }
//
void setMotPasse2(String pwd)
        {
        ed_password2.setText(pwd);
        }
//
String getMotPasse2()
        {
        return ed_password2.getText();
        }
//

void setNom(String n)
        {
        ed_nom.setText(n);
        }
//
```

```
String getNom()
        {
        return ed_nom.getText();
        }
//
void setPrenom(String pn)
        {
        ed_prenom.setText(pn);
        }
//
String getPrenom()
        {
        return ed_prenom.getText();
        }
//
void seteMail(String em)
        {
        ed_e_mail.setText(em);
        }
//
String geteMail()
        {
        return ed_e_mail.getText();
        }
//
public class btEnregListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }
//
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Identification ident = new Identification();
                Frame f = new Frame("Enregistrement d'un nouvel utilisateur");
                f.addWindowListener(new WindowAdapter()
```

```
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,350);
                f.setLocation(200, 50);
                f.add("Center",ident);
                f.setVisible(true);
                }
        }

    //
    public static void main(String[] args)
        {
        JFrame f = new Utilisateur();
        f.addWindowListener(new WindowAdapter()
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
                });
        f.setSize(700,500);
        f.setContentPane(content);
        f.setLocation(80, 100);
        f.setVisible(true);
        }
}
```

```
//Menu des actions possibles

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;


public class Action extends JFrame
    {

    public Action( )
        {
      super("Choix de l'action à effectuer");
          setSize(400, 300);
      setLocation(200, 200);

          // création du menu négociation
      JMenu negoc = new JMenu("Négociation");
```

```
negoc.setMnemonic(KeyEvent.VK_U);

JMenuItem mess = new JMenuItem("Envoyer un message");
mess.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                Message info = new Message();
                final Frame f = new Frame("Demande d'information");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                f.setVisible(false);
                                }
                        });
                f.setSize(500,500);
                f.add("Center",info);
                f.setLocation(150, 30);
                f.setVisible(true);
                }
        });

negoc.add(mess);

JMenuItem hist = new JMenuItem("Consulter l'historique");
hist.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                //System.exit(0);
                }
        });

negoc.add(hist);

JMenuItem fin = new JMenuItem("Terminer la session");
fin.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                System.exit(0);
                }
        });

negoc.add(fin);

negoc.addSeparator( );

JMenuItem abandon = new JMenuItem("Abandonner la négociation");
abandon.addActionListener(new ActionListener( )
        {
```

```
        public void actionPerformed(ActionEvent e)
                {
                int result = JOptionPane.showConfirmDialog(null,"Etes-vous sûr
                de vouloir annuler la négociation?");
                switch (result)
                        {
                        case JOptionPane.YES_OPTION:
                        JOptionPane.showMessageDialog(null, "Négociation
                        annulée. Aurevoir!");
                        System.exit(0);break;
                        case JOptionPane.NO_OPTION: break;
                        case JOptionPane.CANCEL_OPTION: break;
                        case JOptionPane.CLOSED_OPTION: break;
                        }
                }
        });

negoc.add(abandon);


// creation du menu offre
JMenu offre = new JMenu("Offre");
offre.setMnemonic(KeyEvent.VK_S);

JMenuItem of = new JMenuItem("Faire une offre");
of.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                Offre offre1 = new Offre();
                final Frame f = new Frame("Enregistrement d'une offre");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                f.setVisible(false);
                                }
                        });
                f.setSize(650,400);
                f.add("Center",offre1);
                f.setLocation(50, 50);
                f.show();
                }
        });

offre.add(of);

JMenuItem Cof = new JMenuItem("Faire une contre-offre");
Cof.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
```

```
                Offre offre1 = new Offre();
                final Frame f = new Frame("Enregistrement d'une offre");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                f.setVisible(false);
                                }
                        });
                f.setSize(650,400);
                f.add("Center",offre1);
                f.setLocation(50, 50);
                f.show();
                }
        });

offre.add(Cof);

offre.addSeparator( );

JMenuItem acc = new JMenuItem("Accepter une offre");
acc.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                Accord accord = new Accord();
                final Frame f = new Frame ("Acceptation d'une offre");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                        {
                        f.setVisible(false);
                        }
                });
                f.setSize(700,550);
                f.add("Center", accord);
                f.setLocation(50, 50);
                f.setVisible(true);
                }
        });

offre.add(acc);

JMenuItem conf = new JMenuItem("Confirmer une offre");
conf.addActionListener(new ActionListener( )
        {
        public void actionPerformed(ActionEvent e)
                {
                ConfOffre conf1 = new ConfOffre();
                final Frame f = new Frame("Acceptation d'une offre");
                f.addWindowListener(new WindowAdapter()
                        {
```

```
                                        public void windowClosing(WindowEvent we)
                                        {
                                        f.setVisible(false);
                                        }
                                });
                                f.setSize(750,600);
                                f.add("Center", conf1);
                                f.setLocation(30, 30);
                                f.setVisible(true);
                                System.exit(0);
                                }
                        });
                offre.add(conf);


                // create a menu bar and use it in this JFrame
                JMenuBar menuBar = new JMenuBar( );
                menuBar.add(negoc);
                menuBar.add(offre);
                setJMenuBar(menuBar);
        }

 public static void main(String[] args)
        {
  JFrame f = new Action( );
  f.addWindowListener(new WindowAdapter( )
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
          });
  f.setVisible(true);
        }
}
```

==========

```
//Affichage du formulaire de l'offre

//
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Offre extends JPanel
        {
        //attributs
        private int Prix;
        private String DelaiLivraison;
```

```
private Vector ConditionRetour;
/*
ModePaiement=0 => paiement en une fois;
ModePaiement=1 => paiment en tranches
*/
private byte ModePaiement;
private Vector AutresConditions;
GridBagConstraints constraints = new GridBagConstraints();
static Container content = new JPanel();
JPanel p1 = new JPanel();
JPanel p2 = new JPanel();
JPanel p3 = new JPanel();
JPanel p4 = new JPanel();
JPanel p5 = new JPanel();
JPanel p6 = new JPanel();


JLabel coment = new JLabel("* = case obligatoire");
JLabel prix = new JLabel("*Prix :");
JLabel dl = new JLabel("*Délai de livraison :");
JLabel autrescond = new JLabel("Autres conditions :");
JLabel cr = new JLabel("*Conditions de retour :");

JCheckBox cb1 = new JCheckBox("Par tranches");
JCheckBox cb2 = new JCheckBox("En une fois");

JButton bt_envoyer = new JButton("Envoyer");
JButton bt_annuler = new JButton("Annuler");
JButton bt_modifier = new JButton("Modifier");

JTextField pricefield = new JTextField(15);
JTextField dlfield = new JTextField(15);
JTextArea acTArea = new JTextArea(40, 15);
JTextArea crTArea = new JTextArea(40, 15);

//
public Offre()
        {
        AutresConditions = new Vector();
        ConditionRetour = new Vector();
        //
        coment.setFont(new java.awt.Font("Dialog", 1, 12));
        coment.setForeground(Color.blue);
        prix.setFont(new java.awt.Font("Dialog", 1, 14));
        prix.setForeground(Color.blue);
        dl.setFont(new java.awt.Font("Dialog", 1, 14));
        dl.setForeground(Color.blue);

        p1.setBorder(new TitledBorder(new EtchedBorder(),"*Mode de paiement"));
        //p2.setBorder(new EtchedBorder());
        p2.setBackground(SystemColor.info);
        cb1.setBackground(SystemColor.info);
```

```
cb1.setFont(new java.awt.Font("Dialog", 1, 14));
cb1.setForeground(Color.blue);
cb2.setBackground(SystemColor.info);
cb2.setFont(new java.awt.Font("Dialog", 1, 14));
cb2.setForeground(Color.blue);

bt_envoyer.setFont(new java.awt.Font("Dialog", 1, 14));
bt_envoyer.setForeground(Color.blue);
bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
bt_annuler.setForeground(Color.blue);
bt_modifier.setFont(new java.awt.Font("Dialog", 1, 14));
bt_modifier.setForeground(Color.blue);

bt_envoyer.addActionListener(new btEnvoyerListener());
bt_annuler.addActionListener(new btAnnulerListener());
bt_modifier.addActionListener(new btModifierListener());

cr.setFont(new java.awt.Font("Dialog", 1, 14));
cr.setForeground(Color.blue);
Component crTA = new JScrollPane(crTArea);
autrescond.setFont(new java.awt.Font("Dialog", 1, 14));
autrescond.setForeground(Color.blue);
Component acTA = new JScrollPane(acTArea);

setLayout(new GridBagLayout());
this.setBackground(SystemColor.info);
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.CENTER;

GridBagLayout gb1 = new GridBagLayout();
GridBagConstraints constraints1 = new GridBagConstraints();
p1.setLayout(gb1);
p1.setBackground(SystemColor.info);
constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(0, 0, 0, 5);
constraints1.gridwidth = 2;
constraints1.gridx = 0;
constraints1.gridy = 0;
gb1.setConstraints(cb1, constraints1);
constraints1.anchor = GridBagConstraints.EAST;
constraints1.insets = new Insets(0, 5, 0, 0);
constraints1.gridx = 1;
constraints1.gridy = 0;
gb1.setConstraints(cb1, constraints1);
p1.add(cb1);
p1.add(cb2);

GridBagLayout gb2 = new GridBagLayout();
GridBagConstraints constraints2 = new GridBagConstraints();
p2.setLayout(gb2);
constraints2.fill = GridBagConstraints.VERTICAL;
```

```
constraints2.anchor = GridBagConstraints.CENTER;
constraints2.insets = new Insets(5, 5, 5, 5);
constraints2.gridx = 0;
constraints2.gridy = 0;
gb2.setConstraints(bt_envoyer, constraints2);
constraints2.gridx = 1;
constraints2.gridy = 0;
gb2.setConstraints(bt_modifier, constraints2);
constraints2.anchor = GridBagConstraints.CENTER;
constraints2.gridwidth = 2;
constraints2.gridx = 2;
constraints2.gridy = 0;
gb2.setConstraints(bt_annuler, constraints2);
p2.add(bt_envoyer);
p2.add(bt_modifier);
p2.add(bt_annuler);

GridBagLayout gb3 = new GridBagLayout();
GridBagConstraints constraints3 = new GridBagConstraints();
p3.setLayout(gb3);
p3.setBackground(SystemColor.info);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(5, 15, 0, 5);
constraints3.gridx = 0;
constraints3.gridy = 0;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(prix, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(0, 15, 5, 5);
constraints3.gridx = 0;
constraints3.gridy = 1;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(pricefield, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(5, 5, 5, 0);
constraints3.gridx = 1;
constraints3.gridy = 0;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(dl, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(0, 5, 5, 0);
```

```
constraints3.gridx = 1;
constraints3.gridy = 1;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(dlfield, constraints3);

p3.add(prix);
p3.add(pricefield);
p3.add(dl);
p3.add(dlfield);

GridBagLayout gb5 = new GridBagLayout();
GridBagConstraints constraints5 = new GridBagConstraints();
p5.setLayout(gb5);
p5.setBackground(SystemColor.info);

constraints5.fill = GridBagConstraints.HORIZONTAL;
constraints5.anchor = GridBagConstraints.WEST;
constraints5.insets = new Insets(5, 5, 5, 5);
constraints5.gridx = 0;
constraints5.gridy = 0;
constraints5.weightx = 1;
constraints5.weighty = 20;
gb5.setConstraints(cr, constraints5);

constraints5.fill = GridBagConstraints.BOTH;
constraints5.anchor = GridBagConstraints.WEST;
constraints5.insets = new Insets(5, 5, 5, 5);
constraints5.gridx = 0;
constraints5.gridy = 1;
constraints5.weightx = 1;
constraints5.weighty = 80;
gb5.setConstraints(crTA, constraints5);

p5.add(cr);
p5.add(crTA);


GridBagLayout gb6 = new GridBagLayout();
GridBagConstraints constraints6 = new GridBagConstraints();
p6.setLayout(gb6);
p6.setBackground(SystemColor.info);

constraints6.fill = GridBagConstraints.HORIZONTAL;
constraints6.anchor = GridBagConstraints.WEST;
constraints6.insets = new Insets(5, 5, 5, 5);
constraints6.gridx = 0;
constraints6.gridy = 0;
constraints6.weightx = 1;
constraints6.weighty = 20;
gb6.setConstraints(autrescond, constraints6);
```

```
constraints6.fill = GridBagConstraints.BOTH;
constraints6.anchor = GridBagConstraints.WEST;
constraints6.insets = new Insets(5, 5, 5, 5);
constraints6.gridx = 0;
constraints6.gridy = 1;
constraints6.weightx = 1;
constraints6.weighty = 80;
gb6.setConstraints(acTA, constraints6);

p6.add(autrescond);
p6.add(acTA);

GridBagLayout gb4 = new GridBagLayout();
GridBagConstraints constraints4 = new GridBagConstraints();
p4.setLayout(gb4);
p4.setBackground(SystemColor.info);

constraints4.fill = GridBagConstraints.BOTH;
constraints4.anchor = GridBagConstraints.WEST;
constraints4.insets = new Insets(0, 5, 0, 0);
constraints4.gridx = 0;
constraints4.gridy = 0;
constraints4.weightx = 1;
constraints4.weighty = 1;
gb4.setConstraints(p5, constraints4);

constraints4.fill = GridBagConstraints.BOTH;
constraints4.anchor = GridBagConstraints.WEST;
constraints4.insets = new Insets(0, 5, 0, 0);
constraints4.gridx = 1;
constraints4.gridy = 0;
constraints4.weightx = 1;
constraints4.weighty = 1;
gb4.setConstraints(p6, constraints4);

p4.add(p5);
p4.add(p6);

int x, y;

constraints.weightx = 1.0;
constraints.weighty = 1.0;
constraints.insets = new Insets(5, 15, 15, 0);
constraints.fill = GridBagConstraints.NONE;
addGB(coment, x=0, y=3);

constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.CENTER;
constraints.insets = new Insets(5,5,5,5);
addGB(p1, x=0, y=2);
constraints.insets = new Insets(5,5,5,5);
addGB(p2, x=1, y=2);
```

```
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5,5,5,5);
        constraints.gridwidth = 2;
        constraints.gridheight = 1;
        addGB(p3, x=0, y=0);
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5,5,5,5);
        constraints.gridwidth = 2;
        constraints.gridheight = 1;
        addGB(p4, x=0, y=1);
        }

//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                add(component, constraints);
                }
//
public void DesactiverPrix()
        {
        pricefield.setEnabled(false);
        }

//
public void DesactiverDelai()
        {
        dlfield.setEnabled(false);
        }

//
public void DesactiverCondRet()
        {
        crTArea.setEnabled(false);
        }

//
public void DesactiverAutresCond()
        {
        acTArea.setEnabled(false);
        }

//
public void DesactiverCB1()
        {
        cb1.setEnabled(false);
        }
//
public void DesactiverCB2()
```

```
        {
        cb2.setEnabled(false);
        }
//
public void removebtPane()
        {
        remove(p2);
        }
//
public void removeComent()
        {
        remove(coment);
        }
//
public void modifp1()
        {
        GridBagLayout gb1 = new GridBagLayout();
        GridBagConstraints constraints1 = new GridBagConstraints();
        p1.setBorder(new TitledBorder(new EtchedBorder(),"*Mode de paiement"));
        p1.setLayout(gb1);
        p1.setBackground(SystemColor.info);
        cb1.setBackground(SystemColor.info);
        cb1.setFont(new java.awt.Font("Dialog", 1, 14));
        cb1.setForeground(Color.blue);
        cb2.setBackground(SystemColor.info);
        cb2.setFont(new java.awt.Font("Dialog", 1, 14));
        cb2.setForeground(Color.blue);
        constraints1.fill = GridBagConstraints.VERTICAL;
        constraints1.anchor = GridBagConstraints.CENTER;
        constraints1.insets = new Insets(0, 0, 0, 0);
        constraints1.gridwidth = 2;
        constraints1.gridx = 0;
        constraints1.gridy = 0;
        gb1.setConstraints(cb1, constraints1);
        constraints1.anchor = GridBagConstraints.WEST;
        constraints1.gridx = 1;
        constraints1.gridy = 0;
        gb1.setConstraints(cb1, constraints1);
        p1.add(cb1);
        p1.add(cb2);
        }
//
void setPrix(String p)
        {
        pricefield.setText(p);
        }
//
String getPrix()
        {
        return pricefield.getText();
        }
//
```

```
void setDelaiLivraison(String l)
        {
        dlfield.setText(l);
        }
//
String getDelaiLivraison()
        {
        return dlfield.getText();
        }
//
void setConditionRetour(String c)
        {
        crTArea.setText(c);
        }
//
String getConditionRetour()
        {
        return crTArea.getText();
        }
//

void setAutresCond(String c)
        {
        acTArea.setText(c);
        }
//
String getAutresCond()
        {
        return acTArea.getText();
        }
//
void setModePaiement(byte mp)
        {
        ModePaiement=mp;
        }
//
byte getModePaiement()
        {
        return ModePaiement;
        }
//
void addCondition(String s)
        {
        AutresConditions.addElement(s);
        }
//
Vector getAllConditions()
        {
        return AutresConditions;
        }
//
String getCondition(int i)
```

```
        {
        if (i<AutresConditions.size())
                return (String) (AutresConditions.elementAt(i));
        else return null;
        }
//
boolean delCondition(int i)
        {
        if (i<AutresConditions.size())
                {
                AutresConditions.removeElementAt(i);
                return true;
                }
        else return false;
        }
//
void delAllConditions()
        {
        AutresConditions.removeAllElements();
        }
//
public void DesactiverBtModifier()
        {
        bt_modifier.setEnabled(false);
        }
//
public class btEnvoyerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                if ((getPrix() == " ") & (getDelaiLivraison() == " ") &
                (getConditionRetour() == " "))
                JOptionPane.showMessageDialog(null, "Toutes les cases obligatoires ne
                sont pas remplies!");
                else
                        {
                        int result = JOptionPane.showConfirmDialog(null, "Etes-vous sûr
                        de vouloir envoyer cette offre?");
                        switch (result)
                                {
                                case JOptionPane.YES_OPTION:
                                JOptionPane.showMessageDialog(null, "Votre offre a été
                                enregistrée");
                                System.exit(0);break;
                                case JOptionPane.NO_OPTION: break;
                                case JOptionPane.CANCEL_OPTION: break;
                                case JOptionPane.CLOSED_OPTION: break;
                                }
                        }
                }
        }
//
```

```
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }
//
public class btModifierListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                if ((getPrix() != null) & (getDelaiLivraison() != null) &
                   (getConditionRetour() != null) | (getAutresCond() != null))
                        {
                        setPrix(" ");
                        setDelaiLivraison(" ");
                        setConditionRetour(" ");
                        setAutresCond(" ");
                        }
                else JOptionPane.showMessageDialog(null, "Il n'y a rien à modifier!");
                }
        }
//
public static void main(String[] args)
        {
        Offre offre = new Offre();
        Frame f = new Frame("Enregistrement d'une offre");
        f.addWindowListener(new WindowAdapter()
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
                });
        f.setSize(600,450);
        f.add("Center",offre);
        f.setLocation(50, 50);
        f.show();
        }
```

```
        }
```

```
//
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Offre extends JPanel
        {
        //attributs
        private int Prix;
        private String DelaiLivraison;
        private Vector ConditionRetour;
        /*
        ModePaiement=0 => paiement en une fois;
        ModePaiement=1 => paiment en tranches
        */
        private byte ModePaiement;
        private Vector AutresConditions;
        GridBagConstraints constraints = new GridBagConstraints();
        static Container content = new JPanel();
        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JPanel p3 = new JPanel();
        JPanel p4 = new JPanel();
        JPanel p5 = new JPanel();
        JPanel p6 = new JPanel();


        JLabel coment = new JLabel("* = case obligatoire");
        JLabel prix = new JLabel("*Prix :");
        JLabel dl = new JLabel("*Délai de livraison :");
        JLabel autrescond = new JLabel("Autres conditions :");
        JLabel cr = new JLabel("*Conditions de retour :");

        JCheckBox cb1 = new JCheckBox("Par tranches");
        JCheckBox cb2 = new JCheckBox("En une fois");

        JButton bt_envoyer = new JButton("Envoyer");
        JButton bt_annuler = new JButton("Annuler");
        JButton bt_modifier = new JButton("Modifier");

        JTextField pricefield = new JTextField(15);
        JTextField dlfield = new JTextField(15);
        JTextArea acTArea = new JTextArea(40, 15);
        JTextArea crTArea = new JTextArea(40, 15);

        //
        public Offre()
```

```
{
AutresConditions = new Vector();
ConditionRetour = new Vector();
//
coment.setFont(new java.awt.Font("Dialog", 1, 12));
coment.setForeground(Color.blue);
prix.setFont(new java.awt.Font("Dialog", 1, 14));
prix.setForeground(Color.blue);
dl.setFont(new java.awt.Font("Dialog", 1, 14));
dl.setForeground(Color.blue);

p1.setBorder(new TitledBorder(new EtchedBorder(),"*Mode de paiement"));
//p2.setBorder(new EtchedBorder());
p2.setBackground(SystemColor.info);
cb1.setBackground(SystemColor.info);
cb1.setFont(new java.awt.Font("Dialog", 1, 14));
cb1.setForeground(Color.blue);
cb2.setBackground(SystemColor.info);
cb2.setFont(new java.awt.Font("Dialog", 1, 14));
cb2.setForeground(Color.blue);

bt_envoyer.setFont(new java.awt.Font("Dialog", 1, 14));
bt_envoyer.setForeground(Color.blue);
bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
bt_annuler.setForeground(Color.blue);
bt_modifier.setFont(new java.awt.Font("Dialog", 1, 14));
bt_modifier.setForeground(Color.blue);

bt_envoyer.addActionListener(new btEnvoyerListener());
bt_annuler.addActionListener(new btAnnulerListener());
bt_modifier.addActionListener(new btModifierListener());

cr.setFont(new java.awt.Font("Dialog", 1, 14));
cr.setForeground(Color.blue);
Component crTA = new JScrollPane(crTArea);
autrescond.setFont(new java.awt.Font("Dialog", 1, 14));
autrescond.setForeground(Color.blue);
Component acTA = new JScrollPane(acTArea);

setLayout(new GridBagLayout());
this.setBackground(SystemColor.info);
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.CENTER;

GridBagLayout gb1 = new GridBagLayout();
GridBagConstraints constraints1 = new GridBagConstraints();
p1.setLayout(gb1);
p1.setBackground(SystemColor.info);
constraints1.fill = GridBagConstraints.HORIZONTAL;
constraints1.anchor = GridBagConstraints.WEST;
constraints1.insets = new Insets(0, 0, 0, 5);
constraints1.gridwidth = 2;
```

```
constraints1.gridx = 0;
constraints1.gridy = 0;
gb1.setConstraints(cb1, constraints1);
constraints1.anchor = GridBagConstraints.EAST;
constraints1.insets = new Insets(0, 5, 0, 0);
constraints1.gridx = 1;
constraints1.gridy = 0;
gb1.setConstraints(cb1, constraints1);
p1.add(cb1);
p1.add(cb2);

GridBagLayout gb2 = new GridBagLayout();
GridBagConstraints constraints2 = new GridBagConstraints();
p2.setLayout(gb2);
constraints2.fill = GridBagConstraints.VERTICAL;
constraints2.anchor = GridBagConstraints.CENTER;
constraints2.insets = new Insets(5, 5, 5, 5);
constraints2.gridx = 0;
constraints2.gridy = 0;
gb2.setConstraints(bt_envoyer, constraints2);
constraints2.gridx = 1;
constraints2.gridy = 0;
gb2.setConstraints(bt_modifier, constraints2);
constraints2.anchor = GridBagConstraints.CENTER;
constraints2.gridwidth = 2;
constraints2.gridx = 2;
constraints2.gridy = 0;
gb2.setConstraints(bt_annuler, constraints2);
p2.add(bt_envoyer);
p2.add(bt_modifier);
p2.add(bt_annuler);

GridBagLayout gb3 = new GridBagLayout();
GridBagConstraints constraints3 = new GridBagConstraints();
p3.setLayout(gb3);
p3.setBackground(SystemColor.info);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(5, 15, 0, 5);
constraints3.gridx = 0;
constraints3.gridy = 0;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(prix, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(0, 15, 5, 5);
constraints3.gridx = 0;
constraints3.gridy = 1;
constraints3.weightx = 20;
```

```
constraints3.weighty = 1;
gb3.setConstraints(pricefield, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(5, 5, 5, 0);
constraints3.gridx = 1;
constraints3.gridy = 0;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(dl, constraints3);

constraints3.fill = GridBagConstraints.HORIZONTAL;
constraints3.anchor = GridBagConstraints.WEST;
constraints3.insets = new Insets(0, 5, 5, 0);
constraints3.gridx = 1;
constraints3.gridy = 1;
constraints3.weightx = 20;
constraints3.weighty = 1;
gb3.setConstraints(dlfield, constraints3);

p3.add(prix);
p3.add(pricefield);
p3.add(dl);
p3.add(dlfield);

GridBagLayout gb5 = new GridBagLayout();
GridBagConstraints constraints5 = new GridBagConstraints();
p5.setLayout(gb5);
p5.setBackground(SystemColor.info);

constraints5.fill = GridBagConstraints.HORIZONTAL;
constraints5.anchor = GridBagConstraints.WEST;
constraints5.insets = new Insets(5, 5, 5, 5);
constraints5.gridx = 0;
constraints5.gridy = 0;
constraints5.weightx = 1;
constraints5.weighty = 20;
gb5.setConstraints(cr, constraints5);

constraints5.fill = GridBagConstraints.BOTH;
constraints5.anchor = GridBagConstraints.WEST;
constraints5.insets = new Insets(5, 5, 5, 5);
constraints5.gridx = 0;
constraints5.gridy = 1;
constraints5.weightx = 1;
constraints5.weighty = 80;
gb5.setConstraints(crTA, constraints5);

p5.add(cr);
p5.add(crTA);
```

```
GridBagLayout gb6 = new GridBagLayout();
GridBagConstraints constraints6 = new GridBagConstraints();
p6.setLayout(gb6);
p6.setBackground(SystemColor.info);

constraints6.fill = GridBagConstraints.HORIZONTAL;
constraints6.anchor = GridBagConstraints.WEST;
constraints6.insets = new Insets(5, 5, 5, 5);
constraints6.gridx = 0;
constraints6.gridy = 0;
constraints6.weightx = 1;
constraints6.weighty = 20;
gb6.setConstraints(autrescond, constraints6);

constraints6.fill = GridBagConstraints.BOTH;
constraints6.anchor = GridBagConstraints.WEST;
constraints6.insets = new Insets(5, 5, 5, 5);
constraints6.gridx = 0;
constraints6.gridy = 1;
constraints6.weightx = 1;
constraints6.weighty = 80;
gb6.setConstraints(acTA, constraints6);

p6.add(autrescond);
p6.add(acTA);

GridBagLayout gb4 = new GridBagLayout();
GridBagConstraints constraints4 = new GridBagConstraints();
p4.setLayout(gb4);
p4.setBackground(SystemColor.info);

constraints4.fill = GridBagConstraints.BOTH;
constraints4.anchor = GridBagConstraints.WEST;
constraints4.insets = new Insets(0, 5, 0, 0);
constraints4.gridx = 0;
constraints4.gridy = 0;
constraints4.weightx = 1;
constraints4.weighty = 1;
gb4.setConstraints(p5, constraints4);

constraints4.fill = GridBagConstraints.BOTH;
constraints4.anchor = GridBagConstraints.WEST;
constraints4.insets = new Insets(0, 5, 0, 0);
constraints4.gridx = 1;
constraints4.gridy = 0;
constraints4.weightx = 1;
constraints4.weighty = 1;
gb4.setConstraints(p6, constraints4);

p4.add(p5);
p4.add(p6);
```

```
        int x, y;

        constraints.weightx = 1.0;
        constraints.weighty = 1.0;
        constraints.insets = new Insets(5, 15, 15, 0);
        constraints.fill = GridBagConstraints.NONE;
        addGB(coment, x=0, y=3);

        constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5,5,5,5);
        addGB(p1, x=0, y=2);
        constraints.insets = new Insets(5,5,5,5);
        addGB(p2, x=1, y=2);
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5,5,5,5);
        constraints.gridwidth = 2;
        constraints.gridheight = 1;
        addGB(p3, x=0, y=0);
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5,5,5,5);
        constraints.gridwidth = 2;
        constraints.gridheight = 1;
        addGB(p4, x=0, y=1);
        }

//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                add(component, constraints);
                }
//
public void DesactiverPrix()
        {
        pricefield.setEnabled(false);
        }

//
public void DesactiverDelai()
        {
        dlfield.setEnabled(false);
        }

//
public void DesactiverCondRet()
        {
        crTArea.setEnabled(false);
```

```
        }

//
public void DesactiverAutresCond()
        {
        acTArea.setEnabled(false);
        }

//
public void DesactiverCB1()
        {
        cb1.setEnabled(false);
        }
//
public void DesactiverCB2()
        {
        cb2.setEnabled(false);
        }
//
public void removebtPane()
        {
        remove(p2);
        }
//
public void removeComent()
        {
        remove(coment);
        }
//
public void modifp1()
        {
        GridBagLayout gb1 = new GridBagLayout();
        GridBagConstraints constraints1 = new GridBagConstraints();
        p1.setBorder(new TitledBorder(new EtchedBorder(),"*Mode de paiement"));
        p1.setLayout(gb1);
        p1.setBackground(SystemColor.info);
        cb1.setBackground(SystemColor.info);
        cb1.setFont(new java.awt.Font("Dialog", 1, 14));
        cb1.setForeground(Color.blue);
        cb2.setBackground(SystemColor.info);
        cb2.setFont(new java.awt.Font("Dialog", 1, 14));
        cb2.setForeground(Color.blue);
        constraints1.fill = GridBagConstraints.VERTICAL;
        constraints1.anchor = GridBagConstraints.CENTER;
        constraints1.insets = new Insets(0, 0, 0, 0);
        constraints1.gridwidth = 2;
        constraints1.gridx = 0;
        constraints1.gridy = 0;
        gb1.setConstraints(cb1, constraints1);
        constraints1.anchor = GridBagConstraints.WEST;
        constraints1.gridx = 1;
        constraints1.gridy = 0;
```

```
        gb1.setConstraints(cb1, constraints1);
        p1.add(cb1);
        p1.add(cb2);
        }
//
void setPrix(String p)
        {
        pricefield.setText(p);
        }
//
String getPrix()
        {
        return pricefield.getText();
        }
//
void setDelaiLivraison(String l)
        {
        dlfield.setText(l);
        }
//
String getDelaiLivraison()
        {
        return dlfield.getText();
        }
//
void setConditionRetour(String c)
        {
        crTArea.setText(c);
        }
//
String getConditionRetour()
        {
        return crTArea.getText();
        }
//

void setAutresCond(String c)
        {
        acTArea.setText(c);
        }
//
String getAutresCond()
        {
        return acTArea.getText();
        }
//
void setModePaiement(byte mp)
        {
        ModePaiement=mp;
        }
//
byte getModePaiement()
```

```
        {
        return ModePaiement;
        }
//
void addCondition(String s)
        {
        AutresConditions.addElement(s);
        }
//
Vector getAllConditions()
        {
        return AutresConditions;
        }
//
String getCondition(int i)
        {
        if (i<AutresConditions.size())
                return (String) (AutresConditions.elementAt(i));
        else return null;
        }
//
boolean delCondition(int i)
        {
        if (i<AutresConditions.size())
                {
                AutresConditions.removeElementAt(i);
                return true;
                }
        else return false;
        }
//
void delAllConditions()
        {
        AutresConditions.removeAllElements();
        }
//
public void DesactiverBtModifier()
        {
        bt_modifier.setEnabled(false);
        }
//
public class btEnvoyerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                        if ((getPrix() == " ") & (getDelaiLivraison() == " ") &
                        (getConditionRetour() == " "))
                        JOptionPane.showMessageDialog(null, "Toutes les cases
                        obligatoires ne sont pas remplies!");
                        else
                        {
```

```
                    int result = JOptionPane.showConfirmDialog(null, "Etes-vous sûr
                    de vouloir envoyer cette offre?");
                    switch (result)
                        {
                        case JOptionPane.YES_OPTION:
                        JOptionPane.showMessageDialog(null, "Votre offre a été
                        enregistrée");
                        System.exit(0);break;
                        case JOptionPane.NO_OPTION: break;
                        case JOptionPane.CANCEL_OPTION: break;
                        case JOptionPane.CLOSED_OPTION: break;
                        }
                    }
                }
        }
//
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }
//
public class btModifierListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                if ((getPrix() != null) & (getDelaiLivraison() != null) &
                   (getConditionRetour() != null) | (getAutresCond() != null))
                        {
                        setPrix(" ");
                        setDelaiLivraison(" ");
                        setConditionRetour(" ");
                        setAutresCond(" ");
                        }
                else JOptionPane.showMessageDialog(null, "Il n'y a rien à modifier!");
                }
        }
//
```

```
public static void main(String[] args)
        {
        Offre offre = new Offre();
        Frame f = new Frame("Enregistrement d'une offre");
        f.addWindowListener(new WindowAdapter()
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
                });
        f.setSize(600,450);
        f.add("Center",offre);
        f.setLocation(50, 50);
        f.show();
        }
}
```

```
//Affichage du formulaire de demande d'information

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;


class Message extends JPanel
        {
        //attributs
        private String EmailDest;
        private String EmailExp;
        private Date DateEnvoie;
        private String Heure;

        GridBagConstraints constraints = new GridBagConstraints();
        //
        public Message()
                {
                this.setLayout(new GridBagLayout());
                this.setBackground(SystemColor.info);
                constraints.fill = GridBagConstraints.VERTICAL;
                constraints.anchor = GridBagConstraints.EAST;
                constraints.insets = new Insets(5, 15, 0, 5);
                constraints.weightx = 1.0;
                constraints.weighty = 1.0;
                constraints.gridheight = 1;
                constraints.gridwidth = 1;
                int x, y;
                constraints.weightx = 0.3;
                JLabel label1 = new JLabel("Adresse destinataire :");
```

```
label1.setFont(new java.awt.Font("Dialog", 1, 14));
label1.setForeground(Color.blue);
addGB(label1, x=0, y=0);
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(5, 5, 0, 15);
constraints.weightx = 1.0;
addGB(new JTextField(15), x=1, y=0);
constraints.weightx = 0.3;
constraints.fill = GridBagConstraints.VERTICAL;
constraints.anchor = GridBagConstraints.EAST;
constraints.insets = new Insets(0, 15, 0, 5);
JLabel label2 = new JLabel("Adresse expéditeur :");
label2.setFont(new java.awt.Font("Dialog", 1, 14));
label2.setForeground(Color.blue);
addGB(label2, x=0, y=1);
constraints.weightx = 1.0;
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(0, 5, 0, 15);
addGB(new JTextField(15), x=1, y=1);
JLabel label3 = new JLabel("Rappel du contexte de la négociation");
label3.setFont(new java.awt.Font("Dialog", 1, 14));
label3.setForeground(Color.blue);
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(0, 15, 0, 0);
addGB(label3, x=0, y=2);
JTextArea contexte = new JTextArea(50, 5);
Component contexteSP = new JScrollPane(contexte);
constraints.fill = GridBagConstraints.BOTH;
constraints.anchor = GridBagConstraints.CENTER;
constraints.insets = new Insets(0, 15, 0, 15);
constraints.weightx = 1.0;
constraints.weighty = 1.0;
constraints.gridheight = 3;
constraints.gridwidth = 2;
addGB(contexteSP, x=0, y=3);
constraints.gridheight = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(0, 15, 0, 0);
JLabel label4 = new JLabel("Contenu du message à envoyer");
label4.setFont(new java.awt.Font("Dialog", 1, 14));
label4.setForeground(Color.blue);
addGB(label4, x=0, y=8);
JTextArea contenu = new JTextArea(50, 30);
Component contenuSP = new JScrollPane(contenu);
constraints.gridheight = 5;
constraints.fill = GridBagConstraints.BOTH;
constraints.anchor = GridBagConstraints.CENTER;
constraints.insets = new Insets(0, 15, 15, 15);
```

```
        addGB(contenuSP, x=0, y=10);

        constraints.fill = GridBagConstraints.NONE;
        constraints.anchor = GridBagConstraints.EAST;
        constraints.insets = new Insets(15, 0, 15, 15);
        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        JButton btEnvoyer = new JButton("Envoyer");
        btEnvoyer.addActionListener(new btEnvoyerListener());
        btEnvoyer.setFont(new java.awt.Font("Dialog", 1, 14));
        btEnvoyer.setForeground(Color.blue);
        addGB(btEnvoyer, x=0, y=18);
        constraints.anchor = GridBagConstraints.WEST;
        JButton btAnnuler = new JButton("Annuler");
        btAnnuler.addActionListener(new btAnnulerListener());
        btAnnuler.setFont(new java.awt.Font("Dialog", 1, 14));
        btAnnuler.setForeground(Color.blue);
        constraints.insets = new Insets(15, 0, 15, 15);
        addGB(btAnnuler, x=1, y=18);

        }
//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                add(component, constraints);
                }
//
void setEmailDest(String emd)
        {
        EmailDest=emd;
        }
//
String getEmailDest()
        {
        return EmailDest;
        }
//
void setEmailExp(String emex)
        {
        EmailExp=emex;
        }
//
String getEmailExp()
        {
        return EmailExp;
        }
//
void setDateEnvoie(Date d)
        {
        DateEnvoie=d;
```

```
        }
//
Date getDateEnvoie()
        {
        return DateEnvoie;
        }
//
void setHeure(String h)
        {
        Heure=h;
        }
//
String getHeure()
        {
        return Heure;
        }

//
public class btEnvoyerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                JOptionPane.showMessageDialog(null, "Votre message a été envoyé");
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }

//
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
```

```
                    });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }

    //
    public static void main(String[] args)
        {
        Message info = new Message();
        Frame f = new Frame("Demande d'information");
        f.addWindowListener(new WindowAdapter()
            {
            public void windowClosing(WindowEvent we)
                {
                System.exit(0);
                }
            });
        f.setSize(500,500);
        f.add("Center",info);
        f.setLocation(150, 30);
        f.setVisible(true);
        }
    }
```

```
//Affichage du formulaire de l'accord

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

//
class Accord extends JFrame
    {
    //attributs
    private Offre offre = new Offre();
    private String Message;
    GridBagConstraints constraints = new GridBagConstraints();
    static Container content = new JPanel();

    JTextArea MTArea = new JTextArea(50, 15);
    //
    public Accord()
        {
        super("Acceptation d'une offre");
        content.setLayout(new GridBagLayout());
        content.setBackground(SystemColor.info);
```

```
int x, y;
JPanel p1 = new JPanel();
p1.setBorder(new EtchedBorder());
JButton bt_envoyer = new JButton("Envoyer");
bt_envoyer.setFont(new java.awt.Font("Dialog", 1, 14));
bt_envoyer.setForeground(Color.blue);
bt_envoyer.addActionListener(new btEnvoyerListener());
JButton bt_annuler = new JButton("Annuler");
bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
bt_annuler.setForeground(Color.blue);
bt_annuler.addActionListener(new btAnnulerListener());
GridBagLayout gb1 = new GridBagLayout();
GridBagConstraints constraints1 = new GridBagConstraints();
p1.setLayout(gb1);
p1.setBackground(SystemColor.info);
constraints1.fill = GridBagConstraints.NONE;
constraints1.anchor = GridBagConstraints.CENTER;
constraints1.insets = new Insets(5, 15, 5, 5);
constraints1.gridx = 0;
constraints1.gridy = 0;
gb1.setConstraints(bt_envoyer, constraints1);
constraints1.gridwidth = 1;
constraints1.gridx = 1;
constraints1.gridy = 0;
gb1.setConstraints(bt_annuler, constraints1);
p1.add(bt_envoyer);
p1.add(bt_annuler);
constraints.fill = GridBagConstraints.NONE;
constraints.insets = new Insets(5, 5, 5, 5);
constraints.weightx = 1;
constraints.weighty = 1;
constraints.gridheight = 1;
constraints.gridwidth = 1;
addGB(p1, x=0, y=15);

JLabel message = new JLabel("Veuillez introduire ci-dessous votre message");
JLabel message2 = new JLabel("de l'accord portant sur cette offre");
message.setFont(new java.awt.Font("Dialog", 1, 14));
message.setForeground(Color.blue);

message2.setFont(new java.awt.Font("Dialog", 1, 14));
message2.setForeground(Color.blue);

JLabel message1 = new JLabel("Pour un rappel voici l'offre sur laquelle porte
votre accord");
message1.setFont(new java.awt.Font("Dialog", 1, 14));
message1.setForeground(Color.blue);
Component MTA = new JScrollPane(MTArea);

constraints.gridheight = 1;
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
```

```
        constraints.anchor = GridBagConstraints.WEST;
        constraints.insets = new Insets(5, 20, 0, 20);
        addGB(message, x=0, y=10);

        constraints.weightx = 1;
        constraints.weighty = 1;
        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5, 20, 15, 20);
        addGB(MTA, x=0, y=12);

        constraints.weightx = 1;
        constraints.weighty = 1;
        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(15, 15, 15, 15);
        offre.removebtPane();
        offre.removeComent();
        offre.modifp1();
        addGB(offre, x=0, y=1);

        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        constraints.fill = GridBagConstraints.NONE;
        constraints.anchor = GridBagConstraints.WEST;
        constraints.insets = new Insets(5, 20, 0, 20);
        addGB(message1, x=0, y=0);

        constraints.fill = GridBagConstraints.NONE;
        constraints.anchor = GridBagConstraints.WEST;
        constraints.insets = new Insets(0, 20, 0, 20);
        addGB(message2, x=0, y=11);
        }

//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                content.add(component, constraints);
                }

//
void setMessage(String m)
        {
        MTArea.setText(m);
        }
//
```

```
String getMessage()
        {
        return MTArea.getText();
        }

//
public class btEnvoyerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                if (getMessage() != "")
                        {int result = JOptionPane.showConfirmDialog(null, "Etes-vous sûr
                        de vouloir accepter cette offre?");
                        switch (result)
                                {
                                case JOptionPane.YES_OPTION:
                                JOptionPane.showMessageDialog(null, "Votre accord sur
                                cette offre a été enregistrée");
                                 System.exit(0);break;
                                case JOptionPane.NO_OPTION: break;
                                case JOptionPane.CANCEL_OPTION: break;
                                case JOptionPane.CLOSED_OPTION: break;
                                }
                        }
                else JOptionPane.showMessageDialog(null, "Vous n'avez pas introduit
                votre message d'accord!");

                }
        }

//
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(400,400);
                f.setLocation(200, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }

//
```

```
    public static void main(String[] args)
        {
        JFrame f = new Accord();
        f.addWindowListener(new WindowAdapter()
            {
            public void windowClosing(WindowEvent we)
                {
                System.exit(0);
                }
            });
        f.setSize(500,500);
        f.setContentPane(content);
        f.setLocation(30, 30);
        f.setVisible(true);
        }
    }
```

```
//Affichage de la confirmation d'une offre

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

//
class ConfOffre extends JFrame
    {
    //attributs
    private Offre offre = new Offre();
    private String Message;
    GridBagConstraints constraints = new GridBagConstraints();
    static Container content = new JPanel();


    JTextArea MTArea = new JTextArea(50, 15);
    //
    public ConfOffre()
        {
        super("Confirmation d'une offre");
        content.setLayout(new GridBagLayout());
        content.setBackground(SystemColor.info);
        int x, y;
        JPanel p1 = new JPanel();
        p1.setBorder(new EtchedBorder());
        JButton bt_envoyer = new JButton("Envoyer");
        bt_envoyer.setFont(new java.awt.Font("Dialog", 1, 14));
        bt_envoyer.setForeground(Color.blue);
        bt_envoyer.addActionListener(new btEnvoyerListener());
        JButton bt_annuler = new JButton("Annuler");
        bt_annuler.setFont(new java.awt.Font("Dialog", 1, 14));
```

```
bt_annuler.setForeground(Color.blue);
bt_annuler.addActionListener(new btAnnulerListener());
GridBagLayout gb1 = new GridBagLayout();
GridBagConstraints constraints1 = new GridBagConstraints();
p1.setLayout(gb1);
p1.setBackground(SystemColor.info);
constraints1.fill = GridBagConstraints.NONE;
constraints1.anchor = GridBagConstraints.CENTER;
constraints1.insets = new Insets(5, 15, 5, 5);
constraints1.gridx = 0;
constraints1.gridy = 0;
gb1.setConstraints(bt_envoyer, constraints1);
constraints1.gridwidth = 2;
constraints1.gridx = 1;
constraints1.gridy = 0;
gb1.setConstraints(bt_annuler, constraints1);
p1.add(bt_envoyer);
p1.add(bt_annuler);
constraints.fill = GridBagConstraints.NONE;
constraints.insets = new Insets(5, 5, 5, 5);
constraints.weightx = 1;
constraints.weighty = 1;
constraints.gridheight = 1;
constraints.gridwidth = 4;
addGB(p1, x=0, y=15);

JLabel message = new JLabel("Veuillez introduire ci-dessous un message de
confirmantion");
JLabel message2 = new JLabel("de cette offre pour conclure la négociation");
message.setFont(new java.awt.Font("Dialog", 1, 14));
message.setForeground(Color.blue);
message2.setFont(new java.awt.Font("Dialog", 1, 14));
message2.setForeground(Color.blue);
JLabel message1 = new JLabel("Voici l'offre ayant reçu l'accord de votre
partenaire");
message1.setFont(new java.awt.Font("Dialog", 1, 14));
message1.setForeground(Color.blue);
Component MTA = new JScrollPane(MTArea);

constraints.gridheight = 1;
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.anchor = GridBagConstraints.WEST;
constraints.insets = new Insets(5, 20, 0, 20);
addGB(message, x=0, y=10);

constraints.weightx = 1;
constraints.weighty = 1;
constraints.gridheight = 1;
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.BOTH;
constraints.anchor = GridBagConstraints.CENTER;
```

```
        constraints.insets = new Insets(5, 20, 15, 20);
        addGB(MTA, x=0, y=12);

        constraints.weightx = 1;
        constraints.weighty = 1;
        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        constraints.fill = GridBagConstraints.BOTH;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(15, 15, 15, 15);
        offre.removebtPane();
        offre.removeComent();
        offre.modifp1();
        addGB(offre, x=0, y=1);

        constraints.gridheight = 1;
        constraints.gridwidth = 1;
        constraints.fill = GridBagConstraints.NONE;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(5, 20, 0, 20);
        addGB(message1, x=0, y=0);

        constraints.fill = GridBagConstraints.HORIZONTAL;
        constraints.anchor = GridBagConstraints.WEST;
        constraints.insets = new Insets(0, 20, 0, 20);
        addGB(message2, x=0, y=11);
        }

//
void addGB(Component component, int x, int y)
                {
                constraints.gridx = x;
                constraints.gridy = y;
                content.add(component, constraints);
                }

//
void setMessage(String m)
        {
        MTArea.setText(m);
        }
//
String getMessage()
        {
        return MTArea.getText();
        }

//
public class btEnvoyerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
```

```
                if (getMessage() != "")
                        {int result = JOptionPane.showConfirmDialog(null, "Etes-vous sûr
                        de vouloir confirmer cette offre et clôturer la négociation?");
                        switch (result)
                                {
                                case JOptionPane.YES_OPTION:
                                JOptionPane.showMessageDialog(null, "Votre
                                conformation a été enregistrée. La négociation est terminée.
                                Merci, Aurevoir!");
                                System.exit(0);break;
                                case JOptionPane.NO_OPTION: break;
                                case JOptionPane.CANCEL_OPTION: break;
                                case JOptionPane.CLOSED_OPTION: break;
                                }
                        }
                else JOptionPane.showMessageDialog(null, "Vous n'avez pas introduit
                votre message de confirmation!");

                }
        }

//
public class btAnnulerListener implements ActionListener
        {
        public void actionPerformed(ActionEvent e)
                {
                Action action = new Action();
                Frame f = new Frame("Choix de l'action à effectuer ");
                f.addWindowListener(new WindowAdapter()
                        {
                        public void windowClosing(WindowEvent we)
                                {
                                System.exit(0);
                                }
                        });
                f.setSize(300,400);
                f.setLocation(100, 50);
                f.add("Center", action);
                f.setVisible(true);
                }
        }

//
public static void main(String[] args)
        {
        JFrame f = new ConfOffre();
        f.addWindowListener(new WindowAdapter()
                {
                public void windowClosing(WindowEvent we)
                        {
                        System.exit(0);
                        }
```

```
                });
            f.setSize(500,500);
            f.setContentPane(content);
            f.setLocation(30, 30);
            f.setVisible(true);
            }
    }
```

```
//Affichage de l'historique

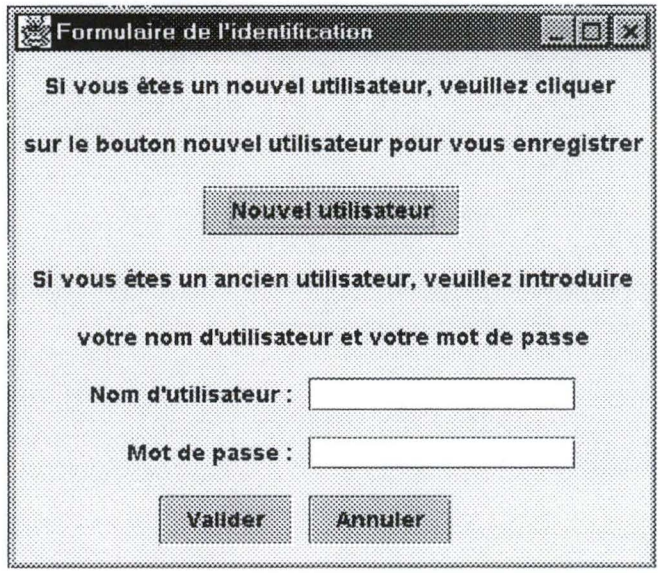

class Historique
    {
    //attributs
    private Vector offre;
    private Vector message;
    private Vector accord;
    private Vector confOffre;
    //
    Historique()
        {
        offre=new Vector();
        message=new Vector();
        accord=new Vector();
        confOffre=new Vector();
        }
    void addOffre(Offre o)
        {
        offre.addElement(o);
        }
    //
    void addMessage(Message m)
        {
        message.addElement(m);
        }
    //
    void addAccord(Accord a)
        {
        accord.addElement(a);
        }
    //
    void addConfOffre(ConfOffre co)
        {
        confOffre.addElement(co);
        }
    //
    Vector getAllOffres()
        {
        return offre;
        }
    //
```

```
Vector getAllMessages()
        {
        return message;
        }
//
Vector getAllAccords()
        {
        return accord;
        }
//
Vector getAllConfOffre()
        {
        return confOffre;
        }
//
String getOffre(int i)
        {
        if (i<offre.size())
                return (String) (offre.elementAt(i));
        else return null;
        }
//
String getMessage(int i)
        {
        if (i<message.size())
                return (String) (message.elementAt(i));
        else return null;
        }
//
String getAccord(int i)
        {
        if (i<accord.size())
                return (String) (accord.elementAt(i));
        else return null;
        }
//
String getConfOffre(int i)
        {
        if (i<confOffre.size())
                return (String) (confOffre.elementAt(i));
        else return null;
        }
}
```

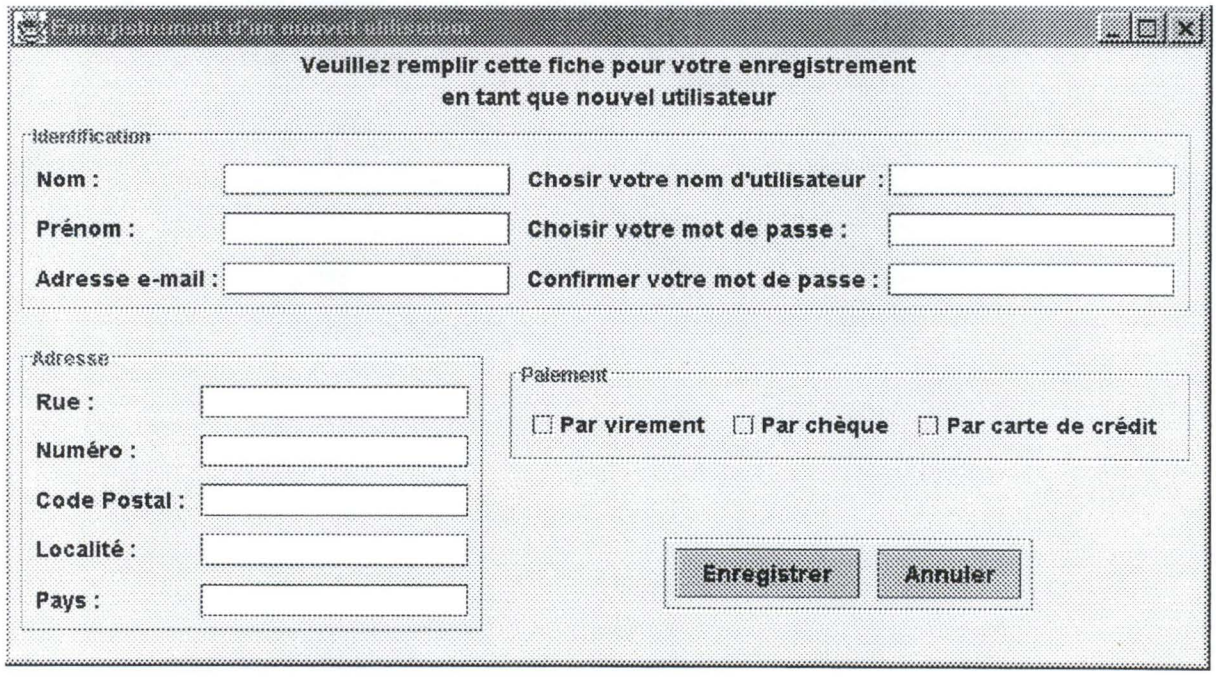
==========

5. Présentation de l'interface du prototype

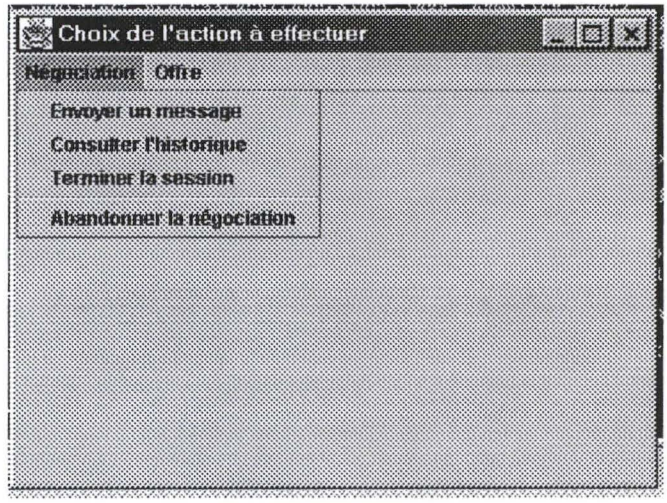
Formulaire d'identification d'un utilisateur

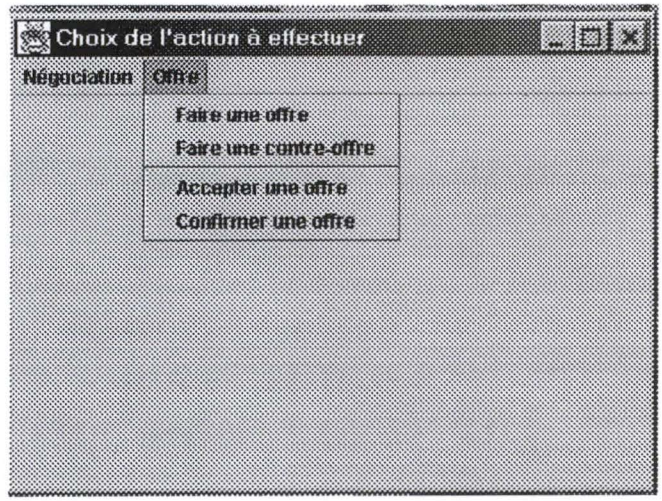


Formulaire d'identification d'un nouvel utilisateur

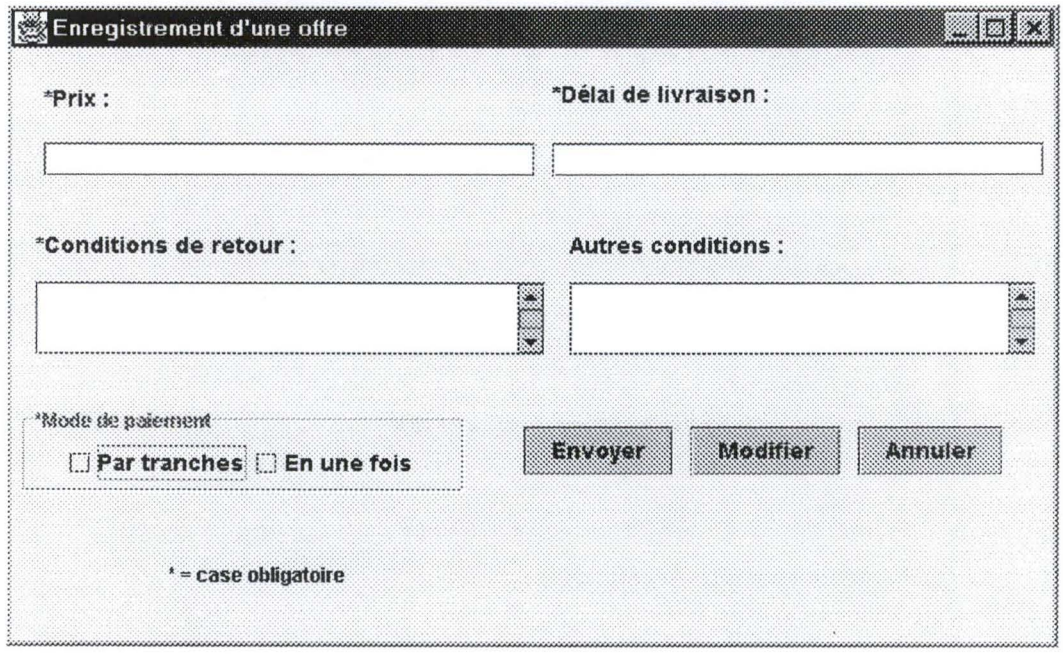


Menu pour le choix d'un dialogue

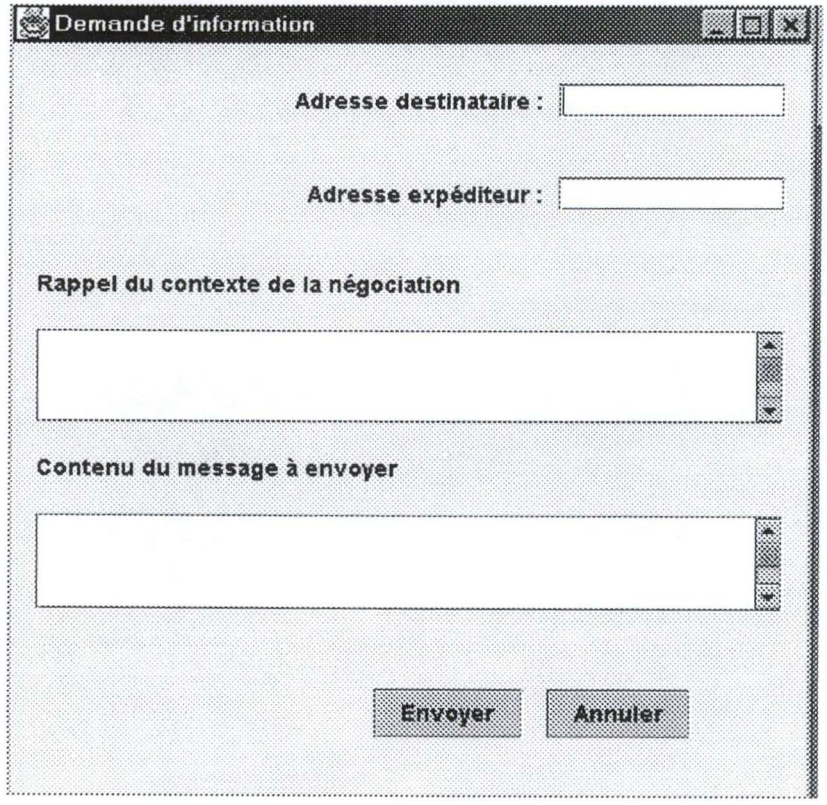




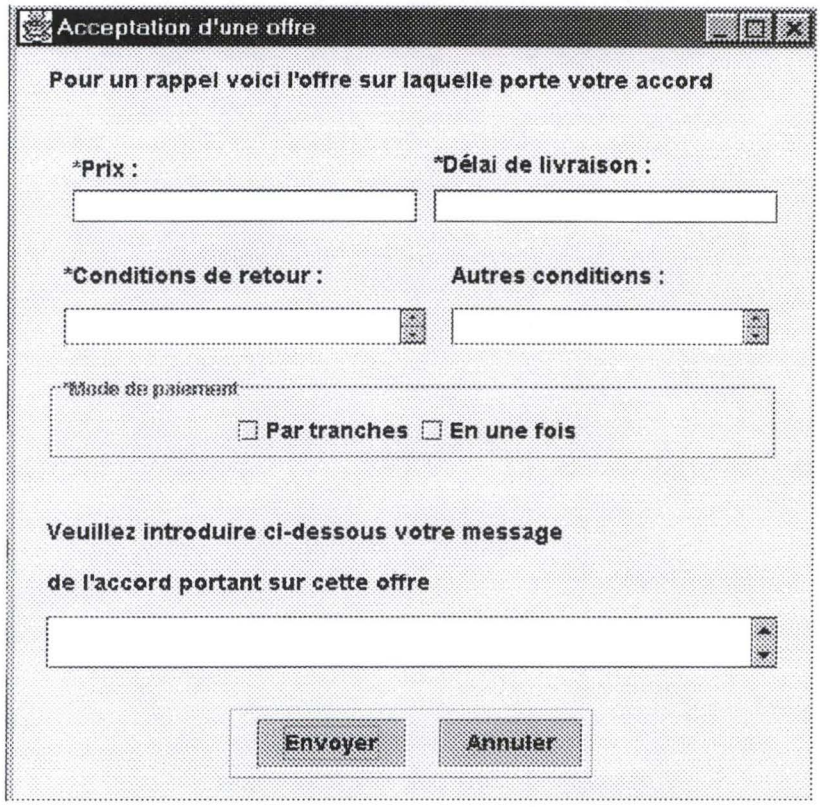
Formulaire de l'offre (et contre-offre)



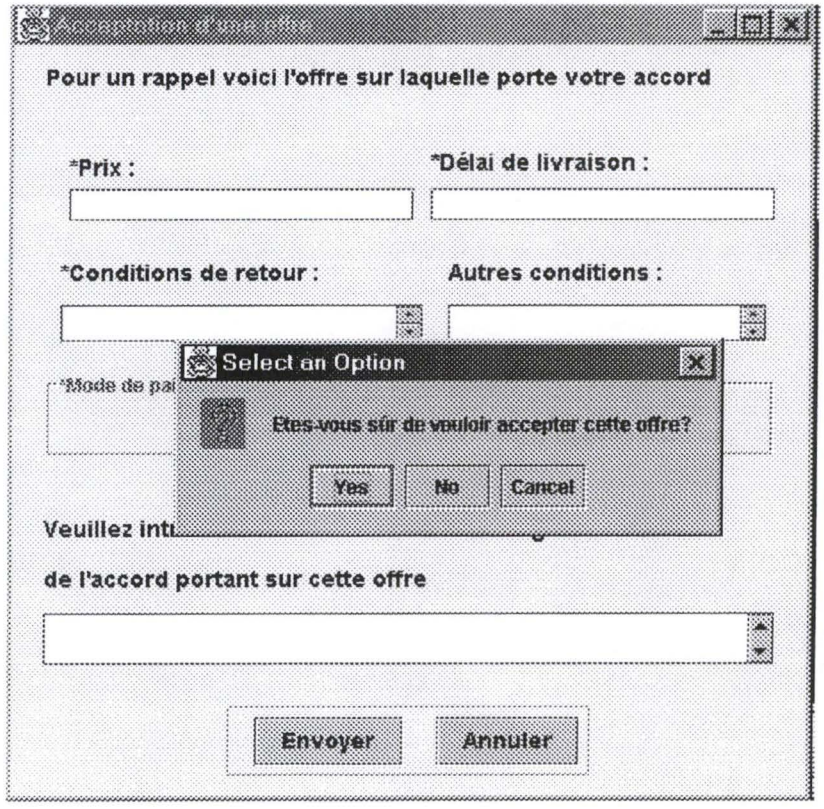
Formulaire demande d'information



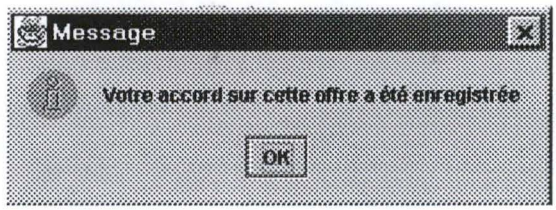
Formulaire de l'accord sur une offre



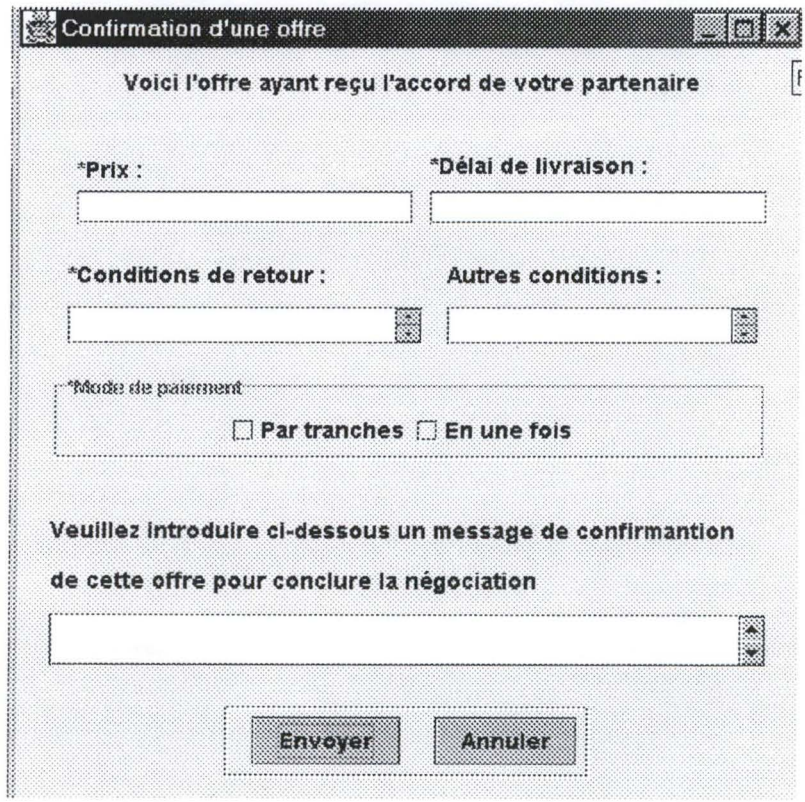
Message reçu par l'utilisateur lorsqu'il appuie sur le bouton "Envoyer"



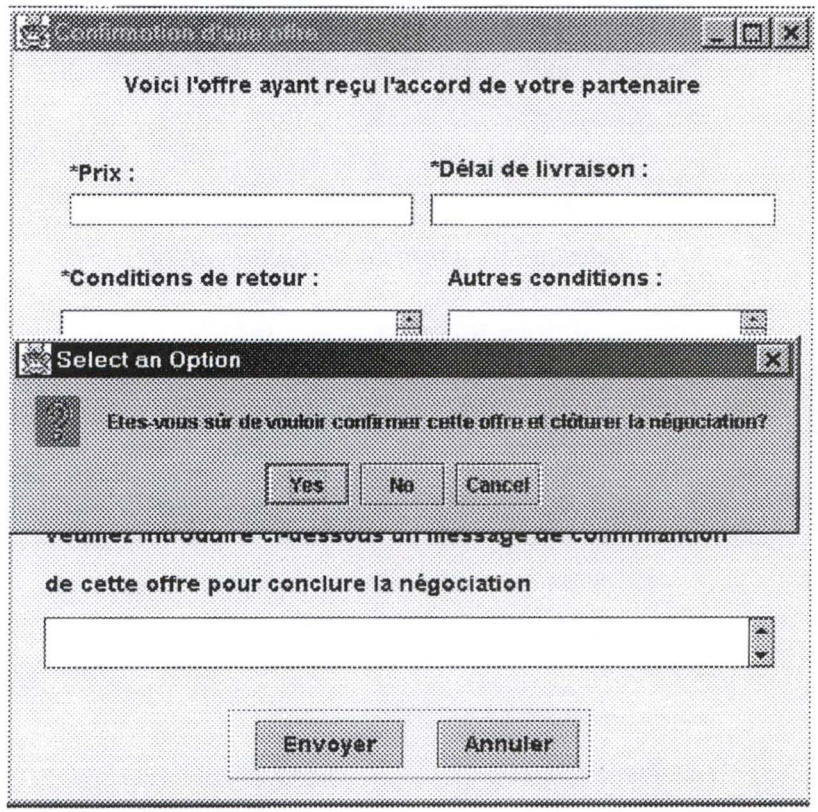
S'il répond "Yes", il reçoit le message suivant



Formulaire de confirmation d'une offre



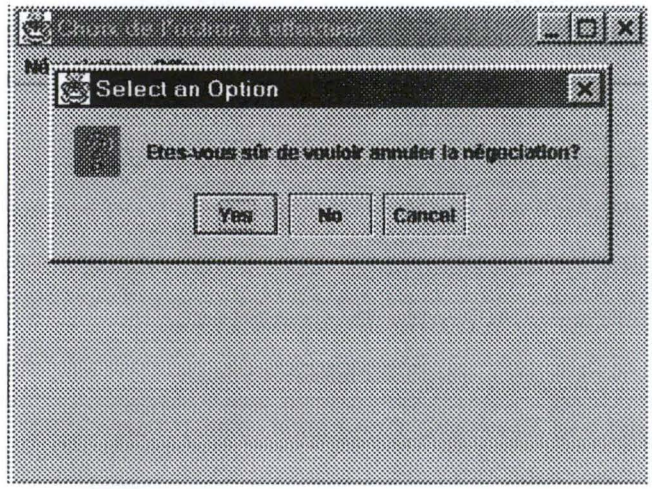
Message reçu par un utilisateur lorsqu'il appuie sur le bouton "Envoyer"



Si l'utilisateur répond "Yes" il reçoit le message suivant



Message reçu lorsqu'un utilisateur veut abandonner la négociation



Références

[1] Alexander Runge, The Need for Supporting Electronic commerce with Electronic Contracting. Media and Communications Management Institute, In: Schmid, Beat F.; Selz, Dorian; Sing, Regine: EM - Electronic Transactions. EM – Electronic Markets, Vol. 8, No. 1, 05/98 URL : http://www.businessmedia.org/netacademy/publications.nsf/all_pk/799

[2] C. Beam, A. Segev, and J. G. Shanthikumar, Electronic Negotiation through Internet-based Auctions, CMIT Working Paper 96-WP-1019. December 1996, URL : http://www.haas.berkeley.edu/~citm/OFFER/index.html

[3] Auctions and Bargaining in Electronic Commerce, The Fisher Center for Management and Information Technology, Sept, 1998. URL : http://www.haas.berkeley.edu/citm/auction/index.html

[4] Caroline Beam, Auctioning and Bidding in Electronic Commerce : The Online Auction, University of Berkeley, http://www.haas.berkeley.edu/~citm/, (dissertation).

[6] William N. Robinson, 1994 http://cis.gsu.edu/~wrobinso/negotiation.html

[7] Carrie Beam and Arie Segev (1997), Automated Negotiations : A survey of the State of the Art. Fisher Center for Information Technology & Management. Walter A. Haas School of Business, University of California, Berkeley; http://haas. Berkeley.edu/~citm/nego-proj.html

[8] Tung Bui, Designing Multiple Criteria Negotiation Support Systems : Frameworks, Issues and Implementation

[9] Going, Going, Gone! A survey of Auction Types, Copyright © 1996, Agorics, Inc. http://www.agorics.com.new.html

[10] Types of Internet Auctions, http://www-ics.ee.ic.ac.uk/surp99/article1/as

[11] R. Borovoy and R. H. Guttman, Auction Theory, Market-based Control, http://belledonna.media.mit.edu/people/guttman/research/commerce/talk3/master

[12] R. Julia Barcelo, E. Montero et A. Salaün, La proposition de directive européenne sur le commerce électronique : Questions choisies. Cahiers du CRID, n° 17.

[13] Robert W. Blaming et Tung X. Bui, Decision support and Internet Commerce, HandBook on Internet commerce, ?

[14] Archer, Norm; Rose, Joseph B.; Suarga, Mr.; Yuan, Yufei : A Web-Based Negotiation Support System. In : Schmid, Beat F.; Selz, Dorian; Sing, Regine : EM - Electronic Contracting. EM - Electronic Markets, Vol. 8, No. 3, 10/98. URL : http://www.businessmedia.org/netacademy/publications.nsf/all_pk/1076>.

[15] Hyacinth S. Nwana, Software Agents : An Overview, Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996. Cambridge University Press

URL : http://www.labs.bt.com/projects/agents/publish/papers/review2.htm

[16] P. Maes, R. H. Guttman and A.G. Moukas, Agents that Buy and Sell : Transforming Commerce as we Know It. Software Agents Group, MIT Media Laboratory, 1998 http://ecommerce.media.mit.edu

[17] Dajun Zeng and Katia Sycara, Bayesian Learning in Negotiation. The Robotics Institute, International journal of Human Computer Systems, Vol. 48, 1998, pp. 125-141

[18] Tung Bui, Building agent-based corporate information systems : an application to telemedecine, European Journal of Operational Research, article No. 4011, September 1999, Elsevier Science B.V.

[20] Melvin F. Shakun, Consciousness, Spirituality and Right Decision/Negociation in Purposeful Complex Adaptative Systems, Group Decision and Negociation 8 : 1-15, 1999, Kluwer Publishers.

[21] Melvin F. Shakun and Tung X. Bui Negotiation Process, Evolutionary Systems Design and NEGOCIATOR, URL : http://www.vacets.org/vtic97/txbui.htm

[22] Garett Dworman, URL : http://www.haas.berkeley.edu/citm/offer/

[23] Dajun Zeng and Katia Sycara, Coordination of Multiple Intelligent Software Agents, International Journal of Cooperative Information Systems, 1996.

[24] Jim R. Oliver, A machine Learning Approach to Automated negotiation and Prospects for electronic Commerce, http://opim.wharton.upenn.edu/õliver27/papers/jmis.ps, July 31, 1996.

[25] Tuomas Sandholm, and Victor Lesser, Automated Negotiation and Electronic Commerce : Extending the Contract Net Framework, First international Conference on Multiagent Systems, San Francisco, 1995.

[26] Tuomas Sandholm, An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations, Eleven National Conference on Artificial Intelligence (AAAI-93), Washington DC, pp. 256-262, 1993.

[27] Garett DwormanSteven Kimbrough, and James Laing, Bargaining by Artificial Agents in Two Coalition Games : A Study in Genetic Programming for Electronic commerce, Proceedings of the AAAI Genetic Programming Conference, Stanford, CA, August 1996. http://opim.wharton.upenn.edu/~dworman/.

[28] Garett DwormanSteven Kimbrough, and James Laing, On Automated Discovery of Models Using Genetic Programming in Game Theoric Contexts, Journal of Management Information Systems, vol.12(3), Winter 1996.

[29] Melvin F. Shakun, Evolutionary System Design : Policy Making Under complexity and Group Decision support Systems, Holden-Day, CA.

[30] Pattie Maes, Robert H. Guttman, Alexandros G.Moukas, Agents that Buy and Sell : Transformaing commerce as we Know It, Software Agents Group, MIT Media Laboratory, http://ecommerce.media.mit.edu

[31] A. Chavez, D. Dreilinger, R. Guttman, and P. Maes, A Real-Life Experiment in Creating an Agent marketplace, Proceedings of the Second International Conference on Electronic Commerce

[32] M. J. Osborne and A. Rubinstein, A course in game theory, the MIT Press, 1994.

[33] J. Rosenschein, and G. Zlotkin, Rules of Encounter, Cambridge, Mass. : MIT Press, 1994,

[34] D. P. Bertsekas, Dynamic Programming and Optimal Control. Belmont, MA : Athena Scientific, 1995, URL : http://www.athenasc.com/dpbook.html.

[35] R. M. Cyert, and M. H. DeGroot, Bayesian analysis and uncertainty in economic theory, Totowa, N.J. : Rowman & Littlefield, 1987,

URL : http://www.weyrich.com/book_reviews/bayesian_economic.html

[36] Jong-Jin Jung, Geun-Sik Jo, Brokerage between buyer and seller agents using constraint Satisfaction Problem models, Decision Support Systems and Electronic Commerce, vol. 28 (2000), Number 4, June 2000.

[37] Ting-Peng Liang, and Jin-Shiang Huang, A framework for applying intelligent agents to support electronic trading, Decision Support Systems and Electronic Commerce, vol. 28 (2000), Number 4, June 2000.

[38] Hyacinth S. Nwana & Divine T. Ndumu, "A Perspective on Software Agents Research", The Knowledge Engineering Review, Vol 14, No 2, pp. 1-18., 1999

[39] Mike Wooldridge and Nick Jennings, "The Pittfalls of Agent-Oriented Development", proceedings of the Second Conference on Autonomous Agents (Agents'98), May 1998

[40] Nicholas R. Jennings and Michael J. Wooldridge, Applications Of Intelligent Agents, in Nicholas R. Jennings and Michael J. Wooldridge (Ed.), Agent Technology Foundations, Applications, and Markets , Springer-Verlag, 1998

[41] Jeffrey Bradshaw, Introduction to Software Agents, in "Software Agents",AAAI Press/The MIT Press, 1997